Project 1: Report

Project Title: Demographic Patterns and Home Loan Clustering: A K-means Analysis in Washington State

1. Project Objectives | Problem Statements 1.1. PO1 | PS1: Segmentation of Consumer Data using Unsupervised Machine Learning Clustering Algorithms 1.2. PO2 | PS2: Identification of Appropriate Number of Segments or Clusters 1.3. PO3 | PS3: Determination of Segment or Cluster Characteristics

2. **Description of Data** 2.1. **Data Source, Size, Shape 2.1.1. Data Source (Website Link)** https://www.kaggle.com/datasets/miker400/washington-state-home-mortgage-hdma2016

** 2.1.2. Data Size (in KB | MB | GB …)**

30 MB

```
**2.1.3. Data Shape (Dimension: Number of Variables | Number of Records)**
```

37 Variables (37 columns) Maximum number of rows 60,000 Total number of records: 60000 Total number of filled cells: 1918780 Missed cells: 301220

2.2. Description of Variables 2.2.1. Index Variable(s): I1, I2, … The index is the sequence_number. This column has 39340 unique count of variables

2.2.2. Variables or Features having Categories | Categorical Variables or Features (CV) Categorical Columns: ['state_name', 'state_abbr', 'respondent_id', 'purchaser_type_name', 'property_type_name', 'preapproval_name', 'owner_occupancy_name', 'msamd_name', 'loan_type_name', 'loan_purpose_name', 'lien_status_name', 'hoepa_status_name', 'edit_status_name', 'denial_reason_name_3', 'denial_reason_name_2', 'denial_reason_name_1', 'county_name', 'co_applicant_sex_name', 'co_applicant_ethnicity_name', 'applicant_sex_name', 'applicant_ethnicity_name', 'agency_name', 'agency_abbr', 'action_taken_name']

Non-Categorical Columns: ['tract_to_msamd_income', 'rate_spread', 'population', 'minority_population', 'number_of_owner_occupied_units', 'number_of_1_to_4_family_units', 'loan_amount_000s', 'hud_median_family_income', 'applicant_income_000s', 'sequence_number', 'census_tract_number', 'as_of_year', 'application_date_indicator']

2.2.2.1. Variables or Features having Nominal Categories | Categorical Variables or Features - Nominal Type: CNV1, CNV2, … All the categorical variables available in the dataset for nominal variables

2.2.2.2. Variables or Features having Ordinal Categories | Categorical Variables or Features - Ordinal Type: COV1, COV2, … No ordinal data available in the dataset

2.2.3. Non-Categorical Variables or Features: NCV1, NCV2, … Non-Categorical Columns: ['tract_to_msamd_income', 'rate_spread', 'population', 'minority_population', 'number_of_owner_occupied_units', 'number_of_1_to_4_family_units', 'loan_amount_000s',

'hud_median_family_income', 'applicant_income_000s', 'sequence_number', 'census_tract_number', 'as_of_year', 'application_date_indicator']

2.3. Descriptive Statistics 2.3.1. Descriptive Statistics: Categorical Variables or Features **2.3.1.1.** Count | Frequency Statistics

Variable	Count	Unique
state_name	60000	1
state_abbr	60000	1
respondent_id	60000	593
purchaser_type_name	60000	10
property_type_name	60000	3
preapproval_name	60000	3
owner_occupancy_name	60000	3
msamd_name	51619	14
loan_type_name	60000	4
loan_purpose_name	60000	3
lien_status_name	60000	4
hoepa_status_name	60000	2
county_name	59908	39
co_applicant_sex_name	60000	5
co_applicant_ethnicity_name	60000	5
applicant_sex_name	60000	4
applicant_ethnicity_name	60000	4
agency_name	60000	6
agency_abbr	60000	6
action_taken_name	60000	8

Variable	Top Value
state_name	Washington
state_abbr	WA
respondent_id	32489
purchaser_type_name	Loan was not originated or was not sold in calendar year covered by register
property_type_name	One-to-four family dwelling (other than manufactured housing)
preapproval_name	Not applicable
owner_occupancy_name	Owner-occupied as a principal dwelling
msamd_name	Seattle, Bellevue, Everett - WA

Variable	Top Value
loan_type_name	Conventional
loan_purpose_name	Refinancing
lien_status_name	Secured by a first lien
hoepa_status_name	Not a HOEPA loan
county_name	King County
co_applicant_sex_name	No co-applicant
co_applicant_ethnicity_name	No co-applicant
applicant_sex_name	Male
applicant_ethnicity_name	Not Hispanic or Latino
agency_name	Department of Housing and Urban Development
agency_abbr	HUD
action_taken_name	Loan originated

Variable	Frequency
state_name	60000
state_abbr	60000
respondent_id	5006
purchaser_type_name	16112
property_type_name	57630
preapproval_name	47832
owner_occupancy_name	53940
msamd_name	17965
loan_type_name	42917
loan_purpose_name	28576
lien_status_name	57046
hoepa_status_name	59996
county_name	12915
co_applicant_sex_name	26987
co_applicant_ethnicity_name	26987
applicant_sex_name	37070
applicant_ethnicity_name	44014
agency_name	27514
agency_abbr	27514
action_taken_name	55815

In the context of catdf dataset: state_name and state_abbr: These columns have only one unique value, which is "Washington" for state_name and "WA" for state_abbr. This suggests that these columns may not provide much information for analysis as they have constant values for all rows.

respondent_id: This column has 593 unique values, and the most frequent respondent_id is "32489" with a frequency of 5006. This column likely identifies different respondents.

Other categorical columns: Each column represents a categorical variable, and the summary provides information about the number of unique categories, the most frequent category (top), and its frequency.

action_taken_name: This column represents the action taken for the loan application. It has 8 unique values, and "Loan originated" is the most frequent action with a frequency of 55815.

2.3.1.2. Proportion (Relative Frequency) Statistics

Variable	Percentage
state_name	Washington: 100.00%
state_abbr	WA: 100.00%
respondent_id	32489: 8.34%
purchaser_type_name	Loan was not originated or was not sold in calendar year covered by register: 79.72%
property_type_name	One-to-four family dwelling (other than manufactured housing): 100.00%
preapproval_name	Not applicable: 79.72%
owner_occupancy_name	Owner-occupied as a principal dwelling: 89.90%
msamd_name	Seattle, Bellevue, Everett - WA: 34.80%
loan_type_name	Conventional: 71.53%
loan_purpose_name	Refinancing: 47.63%
lien_status_name	Secured by a first lien: 95.08%
hoepa_status_name	Not a HOEPA loan: 99.99%
county_name	King County: 21.56%
co_applicant_sex_name	No co-applicant: 44.98%
co_applicant_ethnicity_name	No co-applicant: 44.98%
applicant_sex_name	Male: 61.78%
applicant_ethnicity_name	Not Hispanic or Latino: 73.36%
agency_name	Department of Housing and Urban Development: 41.19%
agency_abbr	HUD: 45.86%
action_taken_name	Loan originated: 93.03%

2.3.2. Descriptive Statistics: Non-Categorical Variables or Features 2.3.2.1. Measures of Central Tendency and Dispersion

Variable	Count/ Mean/ Std	Variable	Count/ Mean/ Std
tract_to_msamd_income	59878	number_of_1_to_4_family_units	59878
	Mean: 107.617351		Mean: 1873.281456
	Std: 28.233471		Std: 738.505184
	Min: 14.050000		Min: 27.000000
	25%: 88.970001		25%: 1414.000000
	50%: 105.550003		50%: 1770.000000
	75%: 123.330002		75%: 2249.000000
	Max: 257.140015		Max: 5893.000000
population	59878	loan_amount_000s	60000
	Mean: 5278.782157		Mean: 291.358717
	Std: 1716.101490		Std: 604.958183
	Min: 98.000000		Min: 1.000000
	25%: 4070.000000		25%: 170.000000
	50%: 5145.000000		50%: 242.000000
	75%: 6382.000000		75%: 337.000000
	Max: 13025.000000		Max: 55000.000000
minority_population	59878	hud_median_family_income	59878
	Mean: 23.244442		Mean: 73869.411136
	Std: 14.416209		Std: 12811.243390
	Min: 2.040000		Min: 48700.000000
	25%: 12.950000		25%: 63100.000000

Variable	Count/ Mean/ Std	Variable	Count/ Mean/ Std
	50%: 19.420000		50%: 73300.000000
	75%: 29.680000		75%: 90300.000000
	Max: 94.790001		Max: 90300.000000
number_of_owner_occupied_units	59876	applicant_income_000s	53630
	Mean: 1399.044375		Mean: 112.822301
	Std: 518.330561		Std: 122.862496
	Min: 15.000000		Min: 1.000000
	25%: 1034.000000		25%: 61.000000
	50%: 1359.000000		50%: 89.000000
	75%: 1722.000000		75%: 132.000000
	Max: 2997.000000		Max: 6161.000000
sequence_number	60000	census_tract_number	59878
	Mean: 77526.47		Mean: 1750.597252
	Std: 150515.7		Std: 3359.676740
	Min: 1.000000		Min: 1.000000
	25%: 3231.500000		25%: 114.020000
	50%: 16481.000000		50%: 403.020000
	75%: 72762.25		75%: 713.100000
	Max: 1241590.0		Max: 9757.000000
as_of_year	60000	application_date_indicator	60000
	Mean: 2016.0		Mean: 0.025867
	Std: 0.0		Std: 0.225976

Variable	Count/ Mean/ Std	Variable	Count/ Mean/ Std
	Min: 2016.0		Min: 0.000000
	25%: 2016.0		25%: 0.000000
	50%: 2016.0		50%: 0.000000
	75%: 2016.0		75%: 0.000000
	Max: 2016.0		Max: 2.000000

2.3.2.3. Correlation Statistics (with Test of Correlation)

3. Analysis of Data 3.1. Data Pre-Processing 3.1.1. Missing Data Statistics and Treatment **3.1.1.1.1.** Missing Data Statistics: Records Number of rows with missing data: 60000 Number of rows with more than 50% missing data: 0

3.1.1.1.2. Missing Data Treatment: Records 3.1.1.1.2.1. Removal of Records with More Than 50% Missing Data: None | R1, R2, … No rows with more than 50% missing values

3.1.1.2.1. Missing Data Statistics: Categorical Variables or Features

Feature	Missing_Records	Percentage_Missing
denial_reason_name_3	59999	99.998333
denial_reason_name_2	59957	99.928333
denial_reason_name_1	59882	99.803333
rate_spread	58134	96.890000
edit_status_name	47549	79.248333
msamd_name	8381	13.968333
applicant_income_000s	6370	10.616667
number_of_owner_occupied_units	124	0.206667
census_tract_number	122	0.203333
tract_to_msamd_income	122	0.203333
hud_median_family_income	122	0.203333
number_of_1_to_4_family_units	122	0.203333
minority_population	122	0.203333
population	122	0.203333
county_name	92	0.153333

3.1.1.2.2. Missing Data Treatment: Categorical Variables or Features

3.1.1.2.2.1. Removal of Variables or Features with More Than 50% Missing Data: None | CV1, CV2, … Removed the below columns as they have more than 50% data missing • denial_reason_name_3 • denial_reason_name_2 • denial_reason_name_1 • rate_spread (non-cat) • edit_status_name

3.1.1.2.2.2. Imputation of Missing Data using Descriptive Statistics: Mode

3.1.1.3.1. Missing Data Statistics: Non-Categorical Variables or Features

Feature	Missing_Records
tract_to_msamd_income	122
population	122
minority_population	122
number_of_owner_occupied_units	124
number_of_1_to_4_family_units	122
loan_amount_000s	0
hud_median_family_income	122
applicant_income_000s	6370
sequence_number	0
census_tract_number	122
as_of_year	0
application_date_indicator	0

3.1.1.3.2. Missing Data Treatment: Non-Categorical Variables or Features 3.1.1.3.2.1. Removal of Variables or Features with More Than 50% Missing Data: None | NCV1, NCV2, … • rate_spread

3.1.1.3.2.2. Imputation of Missing Data using Descriptive Statistics: Mean | Median • Imputing the missing values using mean

3.1.2. Numerical Encoding of Categorical Variables or Features (Encoding Schema - Alphanumeric Order)

Feature	Number_of_Unique_Values
state_name	1
state_abbr	1
respondent_id	593
purchaser_type_name	10
property_type_name	3
preapproval_name	3
owner_occupancy_name	3
msamd_name	14
loan_type_name	4
loan_purpose_name	3

Feature	Number_of_Unique_Values
lien_status_name	4
hoepa_status_name	2
county_name	39
co_applicant_sex_name	5
co_applicant_ethnicity_name	5
applicant_sex_name	4
applicant_ethnicity_name	4
agency_name	6
agency_abbr	6
action_taken_name	8

We are converting the above variables into numeric format in the alpha numeric order

3.1.3. Outlier Statistics and Treatment (Scaling | Transformation) 3.1.3.1.1. Outlier Statistics: Non-Categorical Variables or Features

Outliers count for the Non-Categorical Variables

Variable	Count
tract_to_msamd_income	1309
population	553
minority_population	2641
number_of_owner_occupied_units	478
number_of_1_to_4_family_units	2150
loan_amount_000s	2467
hud_median_family_income	0
applicant_income_000s	3765
sequence_number	7898
census_tract_number	9391
as_of_year	0
application_date_indicator	776

3.1.3.1.2. Outlier Treatment: Non-Categorical Variables or Features 3.1.3.1.2.1. Standardization: OV1, OV2, … 3.1.3.1.2.2. Normalization using Min-Max Scaler: OV3, OV4, … 3.1.3.1.2.3. Log Transformation: OV5, OV6, … I performed scaling using normalization using min-max scaler. But post the scaling, bubbles were still visible in the box plot. This signifies that the outliers present in the non categorical datset are not heavily influenced by the scaling method. The count of outliers seems consistent across different scaling methods.

3.1.4. Data Bifurcation: Training & Testing Sets (Not Required)

3.2. Data Analysis 3.2.1.1. PO1 | PS1:: Unsupervised Machine Learning Clustering Algorithm: K-Means (Base Model) | Metrics Used - Euclidean Distance 3.2.2.1.1. PO2 | PS2:: Clustering Model Performance Evaluation: Silhouette Score | Davies-Bouldin Score (Base Model: K-Mean)

K = 2 Davies-Bouldin Index (DBI): Interpretation: The Davies-Bouldin Index measures the compactness and separation between clusters. Lower DBI values are better: A lower DBI indicates better clustering. It is minimized when clusters are compact (small intra-cluster distances) and well-separated (large inter-cluster distances). Range: The DBI value is non-negative, and lower values are generally better, with 0 being the optimal score. A DBI of 0.462 suggests moderate compactness and separation in your clusters.

Silhouette Score (SS): Interpretation: The Silhouette Score measures how well-defined and separated the clusters are. Range: The SS ranges from -1 to 1. Higher values indicate better-defined clusters. Interpretation of values: Close to +1: Indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. Close to 0: Indicates overlapping clusters. Close to -1: Indicates that the object is poorly matched to its own cluster and may be a member of a neighboring cluster. An SS of 0.663 suggests well-defined clusters with relatively good separation.

The **cluster profiling** report provides a detailed overview of two clusters (Cluster 0 and Cluster 1) obtained through K-Means clustering with k=2. The clustering is based on various variables, and this report aims to characterize each cluster based on the descriptive statistics of significant variables.

Cluster 0: Overview: Number of Observations: 38,597 Dominant Characteristics: Lower values in key variables such as tract_to_msamd_income, minority_population, number_of_owner_occupied_units, number_of_1_to_4_family_units, loan_amount_000s, applicant_income_000s, sequence_number, census_tract_number, hud_median_family_income. Higher value in application_date_indicator.

Characteristic Variables: tract_to_msamd_income: Lower minority_population: Lower number_of_owner_occupied_units: Lower number_of_1_to_4_family_units: Lower loan_amount_000s: Lower applicant_income_000s: Lower sequence_number: Lower census_tract_number: Lower hud_median_family_income: Lower application_date_indicator: Higher

Cluster 1: Overview: Number of Observations: 21,403 Dominant Characteristics: Higher values in key variables such as tract_to_msamd_income, minority_population, number_of_owner_occupied_units, number_of_1_to_4_family_units, loan_amount_000s, applicant_income_000s, sequence_number, census_tract_number, hud_median_family_income. Lower value in application_date_indicator.

Characteristic Variables: tract_to_msamd_income: Higher minority_population: Higher number_of_owner_occupied_units: Higher number_of_1_to_4_family_units: Higher loan_amount_000s: Higher applicant_income_000s: Higher sequence_number: Higher census_tract_number: Higher hud_median_family_income: Higher application_date_indicator: Lower

Notes: state_name and state_abbr: These variables have a chi-square p-value of 1.0, suggesting that they might not be strong differentiators between clusters. as_of_year: It seems to have a constant value for both clusters, possibly not providing useful information for clustering.

Conclusion: The clusters exhibit distinct characteristics in terms of socio-economic and loan-related variables. Further domain-specific analysis and business context consideration are recommended to enhance the understanding of the clusters' implications.

Cluster Profiling Report for K=3 Introduction: The **cluster profilin** report provides a detailed overview of three clusters (Cluster 0, Cluster 1, and Cluster 2) obtained through K-Means clustering with k=3. The clustering is based on various variables, and this report aims to characterize each cluster based on the descriptive statistics of significant variables.

Cluster 0: Overview: Number of Observations: 26,674 Dominant Characteristics: Lower values in key variables such as tract_to_msamd_income, population, minority_population, number_of_owner_occupied_units, number_of_1_to_4_family_units, loan_amount_000s, applicant_income_000s, sequence_number, census_tract_number, hud_median_family_income. Higher value in application_date_indicator.

Characteristic Variables: tract_to_msamd_income: Lower population: Lower minority_population: Lower number_of_owner_occupied_units: Lower number_of_1_to_4_family_units: Lower loan_amount_000s: Lower applicant_income_000s: Lower sequence_number: Lower census_tract_number: Lower hud_median_family_income: Lower application_date_indicator: Higher

Cluster 1: Overview: Number of Observations: 15,361 Dominant Characteristics: Higher values in key variables such as tract_to_msamd_income, population, minority_population, number_of_owner_occupied_units, number_of_1_to_4_family_units, loan_amount_000s, applicant_income_000s, sequence_number, census_tract_number, hud_median_family_income. Lower value in application_date_indicator.

Characteristic Variables: tract_to_msamd_income: Higher population: Higher minority_population: Higher number_of_owner_occupied_units: Higher number_of_1_to_4_family_units: Higher loan_amount_000s: Higher applicant_income_000s: Higher sequence_number: Higher census_tract_number: Higher hud_median_family_income: Higher application_date_indicator: Lower

Cluster 2: Overview: Number of Observations: 17,965 Dominant Characteristics: Moderate values in key variables. Comparable characteristics to both Cluster 0 and Cluster 1.

Characteristic Variables: The cluster exhibits characteristics between Cluster 0 and Cluster 1.

Notes: as_of_year: It seems to have a constant value for all clusters, possibly not providing useful information for clustering.

Conclusion: The clusters exhibit distinct characteristics in terms of socio-economic and loan-related variables. Cluster 2 serves as an intermediate group with features falling between Cluster 0 and Cluster 1. Further domain-specific analysis and business context consideration are recommended to enhance the understanding of the clusters' implications.

Cluster Profiling Report for K=4 Introduction: This **cluster profiling** report provides a detailed analysis of the characteristics of each cluster obtained through K-Means clustering with k=4. The clusters are defined based on various variables, and this report aims to highlight the distinctive features of each cluster.

Cluster 0: Overview: Number of Observations: 8,935 Dominant Characteristics: Moderate values in key variables. Comparable characteristics to Clusters 1 and 2. Lower values in hud_median_family_income compared to other clusters.

Characteristic Variables: tract_to_msamd_income: Moderate population: Moderate minority_population: Moderate number_of_owner_occupied_units: Moderate number_of_1_to_4_family_units: Moderate loan_amount_000s: Moderate applicant_income_000s: Moderate sequence_number: Moderate census_tract_number: Moderate hud_median_family_income: Lower application_date_indicator: Moderate

Cluster 1: Overview: Number of Observations: 10,205 Dominant Characteristics: Higher values in key variables compared to Cluster 0. Comparable characteristics to Clusters 2 and 3.

Characteristic Variables: tract_to_msamd_income: Higher population: Higher minority_population: Higher number_of_owner_occupied_units: Higher number_of_1_to_4_family_units: Higher loan_amount_000s: Higher applicant_income_000s: Higher sequence_number: Higher census_tract_number: Higher hud_median_family_income: Moderate application_date_indicator: Moderate

Cluster 2: Overview: Number of Observations: 17,965 Dominant Characteristics: Moderate values in key variables. Comparable characteristics to Clusters 0 and 1.

Characteristic Variables: tract_to_msamd_income: Moderate population: Moderate minority_population: Moderate number_of_owner_occupied_units: Moderate number_of_1_to_4_family_units: Moderate loan_amount_000s: Moderate applicant_income_000s: Moderate sequence_number: Moderate census_tract_number: Moderate hud_median_family_income: Moderate application_date_indicator: Moderate

Cluster 3: Overview: Number of Observations: 22,895 Dominant Characteristics: Higher values in key variables compared to Clusters 0 and 2. Higher values in hud_median_family_income.

Characteristic Variables: tract_to_msamd_income: Higher population: Higher minority_population: Higher number_of_owner_occupied_units: Higher number_of_1_to_4_family_units: Higher loan_amount_000s: Higher applicant_income_000s: Higher sequence_number: Higher census_tract_number: Higher hud_median_family_income: Higher application_date_indicator: Moderate

Notes: as_of_year: It seems to have a constant value for all clusters, possibly not providing useful information for clustering.

Conclusion: The clusters exhibit distinct characteristics in terms of socio-economic and loan-related variables. Each cluster represents a different profile of observations, providing valuable insights into the underlying patterns within the data. Further domain-specific analysis and business context consideration are recommended to enhance the understanding of the clusters' implications.

Cluster Profiling Report for K=5 Introduction: This **cluster profiling** report provides a comprehensive analysis of the characteristics of each cluster obtained through K-Means clustering with k=5. The clusters are defined based on various variables, and this report aims to highlight the distinctive features of each cluster.

Cluster 0: Overview: Number of Observations: 8,935 Dominant Characteristics: Moderate values in key variables. Lower values in hud_median_family_income. Comparable characteristics to Clusters 1 and 2.

Characteristic Variables: tract_to_msamd_income: Moderate population: Moderate minority_population: Moderate number_of_owner_occupied_units: Moderate number_of_1_to_4_family_units: Moderate loan_amount_000s: Moderate applicant_income_000s: Moderate sequence_number: Moderate census_tract_number: Moderate hud_median_family_income: Lower application_date_indicator: Moderate

Cluster 1: Overview: Number of Observations: 10,205 Dominant Characteristics: Higher values in key variables compared to Cluster 0. Comparable characteristics to Clusters 2 and 3.

Characteristic Variables: tract_to_msamd_income: Higher population: Higher minority_population: Higher number_of_owner_occupied_units: Higher number_of_1_to_4_family_units: Higher loan_amount_000s: Higher applicant_income_000s: Higher sequence_number: Higher census_tract_number: Higher hud_median_family_income: Moderate application_date_indicator: Moderate

Cluster 2: Overview: Number of Observations: 17,965 Dominant Characteristics: Moderate values in key variables. Comparable characteristics to Clusters 0 and 1.

Characteristic Variables: tract_to_msamd_income: Moderate population: Moderate minority_population: Moderate number_of_owner_occupied_units: Moderate number_of_1_to_4_family_units: Moderate loan_amount_000s: Moderate applicant_income_000s: Moderate sequence_number: Moderate census_tract_number: Moderate hud_median_family_income: Moderate application_date_indicator: Moderate

Cluster 3: Overview: Number of Observations: 19,457 Dominant Characteristics: Higher values in key variables compared to Clusters 0 and 2. Higher values in hud_median_family_income. Comparable characteristics to Cluster 4. Characteristic Variables: tract_to_msamd_income: Higher population: Higher minority_population: Higher number_of_owner_occupied_units: Higher number_of_1_to_4_family_units: Higher loan_amount_000s: Higher applicant_income_000s: Higher sequence_number: Higher census_tract_number: Higher hud_median_family_income: Higher application_date_indicator: Moderate

Cluster 4: Overview: Number of Observations: 3,438 Dominant Characteristics: Lower values in key variables compared to other clusters. Lower values in hud_median_family_income. Comparable characteristics to Cluster 0.

Characteristic Variables: tract_to_msamd_income: Lower population: Lower minority_population: Lower number_of_owner_occupied_units: Lower number_of_1_to_4_family_units: Lower loan_amount_000s: Lower applicant_income_000s: Lower sequence_number: Lower census_tract_number: Lower hud_median_family_income: Lower application_date_indicator: Lower

Notes: as_of_year: It seems to have a constant value for all clusters, possibly not providing useful information for clustering.

Conclusion: The clusters exhibit distinct characteristics in terms of socio-economic and loan-related variables. Each cluster represents a different profile of observations, providing valuable insights into the underlying patterns within the data. Further domain-specific analysis and

business context consideration are recommended to enhance the understanding of the clusters' implications. The addition of Cluster 4 introduces a group with lower values across various features, offering a more diversified segmentation.

3.2.2.1.2. PO2 | PS2:: Clustering Model Performance Evaluation: Time Statistics | (CPU | GPU) Memory Statistics (Base Model: K-Mean)

K	Time	Peak Memory
2	67s	408.77 MiB
3	47.3s	432.41 MiB
4	56.9s	443.09 MiB
5	46.2s	447.80 MiB

3.2.3.1. PO3 | PS3:: Cluster Analysis: Base Model (K-Means) 3.2.3.1.1. Cluster Analysis with Categorical Variables or Features: Chi-Square Test of Independence

Categorical Variable	Chi-Sq Value
state_name	0
state_abbr	0
respondent_id	37667.00325
purchaser_type_name	3421.738007
property_type_name	1313.264545
preapproval_name	494.6907302
owner_occupancy_name	326.8773249
msamd_name	187202.447
loan_type_name	4193.389894
loan_purpose_name	209.9763256
lien_status_name	3332.387181
hoepa_status_name	2.423712105
county_name	239169.2752
co_applicant_sex_name	2320.820989
co_applicant_ethnicity_name	2472.609477
applicant_sex_name	988.6046171
applicant_ethnicity_name	1517.789761
agency_name	4859.037635
agency_abbr	4859.037635
action_taken_name	8254.230514

3.2.3.1.2. Cluster Analysis with Non-Categorical Variables or Features: Analysis of Variance (ANOVA)

Non-Categorical Variable	F-value	P-value
hud_median_family_income	1143184.92	0.0

Non-Categorical Variable	F-value	P-value
census_tract_number	7928.80	0.0
number_of_1_to_4_family_units	1808.53	0.0
applicant_income_000s	466.13	0.0
loan_amount_000s	276.42	6.69e-236
minority_population	1543.64	0.0
population	120.34	1.86e-102
tract_to_msamd_income	93.49	2.06e-79
number_of_owner_occupied_units	184.42	2.28e-157
sequence_number	65.08	5.16e-55
application_date_indicator	859.68	0.0

4. Results | Observations 4.1. Appropriate Number of Segments | Clusters: Base Model (K-Means)

K	SS	DBS
2	0.663	0.462
3	0.8003	0.2801
4	0.8316	0.3288
5	0.8465	0.2879

4.2. Cluster Size (Base Model | Comparison Models)

K	C0	C1	C2	C3	C4
2	38597	21403	-	-	-
3	26674	15361	17965	-	-
4	8935	10205	17965	22895	-
5	8935	10205	17965	19457	3438

4.3. Clustering Model Performance: Time & Memory Statistics [Base Model (K-Means)

K	time	peak memory
2	67s	408.77 MiB
3	47.3s	432.41 MiB
4	56.9s	443.09 MiB
5	46.2s	447.80 MiB

4.4. Cluster Analysis: Base Model (K-Means) 4.4.1. Categorical Variables or Features: Contributing or Significant | Non-Contributing or Non-Significant Significant Variable:**

Categorical Variable	Chi-Sq Value
state_name	0
state_abbr	0

Categorical Variable	Chi-Sq Value
respondent_id	37667.00325
purchaser_type_name	3421.738007
property_type_name	1313.264545
preapproval_name	494.6907302
owner_occupancy_name	326.8773249
msamd_name	187202.447
loan_type_name	4193.389894
loan_purpose_name	209.9763256
lien_status_name	3332.387181
hoepa_status_name	2.423712105
county_name	239169.2752
co_applicant_sex_name	2320.820989
co_applicant_ethnicity_name	2472.609477
applicant_sex_name	988.6046171
applicant_ethnicity_name	1517.789761
agency_name	4859.037635
agency_abbr	4859.037635
action_taken_name	8254.230514

Based on the Chi-Square value county_name is the most significant variable. Followed msamd_name, espondent_id, action_taken_name and agency_name are the next most significant variables in the same order. Meanwhile, applicant_sex_name, preapproval_name, owner_occupancy_name, loan_purpose_name, hoepa_status_name, state_name, and state_abbr are least significant in the clusters

4.4.2. Non-Categorical Variables or Features: Contributing or Significant | Non-Contributing or Non-Significant

Non-Significant Variable:

Non-Categorical Variable	F-value	P-value
hud_median_family_income	1143184.92	0.0
census_tract_number	7928.80	0.0
number_of_1_to_4_family_units	1808.53	0.0
applicant_income_000s	466.13	0.0
loan_amount_000s	276.42	6.69e-236
minority_population	1543.64	0.0
population	120.34	1.86e-102

Non-Categorical Variable	F-value	P-value
tract_to_msamd_income	93.49	2.06e-79
number_of_owner_occupied_units	184.42	2.28e-157
sequence_number	65.08	5.16e-55
application_date_indicator	859.68	0.0

As per the f and p value, hud_median_family_income, census_tract_number, number_of_1_to_4_family_unit and minority_population are the most significant non-categorical variable

5. Managerial Insights

5.1. Appropriate Number of Segments | Clusters (Given the Appropriate Model) Silhouette Scores (ss): The silhouette score measures how well-separated the clusters are. A higher silhouette score indicates better-defined clusters. Here, the silhouette score increases from k=2 to k=5, suggesting an improvement in cluster separation as k increases.

Davies-Bouldin Scores (dbs): The Davies-Bouldin index measures cluster compactness and separation. A lower Davies-Bouldin score indicates better clustering. Here, the Davies-Bouldin score decreases from k=2 to k=5, indicating improved cluster quality. Based on the ss and db scores for k=2,3,4 and 5, both the silhouette scores and Davies-Bouldin scores suggest that a higher value of k might be better. Given the increasing trend in silhouette scores and decreasing trend in Davies-Bouldin scores, we consider **k=5** for potentially better-defined and more compact clusters.

Cluster 0: Action Taken: The mean action taken by applicants is around 3.96, with a low standard deviation (0.28), indicating relatively consistent behavior within the cluster. Agency: The mean agency involvement is approximately 1.29, with a wide range of variation (standard deviation = 1.21). Loan Type: Predominantly type 0 and type 3 loans. Median Family Income: The median family income is around $63,375, with relatively low variance (standard deviation = $2,142). Number of 1 to 4 Family Units: The mean value is approximately 0.33, with some variation (standard deviation = 0.12).

Cluster 1: Action Taken: Similar mean action taken as Cluster 0, but with a higher standard deviation (0.85), indicating more variability. Agency: Mean agency involvement is lower compared to Cluster 0, with a similar standard deviation. Loan Type: Predominantly type 0 and type 3 loans, with some variability. Median Family Income: Lower median family income compared to Cluster 0, with slightly higher variance. Number of 1 to 4 Family Units: Higher mean value compared to Cluster 0, with higher variance.

Cluster 2: Action Taken: The mean action taken by applicants is around 3.99, with low standard deviation (0.20), indicating consistent behavior. Agency: Mean agency involvement is higher compared to Clusters 0 and 1, with low variability. Loan Type: Predominantly type 0 loans, with lower variability compared to other clusters. Median Family Income: Highest median family

income among clusters, with no variance. Number of 1 to 4 Family Units: Lower mean value compared to Clusters 0 and 1, with low variance.

Cluster 3: Action Taken: Mean action taken is slightly lower compared to Clusters 0 and 2, with moderate variability. Agency: Similar mean agency involvement as Cluster 1, but with slightly lower variability. Loan Type: Predominantly type 0 and type 3 loans, similar to other clusters. Median Family Income: Lower median family income compared to Clusters 0 and 2, with moderate variance. Number of 1 to 4 Family Units: Similar mean value as Cluster 0, with moderate variability.

Cluster 4: Action Taken: Similar mean action taken as Clusters 0 and 2, with low standard deviation. Agency: Mean agency involvement is lower compared to other clusters, with low variability. Loan Type: Predominantly type 1 loans, with lower variability compared to other clusters. Median Family Income: Similar median family income as Cluster 2, with no variance. Number of 1 to 4 Family Units: Lower mean value compared to other clusters, with low variance.

5.3. Identification of a 'Niche' Segment | Cluster (If Any)

Cluster 4 is a niche segment with 3438 records Cluster 4 emerges as a niche segment within the dataset due to its distinct characteristics and relatively smaller size compared to other clusters. This segment primarily comprises loans predominantly purchased by entities like "Ginnie Mae (GNMA)" and "Freddie Mac (FHLMC)". It is notable for its preference for conventional loans, particularly for one-to-four family dwellings, often utilized for refinancing or home purchases. Moreover, these loans are typically secured by a first lien and are mostly originated without the need for preapproval.

Demographically, the cluster represents properties that are primarily owner-occupied, with a relatively balanced distribution of male and female applicants, most of whom are non-Hispanic or Latino. Despite being a niche segment, the lending activities within this cluster are relatively successful, with a significant proportion resulting in loan origination. From a numerical standpoint, the cluster exhibits moderate economic indicators, such as tract-to-msamd-income ratios and median family incomes, suggesting stability within the localities represented. Overall, this cluster's unique composition and successful lending outcomes make it a distinctive niche within the broader lending landscape.

Managerial Insights: Tract to MSAMD Income and Population: Clusters 0, 1, and 2 have relatively similar average values for these variables, indicating similar socioeconomic conditions in these clusters. Clusters 3 and 4 have higher average values for both variables, suggesting potentially higher-income areas with larger populations.

Minority Population: Cluster 2 has the highest average minority population, indicating a cluster with a higher proportion of minority residents compared to other clusters.

Owner-Occupied Units and 1-4 Family Units: Clusters 3 and 4 have slightly higher average values for both variables, indicating potentially more stable housing conditions with higher rates of ownership.

Loan Amount and Applicant Income: Clusters 2 and 1 have slightly higher average loan amounts and applicant incomes compared to other clusters, indicating higher financial capability in these clusters.

Census Tract Number: Cluster 0 has the highest average census tract number, suggesting potentially more urbanized areas compared to other clusters.

HUD Median Family Income: Cluster 2 has the highest average HUD median family income, indicating potentially higher-income areas compared to other clusters.

Application Date Indicator: Cluster 0 has a significantly higher average application date indicator, suggesting a higher rate of applications within a certain time period compared to other clusters.

Categorical Variables: Each cluster has a dominant category for categorical variables such as purchaser type, property type, preapproval status, etc. These dominant categories can provide insights into the prevalent characteristics or preferences within each cluster.

Clusters with higher average incomes and housing stability (e.g., clusters 2 and 1) may present opportunities for targeting higher-value loan products or services. Understanding the demographics of each cluster (e.g., minority population percentage) can help tailor marketing strategies and product offerings to better serve specific communities. Clusters with higher application rates within a specific time period (e.g., cluster 0) may require additional attention in terms of processing efficiency and customer service. Analyzing dominant categories within each cluster can inform targeted marketing campaigns and product development initiatives tailored to the preferences and needs of each group.

Overall, leveraging these insights can enable financial institutions to better understand their customer base, optimize operations, and tailor their offerings to meet the diverse needs of different communities.

```
# Required Libraries
import pandas as pd, numpy as np # For Data Manipulation
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder # For
Encoding Categorical Data [Nominal | Ordinal]
from sklearn.preprocessing import OneHotEncoder # For Creating Dummy
Variables of Categorical Data [Nominal]
from sklearn.impute import SimpleImputer, KNNImputer # For Imputation
of Missing Data
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
RobustScaler # For Rescaling Data
from sklearn.model_selection import train_test_split # For Splitting
Data into Training & Testing Sets
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import pearsonr
from scipy import stats

# Required Libraries
import pandas as pd, numpy as np # For Data Manipulation
import matplotlib.pyplot as plt, seaborn as sns # For Data
Visualization
import scipy.cluster.hierarchy as sch # For Hierarchical Clustering
from sklearn.cluster import AgglomerativeClustering as agclus, KMeans
as kmclus # For Agglomerative & K-Means Clustering
```

```
from sklearn.metrics import silhouette_score as sscore,
davies_bouldin_score as dbscore # For Clustering Model Evaluation

# @title load library { display-mode: "form" }
# Load IPython extension for measuring time
!pip install ipython-autotime
%reload_ext autotime

# Load IPython extension for memory profiling
!pip install memory-profiler
%reload_ext memory_profiler

# Your imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering as agclus, KMeans
as kmclus
from sklearn.metrics import silhouette_score as sscore,
davies_bouldin_score as dbscore
from scipy.cluster.hierarchy import dendrogram, linkage
import plotly.graph_objects as go

# Load preprocessing libraries
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder,
OneHotEncoder
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
RobustScaler
from sklearn.model_selection import train_test_split

from scipy.stats import f_oneway

Collecting ipython-autotime
  Downloading ipython_autotime-0.3.2-py2.py3-none-any.whl (7.0 kB)
Requirement already satisfied: ipython in
/usr/local/lib/python3.10/dist-packages (from ipython-autotime)
(7.34.0)
Requirement already satisfied: setuptools>=18.5 in
/usr/local/lib/python3.10/dist-packages (from ipython->ipython-
autotime) (67.7.2)
Collecting jedi>=0.16 (from ipython->ipython-autotime)
  Downloading jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 7.2 MB/s eta
0:00:00
ent already satisfied: decorator in /usr/local/lib/python3.10/dist-
packages (from ipython->ipython-autotime) (4.4.2)
Requirement already satisfied: pickleshare in
```

```
/usr/local/lib/python3.10/dist-packages (from ipython->ipython-
autotime) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in
/usr/local/lib/python3.10/dist-packages (from ipython->ipython-
autotime) (5.7.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!
=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from
ipython->ipython-autotime) (3.0.43)
Requirement already satisfied: pygments in
/usr/local/lib/python3.10/dist-packages (from ipython->ipython-
autotime) (2.16.1)
Requirement already satisfied: backcall in
/usr/local/lib/python3.10/dist-packages (from ipython->ipython-
autotime) (0.2.0)
Requirement already satisfied: matplotlib-inline in
/usr/local/lib/python3.10/dist-packages (from ipython->ipython-
autotime) (0.1.6)
Requirement already satisfied: pexpect>4.3 in
/usr/local/lib/python3.10/dist-packages (from ipython->ipython-
autotime) (4.9.0)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from jedi>=0.16->ipython-
>ipython-autotime) (0.8.3)
Requirement already satisfied: ptyprocess>=0.5 in
/usr/local/lib/python3.10/dist-packages (from pexpect>4.3->ipython-
>ipython-autotime) (0.7.0)
Requirement already satisfied: wcwidth in
/usr/local/lib/python3.10/dist-packages (from prompt-toolkit!=3.0.0,!
=3.0.1,<3.1.0,>=2.0.0->ipython->ipython-autotime) (0.2.13)
Installing collected packages: jedi, ipython-autotime
Successfully installed ipython-autotime-0.3.2 jedi-0.19.1
Collecting memory-profiler
  Downloading memory_profiler-0.61.0-py3-none-any.whl (31 kB)
Requirement already satisfied: psutil in
/usr/local/lib/python3.10/dist-packages (from memory-profiler) (5.9.5)
Installing collected packages: memory-profiler
Successfully installed memory-profiler-0.61.0
time: 6.45 s (started: 2024-03-16 13:28:03 +00:00)
```

```python
import pandas as pd

# Specify the path to your CSV file
csv_file_path = "/content/Washington_State_HDMA-2016 - Copy -
Copy.csv"

# Read the CSV file into a pandas DataFrame
df = pd.read_csv(csv_file_path)

# Display the first few rows of the DataFrame to verify the data
print(df.head())
```

```
   tract_to_msamd_income  rate_spread  population  minority_population
\
0             121.690002          NaN      8381.0            23.790001

1              83.370003          NaN      4915.0            23.990000

2              91.129997          NaN      5075.0            11.820000

3             146.169998          NaN      5032.0             8.590000

4             162.470001          NaN      5183.0            10.500000


   number_of_owner_occupied_units  number_of_1_to_4_family_units  \
0                          2175.0                         2660.0
1                          1268.0                         1777.0
2                          1136.0                         1838.0
3                          1525.0                         1820.0
4                          1705.0                         2104.0

   loan_amount_000s  hud_median_family_income
applicant_income_000s  \
0               227                   73300.0                   116.0

1               240                   57900.0                    42.0

2               241                   73300.0                   117.0

3               351                   73300.0                   315.0

4               417                   78100.0                   114.0


   state_name  ... co_applicant_sex_name  \
0  Washington  ...                 Male
1  Washington  ...       No co-applicant
2  Washington  ...               Female
3  Washington  ...               Female
4  Washington  ...                 Male

                         co_applicant_ethnicity_name
census_tract_number  \
0                            Not Hispanic or Latino
413.27
1                                 No co-applicant
9208.01
2                            Not Hispanic or Latino
414.00
3  Information not provided by applicant in mail,...
405.10
4                            Not Hispanic or Latino
```

```
907.00

  as_of_year application_date_indicator applicant_sex_name  \
0       2016                          0             Female
1       2016                          0               Male
2       2016                          0               Male
3       2016                          0               Male
4       2016                          0             Female

                            applicant_ethnicity_name  \
0                             Not Hispanic or Latino
1                                 Hispanic or Latino
2                             Not Hispanic or Latino
3  Information not provided by applicant in mail,...
4                             Not Hispanic or Latino

                                  agency_name agency_abbr
action_taken_name
0         Consumer Financial Protection Bureau        CFPB   Loan
originated
1  Department of Housing and Urban Development         HUD   Loan
originated
2  Department of Housing and Urban Development         HUD   Loan
originated
3         National Credit Union Administration        NCUA   Loan
originated
4        Federal Deposit Insurance Corporation        FDIC   Loan
originated

[5 rows x 37 columns]
time: 633 ms (started: 2024-03-16 13:28:10 +00:00)

<ipython-input-2-d48d86d36c46>:7: DtypeWarning: Columns (23,24,25)
have mixed types. Specify dtype option on import or set
low_memory=False.
  df = pd.read_csv(csv_file_path)
```

```
df.info()
list(df.columns)

# Assuming df is your original DataFrame
# Add your normalization or standardization code here

# Display summary statistics
df.describe()

total_records = len(df)
print(f"Total number of records: {total_records}")

# Calculate the total number of filled cells in each column
```

```python
filled_cells_count = df.count()

# Sum up the counts to get the total number of filled cells in the
DataFrame
total_filled_cells = filled_cells_count.sum()

print(f"Total number of filled cells: {total_filled_cells}")

# Assuming df is your DataFrame
unique_counts = df.nunique()

# Display the number of unique values in each column
print(unique_counts)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Data columns (total 37 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   tract_to_msamd_income           59878 non-null  float64
 1   rate_spread                     1866 non-null   float64
 2   population                      59878 non-null  float64
 3   minority_population             59878 non-null  float64
 4   number_of_owner_occupied_units  59876 non-null  float64
 5   number_of_1_to_4_family_units   59878 non-null  float64
 6   loan_amount_000s                60000 non-null  int64
 7   hud_median_family_income        59878 non-null  float64
 8   applicant_income_000s           53630 non-null  float64
 9   state_name                      60000 non-null  object
 10  state_abbr                      60000 non-null  object
 11  sequence_number                 60000 non-null  int64
 12  respondent_id                   60000 non-null  object
 13  purchaser_type_name             60000 non-null  object
 14  property_type_name              60000 non-null  object
 15  preapproval_name                60000 non-null  object
 16  owner_occupancy_name            60000 non-null  object
 17  msamd_name                      51619 non-null  object
 18  loan_type_name                  60000 non-null  object
 19  loan_purpose_name               60000 non-null  object
 20  lien_status_name                60000 non-null  object
 21  hoepa_status_name               60000 non-null  object
 22  edit_status_name                12451 non-null  object
 23  denial_reason_name_3            1 non-null      object
 24  denial_reason_name_2            43 non-null     object
 25  denial_reason_name_1            118 non-null    object
 26  county_name                     59908 non-null  object
 27  co_applicant_sex_name           60000 non-null  object
 28  co_applicant_ethnicity_name     60000 non-null  object
 29  census_tract_number             59878 non-null  float64
 30  as_of_year                      60000 non-null  int64
```

```
 31  application_date_indicator      60000 non-null  int64
 32  applicant_sex_name              60000 non-null  object
 33  applicant_ethnicity_name        60000 non-null  object
 34  agency_name                     60000 non-null  object
 35  agency_abbr                     60000 non-null  object
 36  action_taken_name               60000 non-null  object
dtypes: float64(9), int64(4), object(24)
memory usage: 16.9+ MB
Total number of records: 60000
Total number of filled cells: 1918780
tract_to_msamd_income              1327
rate_spread                         320
population                         1283
minority_population                1216
number_of_owner_occupied_units      996
number_of_1_to_4_family_units      1051
loan_amount_000s                   1344
hud_median_family_income             15
applicant_income_000s               807
state_name                            1
state_abbr                            1
sequence_number                   39340
respondent_id                       593
purchaser_type_name                  10
property_type_name                    3
preapproval_name                      3
owner_occupancy_name                  3
msamd_name                           14
loan_type_name                        4
loan_purpose_name                     3
lien_status_name                      4
hoepa_status_name                     2
edit_status_name                      1
denial_reason_name_3                  1
denial_reason_name_2                  8
denial_reason_name_1                  8
county_name                          39
co_applicant_sex_name                 5
co_applicant_ethnicity_name           5
census_tract_number                1108
as_of_year                            1
application_date_indicator            2
applicant_sex_name                    4
applicant_ethnicity_name              4
agency_name                           6
agency_abbr                           6
action_taken_name                     8
dtype: int64
time: 621 ms (started: 2024-03-16 13:28:11 +00:00)
```

```
# Assuming df is your DataFrame
columns_list = df.columns.tolist()
columns_list
```

```
list(df.columns)
```

```
['tract_to_msamd_income',
 'rate_spread',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'hud_median_family_income',
 'applicant_income_000s',
 'state_name',
 'state_abbr',
 'sequence_number',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'edit_status_name',
 'denial_reason_name_3',
 'denial_reason_name_2',
 'denial_reason_name_1',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'census_tract_number',
 'as_of_year',
 'application_date_indicator',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']
```

```
time: 5.04 ms (started: 2024-03-16 13:28:11 +00:00)
```

```
# Nominal and Ordinal Columns
```

```
# Continuous and Non Continuous Columns
```

```
import pandas as pd

# Assuming df is your DataFrame
continuous_columns = df.select_dtypes(include=['float64',
'int64']).columns
non_continuous_columns = df.select_dtypes(exclude=['float64',
'int64']).columns

print("Continuous Columns:", list(continuous_columns))
print("Non-Continuous Columns:", list(non_continuous_columns))


# Assuming df is your DataFrame
categorical_columns = df.select_dtypes(include=['object',
'category']).columns
non_categorical_columns = df.select_dtypes(exclude=['object',
'category']).columns

print("Categorical Columns:", list(categorical_columns))
print("Non-Categorical Columns:", list(non_categorical_columns))

'''
# Assuming df is your DataFrame

# Define a dictionary to store information about the nature of each
variable
variable_info = {}

# Loop through columns in the DataFrame
for column in df.columns:
    # Check if the variable has a limited number of unique values
    unique_values = df[column].nunique()

    if unique_values <= 10:
        # If the number of unique values is small, it could be an
ordinal variable
        variable_info[column] = 'Ordinal'
    else:
        # If the number of unique values is larger, it could be a
nominal variable
        variable_info[column] = 'Nominal'

# Print the information about each variable
for column, nature in variable_info.items():
    print(f"{column}: {nature}")
'''

Continuous Columns: ['tract_to_msamd_income', 'rate_spread',
'population', 'minority_population', 'number_of_owner_occupied_units',
'number_of_1_to_4_family_units', 'loan_amount_000s',
```

```
'hud_median_family_income', 'applicant_income_000s',
'sequence_number', 'census_tract_number', 'as_of_year',
'application_date_indicator']
Non-Continuous Columns: ['state_name', 'state_abbr', 'respondent_id',
'purchaser_type_name', 'property_type_name', 'preapproval_name',
'owner_occupancy_name', 'msamd_name', 'loan_type_name',
'loan_purpose_name', 'lien_status_name', 'hoepa_status_name',
'edit_status_name', 'denial_reason_name_3', 'denial_reason_name_2',
'denial_reason_name_1', 'county_name', 'co_applicant_sex_name',
'co_applicant_ethnicity_name', 'applicant_sex_name',
'applicant_ethnicity_name', 'agency_name', 'agency_abbr',
'action_taken_name']
Categorical Columns: ['state_name', 'state_abbr', 'respondent_id',
'purchaser_type_name', 'property_type_name', 'preapproval_name',
'owner_occupancy_name', 'msamd_name', 'loan_type_name',
'loan_purpose_name', 'lien_status_name', 'hoepa_status_name',
'edit_status_name', 'denial_reason_name_3', 'denial_reason_name_2',
'denial_reason_name_1', 'county_name', 'co_applicant_sex_name',
'co_applicant_ethnicity_name', 'applicant_sex_name',
'applicant_ethnicity_name', 'agency_name', 'agency_abbr',
'action_taken_name']
Non-Categorical Columns: ['tract_to_msamd_income', 'rate_spread',
'population', 'minority_population', 'number_of_owner_occupied_units',
'number_of_1_to_4_family_units', 'loan_amount_000s',
'hud_median_family_income', 'applicant_income_000s',
'sequence_number', 'census_tract_number', 'as_of_year',
'application_date_indicator']
```

{"type":"string"}

```
time: 35.7 ms (started: 2024-03-16 13:28:11 +00:00)
```

```
###  Missing Data Statistics and Treatment
###  Missing Data Statistics: Records

# Assuming df is your DataFrame

# Count the missing values in each column
missing_data = df.isnull().sum()

# Create a DataFrame to display missing data statistics
missing_data_stats = pd.DataFrame({
    'Column': missing_data.index,
    'Missing Records': missing_data.values,
    'Percentage Missing': (missing_data / len(df)) * 100
})

# Sort the DataFrame by the percentage of missing values in descending
order
missing_data_stats = missing_data_stats.sort_values(by='Percentage
```

```
Missing', ascending=False)

# Print the missing data statistics
print(missing_data_stats)
```

```
                                                Column  \
denial_reason_name_3                   denial_reason_name_3
denial_reason_name_2                   denial_reason_name_2
denial_reason_name_1                   denial_reason_name_1
rate_spread                                     rate_spread
edit_status_name                           edit_status_name
msamd_name                                       msamd_name
applicant_income_000s                 applicant_income_000s
number_of_owner_occupied_units  number_of_owner_occupied_units
census_tract_number                       census_tract_number
tract_to_msamd_income                 tract_to_msamd_income
hud_median_family_income           hud_median_family_income
number_of_1_to_4_family_units    number_of_1_to_4_family_units
minority_population                     minority_population
population                                       population
county_name                                     county_name
applicant_ethnicity_name           applicant_ethnicity_name
agency_name                                     agency_name
application_date_indicator        application_date_indicator
as_of_year                                       as_of_year
co_applicant_ethnicity_name      co_applicant_ethnicity_name
co_applicant_sex_name                 co_applicant_sex_name
agency_abbr                                     agency_abbr
applicant_sex_name                   applicant_sex_name
loan_type_name                           loan_type_name
hoepa_status_name                     hoepa_status_name
lien_status_name                       lien_status_name
loan_purpose_name                     loan_purpose_name
owner_occupancy_name                 owner_occupancy_name
preapproval_name                         preapproval_name
property_type_name                   property_type_name
purchaser_type_name                 purchaser_type_name
respondent_id                             respondent_id
sequence_number                         sequence_number
state_abbr                                   state_abbr
state_name                                   state_name
loan_amount_000s                       loan_amount_000s
action_taken_name                     action_taken_name

                                Missing Records  Percentage Missing
denial_reason_name_3                      59999           99.998333
denial_reason_name_2                      59957           99.928333
denial_reason_name_1                      59882           99.803333
rate_spread                               58134           96.890000
edit_status_name                          47549           79.248333
```

```
msamd_name                          8381            13.968333
applicant_income_000s               6370            10.616667
number_of_owner_occupied_units       124             0.206667
census_tract_number                  122             0.203333
tract_to_msamd_income                122             0.203333
hud_median_family_income             122             0.203333
number_of_1_to_4_family_units        122             0.203333
minority_population                  122             0.203333
population                           122             0.203333
county_name                           92             0.153333
applicant_ethnicity_name               0             0.000000
agency_name                            0             0.000000
application_date_indicator             0             0.000000
as_of_year                             0             0.000000
co_applicant_ethnicity_name            0             0.000000
co_applicant_sex_name                  0             0.000000
agency_abbr                            0             0.000000
applicant_sex_name                     0             0.000000
loan_type_name                         0             0.000000
hoepa_status_name                      0             0.000000
lien_status_name                       0             0.000000
loan_purpose_name                      0             0.000000
owner_occupancy_name                   0             0.000000
preapproval_name                       0             0.000000
property_type_name                     0             0.000000
purchaser_type_name                    0             0.000000
respondent_id                          0             0.000000
sequence_number                        0             0.000000
state_abbr                             0             0.000000
state_name                             0             0.000000
loan_amount_000s                       0             0.000000
action_taken_name                      0             0.000000
time: 211 ms (started: 2024-03-16 13:28:11 +00:00)
```

```
# List of columns to drop
columns_to_drop = ['denial_reason_name_3', 'denial_reason_name_2',
'denial_reason_name_1', 'rate_spread', 'edit_status_name']

# Drop columns with more than 50% missing values
df_cleaned = df.drop(columns=columns_to_drop)

# Print the cleaned DataFrame
df1 = df_cleaned

# Count the missing values in each column
missing_data = df1.isnull().sum()

# Create a DataFrame to display missing data statistics
missing_data_stats = pd.DataFrame({
    'Column': missing_data.index,
```

```
    'Missing Records': missing_data.values,
    'Percentage Missing': (missing_data / len(df)) * 100
})
# Sort the DataFrame by the percentage of missing values in descending
order
missing_data_stats = missing_data_stats.sort_values(by='Percentage
Missing', ascending=False)

# Print the missing data statistics
print(missing_data_stats)
```

```
                                                    Column  \
msamd_name                                      msamd_name
applicant_income_000s                  applicant_income_000s
number_of_owner_occupied_units  number_of_owner_occupied_units
hud_median_family_income              hud_median_family_income
census_tract_number                        census_tract_number
population                                          population
tract_to_msamd_income                  tract_to_msamd_income
minority_population                      minority_population
number_of_1_to_4_family_units    number_of_1_to_4_family_units
county_name                                    county_name
loan_amount_000s                          loan_amount_000s
hoepa_status_name                        hoepa_status_name
agency_abbr                                    agency_abbr
agency_name                                    agency_name
applicant_ethnicity_name            applicant_ethnicity_name
applicant_sex_name                        applicant_sex_name
application_date_indicator          application_date_indicator
as_of_year                                      as_of_year
co_applicant_ethnicity_name        co_applicant_ethnicity_name
co_applicant_sex_name                    co_applicant_sex_name
lien_status_name                          lien_status_name
state_name                                      state_name
loan_purpose_name                        loan_purpose_name
loan_type_name                              loan_type_name
owner_occupancy_name                  owner_occupancy_name
preapproval_name                          preapproval_name
property_type_name                      property_type_name
purchaser_type_name                    purchaser_type_name
respondent_id                                respondent_id
sequence_number                            sequence_number
state_abbr                                      state_abbr
action_taken_name                        action_taken_name

                                Missing Records  Percentage Missing
msamd_name                                 8381           13.968333
applicant_income_000s                      6370           10.616667
number_of_owner_occupied_units              124            0.206667
hud_median_family_income                    122            0.203333
```

```
census_tract_number                    122            0.203333
population                             122            0.203333
tract_to_msamd_income                  122            0.203333
minority_population                    122            0.203333
number_of_1_to_4_family_units          122            0.203333
county_name                             92            0.153333
loan_amount_000s                         0            0.000000
hoepa_status_name                        0            0.000000
agency_abbr                              0            0.000000
agency_name                              0            0.000000
applicant_ethnicity_name                 0            0.000000
applicant_sex_name                       0            0.000000
application_date_indicator               0            0.000000
as_of_year                               0            0.000000
co_applicant_ethnicity_name              0            0.000000
co_applicant_sex_name                    0            0.000000
lien_status_name                         0            0.000000
state_name                               0            0.000000
loan_purpose_name                        0            0.000000
loan_type_name                           0            0.000000
owner_occupancy_name                     0            0.000000
preapproval_name                         0            0.000000
property_type_name                       0            0.000000
purchaser_type_name                      0            0.000000
respondent_id                            0            0.000000
sequence_number                          0            0.000000
state_abbr                               0            0.000000
action_taken_name                        0            0.000000
time: 294 ms (started: 2024-03-16 13:28:11 +00:00)
```

```
### Missing Records (ROWS)

# Count the missing values in each row
missing_rows = df1.isnull().sum(axis=1)

# Count the number of rows with at least one missing value
num_rows_with_missing = len(missing_rows[missing_rows > 0])

# Print the number of rows with missing data
print("Number of rows with missing data:", num_rows_with_missing)


# Calculate the percentage of missing values in each row
missing_percentage_rows = (df1.isnull().sum(axis=1) /
len(df1.columns)) * 100

# Count the number of rows with more than 50% missing data
num_rows_more_than_50_percent_missing =
len(missing_percentage_rows[missing_percentage_rows > 50])
```

```
# Print the number of rows with more than 50% missing data
print("Number of rows with more than 50% missing data:",
num_rows_more_than_50_percent_missing)
```

```
Number of rows with missing data: 13629
Number of rows with more than 50% missing data: 0
time: 522 ms (started: 2024-03-16 13:28:12 +00:00)
```

```
# DIVIDING DF1 into Cat and Non Cat

# Assuming df1 is your DataFrame
cat_columns = df1.select_dtypes(include=['object']).columns
noncat_columns = df1.select_dtypes(exclude=['object']).columns

# Creating categorical and non-categorical DataFrames
catdf = df1[cat_columns]
noncatdf = df1[noncat_columns]

#print(list(catdf.columns))
#print(list(noncatdf.columns))
list(catdf.columns)
#20
#list(noncatdf.columns)
```

```
['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']
```

```
time: 35.2 ms (started: 2024-03-16 13:28:12 +00:00)
```

```
####  STATISTICS OF CAT DATASET

# Count and frequency statistics for each column in catdf
catdf_stats = pd.DataFrame()
```

```
for column in catdf.columns:
    col_count = catdf[column].value_counts().reset_index()
    col_count.columns = [column, 'Frequency']
    catdf_stats = pd.concat([catdf_stats, col_count], axis=1)

# Display the count and frequency statistics
#print(catdf_stats)

# Summary for each column in catdf
catdf_summary = catdf.describe(include='all').transpose()

# Display the summary
print(catdf_summary)


# Calculate the proportion (relative frequency) for each categorical
column
proportion_stats = catdf.apply(lambda x:
x.value_counts(normalize=True).idxmax() + ': ' +
"{:.2%}".format(x.value_counts(normalize=True).max()))

# Display the proportion statistics
print(proportion_stats)
```

```
                            count unique  \
state_name                  60000      1
state_abbr                  60000      1
respondent_id               60000    593
purchaser_type_name         60000     10
property_type_name          60000      3
preapproval_name            60000      3
owner_occupancy_name        60000      3
msamd_name                  51619     14
loan_type_name              60000      4
loan_purpose_name           60000      3
lien_status_name            60000      4
hoepa_status_name           60000      2
county_name                 59908     39
co_applicant_sex_name       60000      5
co_applicant_ethnicity_name 60000      5
applicant_sex_name          60000      4
applicant_ethnicity_name    60000      4
agency_name                 60000      6
agency_abbr                 60000      6
action_taken_name           60000      8


top  \
state_name
```

```
Washington
state_abbr
WA
respondent_id
32489
purchaser_type_name          Loan was not originated or was not sold
in cal...
property_type_name           One-to-four family dwelling (other than
manufa...
preapproval_name                                                 Not
applicable
owner_occupancy_name                    Owner-occupied as a principal
dwelling
msamd_name                                    Seattle, Bellevue,
Everett - WA
loan_type_name
Conventional
loan_purpose_name
Refinancing
lien_status_name                                          Secured by a
first lien
hoepa_status_name                                            Not a
HOEPA loan
county_name
King County
co_applicant_sex_name                                         No co-
applicant
co_applicant_ethnicity_name                                   No co-
applicant
applicant_sex_name
Male
applicant_ethnicity_name                                Not Hispanic
or Latino
agency_name                          Department of Housing and Urban
Development
agency_abbr
HUD
action_taken_name                                             Loan
originated

                              freq
state_name                   60000
state_abbr                   60000
respondent_id                 5006
purchaser_type_name          16112
property_type_name           57630
preapproval_name             47832
owner_occupancy_name         53940
msamd_name                   17965
```

```
loan_type_name                42917
loan_purpose_name             28576
lien_status_name              57046
hoepa_status_name             59996
county_name                   12915
co_applicant_sex_name         26987
co_applicant_ethnicity_name   26987
applicant_sex_name            37070
applicant_ethnicity_name      44014
agency_name                   27514
agency_abbr                   27514
action_taken_name             55815
state_name
Washington: 100.00%
state_abbr
WA: 100.00%
respondent_id
32489: 8.34%
purchaser_type_name             Loan was not originated or was not sold
in cal...
property_type_name              One-to-four family dwelling (other than
manufa...
preapproval_name                                        Not
applicable: 79.72%
owner_occupancy_name               Owner-occupied as a principal
dwelling: 89.90%
msamd_name                                Seattle, Bellevue, Everett -
WA: 34.80%
loan_type_name
Conventional: 71.53%
loan_purpose_name
Refinancing: 47.63%
lien_status_name                                    Secured by a first
lien: 95.08%
hoepa_status_name                                       Not a HOEPA
loan: 99.99%
county_name                                                 King
County: 21.56%
co_applicant_sex_name                                    No co-
applicant: 44.98%
co_applicant_ethnicity_name                              No co-
applicant: 44.98%
applicant_sex_name
Male: 61.78%
applicant_ethnicity_name                          Not Hispanic or
Latino: 73.36%
agency_name                    Department of Housing and Urban
Development: 4...
agency_abbr
```

```
HUD: 45.86%
action_taken_name                                       Loan
originated: 93.03%
dtype: object
time: 552 ms (started: 2024-03-16 13:28:12 +00:00)
```

```
#### STATISTICS OF NONCAT DATASET

# Display descriptive statistics for non-categorical variables
noncatdf_descriptive_stats = noncatdf.describe()

# Print the descriptive statistics
print(noncatdf_descriptive_stats)
```

```
       tract_to_msamd_income    population  minority_population  \
count           59878.000000  59878.000000         59878.000000
mean              107.617351   5278.782157            23.244442
std                28.233471   1716.101490            14.416209
min                14.050000     98.000000             2.040000
25%                88.970001   4070.000000            12.950000
50%               105.550003   5145.000000            19.420000
75%               123.330002   6382.000000            29.680000
max               257.140015  13025.000000            94.790001

       number_of_owner_occupied_units
number_of_1_to_4_family_units  \
count                     59876.000000                     59878.000000

mean                       1399.044375                      1873.281456

std                         518.330561                       738.505184

min                           15.000000                       27.000000

25%                        1034.000000                      1414.000000

50%                        1359.000000                      1770.000000

75%                        1722.000000                      2249.000000

max                        2997.000000                      5893.000000


       loan_amount_000s  hud_median_family_income
applicant_income_000s  \
count      60000.000000              59878.000000
53630.000000
mean         291.358717              73869.411136
112.822301
std          604.958183              12811.243390
122.862496
```

```
min             1.000000              48700.000000
1.000000
25%           170.000000              63100.000000
61.000000
50%           242.000000              73300.000000
89.000000
75%           337.000000              90300.000000
132.000000
max         55000.000000              90300.000000
6161.000000

       sequence_number  census_tract_number  as_of_year  \
count     6.000000e+04         59878.000000     60000.0
mean      7.752647e+04          1750.597252      2016.0
std       1.505157e+05          3359.676740         0.0
min       1.000000e+00             1.000000      2016.0
25%       3.231500e+03           114.020000      2016.0
50%       1.648100e+04           403.020000      2016.0
75%       7.276225e+04           713.100000      2016.0
max       1.241590e+06          9757.000000      2016.0

       application_date_indicator
count                60000.000000
mean                     0.025867
std                      0.225976
min                      0.000000
25%                      0.000000
50%                      0.000000
75%                      0.000000
max                      2.000000
time: 70.7 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
'''
correlation_matrix = noncatdf.corr()

# Initialize empty matrices for p-values
p_values = np.zeros_like(correlation_matrix)
n = len(noncatdf)

# Calculate correlation coefficients and p-values
for i in range(len(noncatdf.columns)):
    for j in range(i, len(noncatdf.columns)):
        corr, p_value = pearsonr(noncatdf.iloc[:, i], noncatdf.iloc[:,
j])
        correlation_matrix.iloc[i, j] = corr
        correlation_matrix.iloc[j, i] = corr
        p_values[i, j] = p_value
        p_values[j, i] = p_value

# Create DataFrames for correlation coefficients and p-values
```

```
correlation_df = pd.DataFrame(correlation_matrix,
columns=noncatdf.columns, index=noncatdf.columns)
p_values_df = pd.DataFrame(p_values, columns=noncatdf.columns,
index=noncatdf.columns)
'''
```

```
{"type":"string"}
```

```
time: 4.18 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
# Missing Data Statistics: Non-Categorical Variables or Features

# Calculate missing data statistics for non-categorical columns
missing_data_non_categorical = noncatdf.isnull().sum().reset_index()
missing_data_non_categorical.columns = ['Feature', 'Missing_Records']

# Display the missing data statistics
print(missing_data_non_categorical)
```

```
                         Feature  Missing_Records
0            tract_to_msamd_income              122
1                       population              122
2              minority_population              122
3   number_of_owner_occupied_units              124
4    number_of_1_to_4_family_units              122
5                 loan_amount_000s                0
6         hud_median_family_income              122
7            applicant_income_000s             6370
8                 sequence_number                0
9              census_tract_number              122
10                      as_of_year                0
11      application_date_indicator                0
time: 13.9 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
#  Missing Data Treatment: Non-Categorical Variables or Features

# Dataset Used : df_noncat

si_noncat = SimpleImputer(missing_values=np.nan, strategy='mean') #
Other Strategy : mean | median | most_frequent | constant
si_noncat_fit = si_noncat.fit_transform(noncatdf)
imputed_data_non_categorical = pd.DataFrame(si_noncat_fit,
columns=noncatdf.columns); # Missing Non-Categorical Data Imputed
Subset using Simple Imputer
imputed_data_non_categorical.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
```

```
---  ------                              --------------  -----
 0   tract_to_msamd_income               60000 non-null  float64
 1   population                          60000 non-null  float64
 2   minority_population                 60000 non-null  float64
 3   number_of_owner_occupied_units      60000 non-null  float64
 4   number_of_1_to_4_family_units       60000 non-null  float64
 5   loan_amount_000s                    60000 non-null  float64
 6   hud_median_family_income            60000 non-null  float64
 7   applicant_income_000s               60000 non-null  float64
 8   sequence_number                     60000 non-null  float64
 9   census_tract_number                 60000 non-null  float64
 10  as_of_year                          60000 non-null  float64
 11  application_date_indicator          60000 non-null  float64
dtypes: float64(12)
memory usage: 5.5 MB
time: 63.4 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
# Calculate standard deviation for non-categorical columns
std_deviation_non_categorical = imputed_data_non_categorical.std()

# Creating a DataFrame to display the results
dispersion_non_categorical_df = pd.DataFrame({
    'Variable': imputed_data_non_categorical.columns,
    'Standard Deviation': std_deviation_non_categorical.values
})

print(dispersion_non_categorical_df)
```

```
                         Variable  Standard Deviation
0            tract_to_msamd_income           28.204752
1                       population         1714.355870
2              minority_population           14.401545
3   number_of_owner_occupied_units          517.794667
4    number_of_1_to_4_family_units          737.753976
5                  loan_amount_000s          604.958183
6         hud_median_family_income        12798.211781
7            applicant_income_000s          116.157479
8                 sequence_number       150515.678152
9              census_tract_number         3356.259273
10                      as_of_year            0.000000
11      application_date_indicator            0.225976
time: 23.9 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
# Compute the correlation matrix for non-categorical variables
correlation_matrix_non_categorical =
imputed_data_non_categorical.corr()

# Print correlation matrix
print("Correlation Matrix (Non-Categorical Variables):")
print(correlation_matrix_non_categorical)
```

```python
# Compute correlation coefficients for non-categorical variables
correlation_coefficients_non_categorical =
correlation_matrix_non_categorical.unstack().sort_values(ascending=Fal
se)

# Print correlation coefficients
print("\nCorrelation Coefficients (Non-Categorical Variables):")
print(correlation_coefficients_non_categorical)

# Test of correlation (using Pearson correlation coefficient) for
specific variables
corr_tract_income, p_value_tract_income =
stats.pearsonr(imputed_data_non_categorical['tract_to_msamd_income'],
imputed_data_non_categorical['hud_median_family_income'])
corr_population_minority, p_value_population_minority =
stats.pearsonr(imputed_data_non_categorical['population'],
imputed_data_non_categorical['minority_population'])
corr_owner_units_loan, p_value_owner_units_loan =
stats.pearsonr(imputed_data_non_categorical['number_of_owner_occupied_
units'], imputed_data_non_categorical['loan_amount_000s'])

# Print correlation coefficients and p-values for specific variables
print("\nCorrelation Coefficient (tract_to_msamd_income vs
hud_median_family_income):", corr_tract_income)
print("P-value:", p_value_tract_income)

print("\nCorrelation Coefficient (population vs
minority_population):", corr_population_minority)
print("P-value:", p_value_population_minority)

print("\nCorrelation Coefficient (number_of_owner_occupied_units vs
loan_amount_000s):", corr_owner_units_loan)
print("P-value:", p_value_owner_units_loan)
```

```
Correlation Matrix (Non-Categorical Variables):
                                tract_to_msamd_income  population  \
tract_to_msamd_income                        1.000000    0.048580
population                                   0.048580    1.000000
minority_population                         -0.403257    0.184727
number_of_owner_occupied_units               0.370891    0.777449
number_of_1_to_4_family_units                0.132696    0.709542
loan_amount_000s                             0.081783    0.002061
hud_median_family_income                    -0.009365   -0.012832
applicant_income_000s                        0.184541    0.002256
sequence_number                              0.012611    0.004819
census_tract_number                         -0.022382   -0.105298
as_of_year                                        NaN         NaN
application_date_indicator                   0.011541   -0.015836
```

```
                                  minority_population  \
tract_to_msamd_income                       -0.403257
population                                   0.184727
minority_population                          1.000000
number_of_owner_occupied_units              -0.202225
number_of_1_to_4_family_units               -0.175571
loan_amount_000s                             0.012030
hud_median_family_income                     0.242211
applicant_income_000s                       -0.031117
sequence_number                             -0.022516
census_tract_number                         -0.143931
as_of_year                                        NaN
application_date_indicator                  -0.042659

                                  number_of_owner_occupied_units  \
tract_to_msamd_income                                   0.370891
population                                              0.777449
minority_population                                    -0.202225
number_of_owner_occupied_units                          1.000000
number_of_1_to_4_family_units                           0.849083
loan_amount_000s                                        0.002045
hud_median_family_income                               -0.079037
applicant_income_000s                                   0.037969
sequence_number                                         0.015118
census_tract_number                                     0.019509
as_of_year                                                   NaN
application_date_indicator                              0.016227

                                  number_of_1_to_4_family_units  \
tract_to_msamd_income                                  0.132696
population                                             0.709542
minority_population                                   -0.175571
number_of_owner_occupied_units                         0.849083
number_of_1_to_4_family_units                          1.000000
loan_amount_000s                                      -0.032803
hud_median_family_income                              -0.298878
applicant_income_000s                                  0.002561
sequence_number                                        0.024564
census_tract_number                                    0.205901
as_of_year                                                  NaN
application_date_indicator                             0.070299

                                  loan_amount_000s
hud_median_family_income  \
tract_to_msamd_income                     0.081783                 -
0.009365
population                                0.002061                 -
0.012832
minority_population                       0.012030
0.242211
```

```
number_of_owner_occupied_units          0.002045                  -
0.079037
number_of_1_to_4_family_units          -0.032803                  -
0.298878
loan_amount_000s                        1.000000
0.128613
hud_median_family_income                0.128613
1.000000
applicant_income_000s                   0.146546
0.153824
sequence_number                         0.011604                  -
0.060147
census_tract_number                    -0.040378                  -
0.493106
as_of_year                                   NaN
NaN
application_date_indicator             -0.000406                  -
0.153317

                                  applicant_income_000s  sequence_number
\
tract_to_msamd_income                          0.184541          0.012611

population                                     0.002256          0.004819

minority_population                           -0.031117         -0.022516

number_of_owner_occupied_units                 0.037969          0.015118

number_of_1_to_4_family_units                  0.002561          0.024564

loan_amount_000s                               0.146546          0.011604

hud_median_family_income                       0.153824         -0.060147

applicant_income_000s                          1.000000         -0.002903

sequence_number                               -0.002903          1.000000

census_tract_number                           -0.033578          0.028415

as_of_year                                          NaN               NaN

application_date_indicator                    -0.010670          0.124127


                                  census_tract_number  as_of_year  \
tract_to_msamd_income                       -0.022382         NaN
population                                  -0.105298         NaN
minority_population                         -0.143931         NaN
number_of_owner_occupied_units               0.019509         NaN
```

```
number_of_1_to_4_family_units              0.205901         NaN
loan_amount_000s                          -0.040378         NaN
hud_median_family_income                  -0.493106         NaN
applicant_income_000s                     -0.033578         NaN
sequence_number                            0.028415         NaN
census_tract_number                        1.000000         NaN
as_of_year                                      NaN         NaN
application_date_indicator                 0.149146         NaN

                                  application_date_indicator
tract_to_msamd_income                               0.011541
population                                         -0.015836
minority_population                                -0.042659
number_of_owner_occupied_units                      0.016227
number_of_1_to_4_family_units                       0.070299
loan_amount_000s                                   -0.000406
hud_median_family_income                           -0.153317
applicant_income_000s                              -0.010670
sequence_number                                     0.124127
census_tract_number                                 0.149146
as_of_year                                               NaN
application_date_indicator                          1.000000

Correlation Coefficients (Non-Categorical Variables):
tract_to_msamd_income       tract_to_msamd_income         1.0
population                  population                    1.0
census_tract_number         census_tract_number           1.0
sequence_number             sequence_number               1.0
applicant_income_000s       applicant_income_000s         1.0
                                                         ...
as_of_year                  sequence_number               NaN
                            census_tract_number           NaN
                            as_of_year                    NaN
                            application_date_indicator    NaN
application_date_indicator  as_of_year                    NaN
Length: 144, dtype: float64

Correlation Coefficient (tract_to_msamd_income vs
hud_median_family_income): -0.00936470130537446
P-value: 0.02179759460285561

Correlation Coefficient (population vs minority_population):
0.18472678320300429
P-value: 0.0

Correlation Coefficient (number_of_owner_occupied_units vs
loan_amount_000s): 0.0020453682357100228
P-value: 0.6163704631114627
time: 92.9 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
'''

####   Correlation Statistics (with Test of Correlation)

# Initialize correlation matrix and p-values matrix
correlation_matrix =
pd.DataFrame(index=imputed_data_non_categorical.columns,
columns=imputed_data_non_categorical.columns)
p_values = np.zeros((len(imputed_data_non_categorical.columns),
len(imputed_data_non_categorical.columns)))

# Calculate correlations and p-values
for i in range(len(imputed_data_non_categorical.columns)):
    for j in range(i, len(imputed_data_non_categorical.columns)):
        if imputed_data_non_categorical.iloc[:, i].nunique() > 1 and
imputed_data_non_categorical.iloc[:, j].nunique() > 1:
            corr, p_value =
pearsonr(imputed_data_non_categorical.iloc[:, i],
imputed_data_non_categorical.iloc[:, j])
            correlation_matrix.iloc[i, j] = corr
            correlation_matrix.iloc[j, i] = corr
            p_values[i, j] = p_value
            p_values[j, i] = p_value

# Create DataFrames for correlation coefficients and p-values
correlation_df = pd.DataFrame(correlation_matrix,
columns=imputed_data_non_categorical.columns,
index=imputed_data_non_categorical.columns)
p_values_df = pd.DataFrame(p_values,
columns=imputed_data_non_categorical.columns,
index=imputed_data_non_categorical.columns)
correlation_df
p_values_df

'''
```

{"type":"string"}

time: 13.2 ms (started: 2024-03-16 13:28:13 +00:00)

```
'''
#  Missing Data Treatment: Non-Categorical Variables or Features

# Impute missing values with the mean
df_non_categorical_imputed = noncatdf.fillna(noncatdf.mean())

# Calculate missing data statistics for non-categorical columns
imputed_data_non_categorical =
df_non_categorical_imputed.isnull().sum().reset_index()
imputed_data_non_categorical.columns = ['Feature', 'Missing_Records']
```

```
# Display the missing data statistics
print(imputed_data_non_categorical)
'''
```

```
{"type":"string"}
```

```
time: 4.29 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
'''
# Missing Data Treatment: Categorical Variables or Features

# Impute missing values with the mode
df_categorical_imputed = catdf.apply(lambda x: x.fillna(x.mode()[0]))

# Calculate missing data statistics for categorical columns
imputed_data_categorical =
df_categorical_imputed.isnull().sum().reset_index()
imputed_data_categorical.columns = ['Feature', 'Missing_Records']

# Display the missing data statistics
print(imputed_data_categorical)
'''
```

```
{"type":"string"}
```

```
time: 8.07 ms (started: 2024-03-16 13:28:13 +00:00)
```

```
# Dataset Used : df_cat

si_cat = SimpleImputer(missing_values=np.nan,
strategy='most_frequent') # Strategy = median [When Odd Number of
Categories Exists]
si_cat_fit = si_cat.fit_transform(catdf)
imputed_data_categorical = pd.DataFrame(si_cat_fit,
columns=catdf.columns); # Missing Categorical Data Imputed Subset
imputed_data_categorical.info()
imputed_data_categorical.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Data columns (total 20 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   state_name                   60000 non-null  object
 1   state_abbr                   60000 non-null  object
 2   respondent_id                60000 non-null  object
 3   purchaser_type_name          60000 non-null  object
 4   property_type_name           60000 non-null  object
 5   preapproval_name             60000 non-null  object
 6   owner_occupancy_name         60000 non-null  object
 7   msamd_name                   60000 non-null  object
```

```
 8   loan_type_name                60000 non-null  object
 9   loan_purpose_name             60000 non-null  object
 10  lien_status_name              60000 non-null  object
 11  hoepa_status_name             60000 non-null  object
 12  county_name                   60000 non-null  object
 13  co_applicant_sex_name         60000 non-null  object
 14  co_applicant_ethnicity_name   60000 non-null  object
 15  applicant_sex_name            60000 non-null  object
 16  applicant_ethnicity_name      60000 non-null  object
 17  agency_name                   60000 non-null  object
 18  agency_abbr                   60000 non-null  object
 19  action_taken_name             60000 non-null  object
dtypes: object(20)
memory usage: 9.2+ MB
```


```
{"summary":"{\n  \"name\": \"imputed_data_categorical\",\n  \"rows\":
60000,\n  \"fields\": [\n    {\n      \"column\": \"state_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n        \"samples\": [\n
\"Washington\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"state_abbr\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"WA\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"respondent_id\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 593,\n
\"samples\": [\n          \"86-0860478\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"purchaser_type_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n
\"Private securitization\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"property_type_name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"One-to-four family dwelling (other than
manufactured housing)\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"preapproval_name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Not applicable\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"owner_occupancy_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"Owner-
occupied as a principal dwelling\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"msamd_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
```

```
\"num_unique_values\": 14,\n        \"samples\": [\n
\"Spokane, Spokane Valley - WA\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"loan_type_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"FHA-
insured\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"loan_purpose_name\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Refinancing\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"lien_status_name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"Secured by a subordinate lien\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"hoepa_status_name\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"HOEPA loan\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"county_name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 39,\n
\"samples\": [\n          \"Whitman County\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"co_applicant_sex_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"No co-
applicant\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"co_applicant_ethnicity_name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"No co-applicant\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"applicant_sex_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"Male\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"applicant_ethnicity_name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"Hispanic or Latino\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"agency_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n
\"Consumer Financial Protection Bureau\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"agency_abbr\",\n
```

\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n   \"samples\": [\n   \"CFPB\"\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"action_taken_name\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 8,\n   \"samples\":
[\n   \"Application approved but not accepted\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\
n}","type":"dataframe","variable_name":"imputed_data_categorical"}

```
time: 1.5 s (started: 2024-03-16 13:28:13 +00:00)
```

```
# ENCODING
# Converting Categorical Variable into Numeric

# Calculate the number of unique values in each column
unique_values_categorical =
imputed_data_categorical.nunique().reset_index()
unique_values_categorical.columns = ['Feature',
'Number_of_Unique_Values']

# Display the number of unique values
print(unique_values_categorical)


# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Create a copy of the imputed_data_categorical dataframe to avoid
modifying the original
encoded_data_categorical = imputed_data_categorical.copy()

# Iterate through each column in the dataframe
mapping = {}  # To store the mapping of variable names to numeric
representation

for column in encoded_data_categorical.columns:
    # Perform numerical encoding
    encoded_data_categorical[column] =
label_encoder.fit_transform(encoded_data_categorical[column])

    # Store the mapping information
    mapping[column] = dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)))

# Display the mapping
for variable, variable_mapping in mapping.items():
    print(f"\nMapping for {variable}:")
    print(variable_mapping)
```

```
# Display the encoded data
print(encoded_data_categorical)
```

```
                        Feature  Number_of_Unique_Values
0                    state_name                        1
1                    state_abbr                        1
2                 respondent_id                      593
3           purchaser_type_name                       10
4            property_type_name                        3
5              preapproval_name                        3
6          owner_occupancy_name                        3
7                    msamd_name                       14
8                 loan_type_name                        4
9              loan_purpose_name                        3
10              lien_status_name                        4
11             hoepa_status_name                        2
12                   county_name                       39
13        co_applicant_sex_name                        5
14  co_applicant_ethnicity_name                        5
15           applicant_sex_name                        4
16     applicant_ethnicity_name                        4
17                   agency_name                       6
18                   agency_abbr                       6
19             action_taken_name                       8

Mapping for state_name:
{'Washington': 0}

Mapping for state_abbr:
{'WA': 0}

Mapping for respondent_id:
{'01-0681100': 0, '01-0726495': 1, '02-0793125': 2, '03-0488052': 3,
'04-3212636': 4, '04-3568208': 5, '04-3660901': 6, '04-7534967': 7,
'05-0402708': 8, '06-1016329': 9, '1015560': 10, '10257': 11, '1047':
12, '1097500000': 13, '1099800006': 14, '11-3399725': 15, '11-
3412303': 16, '11-3714032': 17, '11162': 18, '112837': 19, '11443':
20, '1146500007': 21, '11729': 22, '11734': 23, '12135': 24,
'1216826': 25, '12219': 26, '1227300009': 27, '12311': 28, '1265': 29,
'1281': 30, '13-3222578': 31, '13-3602661': 32, '13-3753941': 33, '13-
4225190': 34, '13-4362989': 35, '13-6131491': 36, '13232': 37,
'13303': 38, '13392': 39, '13778': 40, '14-1841762': 41, '14206': 42,
'14252': 43, '143662': 44, '1461700004': 45, '14662': 46, '146672':
47, '14740': 48, '14843': 49, '151': 50, '15219': 51, '15732': 52,
'16-1686740': 53, '16243': 54, '1635900004': 55, '16450': 56, '16814':
57, '169653': 58, '17587': 59, '17672': 60, '177957': 61, '17874': 62,
'17884': 63, '1842065': 64, '19307': 65, '19628': 66, '197478': 67,
'19899': 68, '19976': 69, '20-0142846': 70, '20-0192872': 71, '20-
0304793': 72, '20-0640473': 73, '20-0740151': 74, '20-1255434': 75,
```

'20-1832276': 76, '20-2053401': 77, '20-2355296': 78, '20-2470783': 79, '20-2471369': 80, '20-2485875': 81, '20-2693054': 82, '20-2718340': 83, '20-2752826': 84, '20-2928975': 85, '20-3702275': 86, '20-3828708': 87, '20-4136310': 88, '20-4224234': 89, '20-4255880': 90, '20-4866754': 91, '20-5238443': 92, '20-5239910': 93, '20-5741925': 94, '20-8006279': 95, '20-8083209': 96, '20-8544905': 97, '20-8745846': 98, '20-8803449': 99, '20-8921389': 100, '2003500009': 101, '20061': 102, '20068': 103, '20214': 104, '20516': 105, '20624': 106, '20774': 107, '210434': 108, '21122': 109, '212465': 110, '2137100009': 111, '2149009991': 112, '21717': 113, '2191': 114, '2193616': 115, '22-3039688': 116, '22-3470404': 117, '22-3554558': 118, '22-3626426': 119, '22-3747694': 120, '22-3887207': 121, '22134': 122, '22157': 123, '22407': 124, '22444': 125, '22637': 126, '2285': 127, '22939': 128, '23-2470039': 129, '23-2769131': 130, '23041': 131, '2317700005': 132, '23216': 133, '23416': 134, '23521': 135, '23850': 136, '23922': 137, '23957': 138, '24077': 139, '24080': 140, '24107': 141, '24169': 142, '24224': 143, '24235': 144, '24326': 145, '24382': 146, '24671': 147, '24708': 148, '24713': 149, '24719': 150, '24753': 151, '24760': 152, '24831': 153, '24849': 154, '2489805': 155, '25080': 156, '25093': 157, '25103': 158, '2562164': 159, '2590037': 160, '26-0012825': 161, '26-0021318': 162, '26-0335190': 163, '26-0360466': 164, '26-0362771': 165, '26-0423240': 166, '26-0455770': 167, '26-0508430': 168, '26-0595342': 169, '26-0707492': 170, '26-1242154': 171, '26-1334020': 172, '26-1589507': 173, '26-1773722': 174, '26-2049351': 175, '26-2261031': 176, '26-2593704': 177, '26-2689428': 178, '26-2916887': 179, '26-3264687': 180, '26-3416474': 181, '26-3780954': 182, '26-4193875': 183, '26-4461592': 184, '26-4558390': 185, '26-4599244': 186, '2618780': 187, '26610': 188, '267100004': 189, '27-0222046': 190, '27-0267182': 191, '27-0684906': 192, '27-0812934': 193, '27-1190043': 194, '27-1438405': 195, '27-2389039': 196, '27-3459350': 197, '27-4023565': 198, '27-4626537': 199, '27280': 200, '2734': 201, '2735146': 202, '27601': 203, '276579': 204, '280110': 205, '28116': 206, '28151': 207, '28316': 208, '28405': 209, '28453': 210, '28454': 211, '28489': 212, '28599': 213, '2888798': 214, '29012': 215, '29058': 216, '29209': 217, '2945': 218, '3027509990': 219, '3076248': 220, '30788': 221, '30810': 222, '31-1197926': 223, '31-1690008': 224, '31-1712553': 225, '311845': 226, '3121': 227, '31286': 228, '3150447': 229, '319': 230, '32-0016270': 231, '3212149': 232, '32178': 233, '32489': 234, '3284070': 235, '33-0231744': 236, '33-0397503': 237, '33-0419992': 238, '33-0594693': 239, '33-0750812': 240, '33-0816610': 241, '33-0828099': 242, '33-0941669': 243, '33-0962918': 244, '33-0975529': 245, '33183': 246, '33508': 247, '3374412': 248, '33806': 249, '33826': 250, '339858': 251, '34-1194858': 252, '34-1633105': 253, '34-1716542': 254, '34-1719615': 255, '34-2000096': 256, '34106': 257, '34214': 258, '34585': 259, '34607': 260, '34627': 261, '34953': 262, '35-2486440': 263, '35013': 264, '35014': 265, '35139': 266, '35261': 267, '35355': 268, '3537897': 269, '35406': 270, '36-4327855': 271, '365325': 272, '37-1493496': 273, '37-1542226': 274, '38-2434249': 275, '38-2749215':

276, '38-2750395': 277, '38-2799035': 278, '38-3564305': 279, '39-2001010': 280, '3918898': 281, '3938186': 282, '3956': 283, '4004574': 284, '41-1795868': 285, '41-1914032': 286, '41-2277737': 287, '4114567': 288, '413208': 289, '4142': 290, '42-1554181': 291, '42-1739728': 292, '4239': 293, '43-0951349': 294, '43-1965151': 295, '435': 296, '45-3143795': 297, '45-3508295': 298, '451965': 299, '46-0530020': 300, '46-1728831': 301, '46-1836968': 302, '46-2477192': 303, '46-2888046': 304, '46-3435079': 305, '46-3528270': 306, '46-3676810': 307, '46-5671661': 308, '47-0873092': 309, '47-0912342': 310, '47-0933090': 311, '47-3632618': 312, '471809999': 313, '476810': 314, '48-1148159': 315, '48-1236121': 316, '480228': 317, '4878': 318, '491224': 319, '501105': 320, '504713': 321, '51-0488301': 322, '51-0517525': 323, '52-2091594': 324, '52-2276553': 325, '52-2321476': 326, '52-2323186': 327, '5380': 328, '54-0259290': 329, '54-1094297': 330, '54-1994393': 331, '54-2070914': 332, '542409990': 333, '542649': 334, '546571': 335, '55130': 336, '5556209999': 337, '5582': 338, '5588': 339, '56-2103469': 340, '56-2237729': 341, '56-2471041': 342, '566': 343, '57-1175755': 344, '57033': 345, '57071': 346, '57074': 347, '57167': 348, '57451': 349, '57542': 350, '57607': 351, '57614': 352, '57633': 353, '57776': 354, '57777': 355, '57949': 356, '57955': 357, '57978': 358, '58-1865166': 359, '58305': 360, '58322': 361, '58341': 362, '58380': 363, '58778': 364, '59-3378746': 365, '5912': 366, '5913': 367, '592448': 368, '595270': 369, '595869': 370, '60042': 371, '60059': 372, '60079': 373, '601050': 374, '60143': 375, '60438': 376, '606046': 377, '60613': 378, '6158': 379, '6161': 380, '617677': 381, '619877': 382, '62-0997810': 383, '62-1494087': 384, '62-1532940': 385, '624': 386, '6248': 387, '62659': 388, '62665': 389, '62745': 390, '6288': 391, '63-1052225': 392, '63069': 393, '63194': 394, '63196': 395, '6328': 396, '63315': 397, '63440': 398, '63799': 399, '64103': 400, '644': 401, '6443809990': 402, '64482': 403, '64546': 404, '64552': 405, '65595': 406, '656377': 407, '65644': 408, '656733': 409, '66157': 410, '66328': 411, '66331': 412, '66337': 413, '66349': 414, '66373': 415, '66399': 416, '665': 417, '66734': 418, '66751': 419, '66841': 420, '67201': 421, '67262': 422, '67264': 423, '67389': 424, '675332': 425, '67911': 426, '67955': 427, '68-0151632': 428, '68-0295876': 429, '68-0309242': 430, '68061': 431, '68095': 432, '68186': 433, '68187': 434, '68196': 435, '68203': 436, '68205': 437, '68222': 438, '68223': 439, '68237': 440, '68239': 441, '68253': 442, '68255': 443, '68271': 444, '68278': 445, '68284': 446, '68293': 447, '68298': 448, '68304': 449, '68315': 450, '68362': 451, '68375': 452, '68394': 453, '68423': 454, '68457': 455, '68465': 456, '68517': 457, '68530': 458, '68546': 459, '685676': 460, '68576': 461, '68598': 462, '68601': 463, '68613': 464, '68622': 465, '694904': 466, '697633': 467, '702825': 468, '702889': 469, '703927': 470, '704347': 471, '7056000000': 472, '706707': 473, '706809': 474, '707674': 475, '708146': 476, '708186': 477, '7101': 478, '712504': 479, '713570': 480, '713964': 481, '714970': 482, '715018': 483, '716195': 484, '7162800002': 485, '716456': 486, '717936': 487, '7197000003': 488, '72-1545376': 489, '7257500009': 490, '7272800006': 491, '7275700004':

```
492, '73-1374559': 493, '73-1545233': 494, '73-1577221': 495, '74-
2508160': 496, '75-2585326': 497, '75-2695327': 498, '75-2921540':
499, '75-3170028': 500, '7505400005': 501, '7516800003': 502, '75633':
503, '76-0236067': 504, '76-0503625': 505, '76-0561995': 506, '76-
0629353': 507, '7635500004': 508, '7638200000': 509, '7667200009':
510, '7674000006': 511, '77-0158990': 512, '77-0605392': 513, '77-
0672274': 514, '77-0717225': 515, '7748': 516, '7756300009': 517,
'7810600004': 518, '7811300008': 519, '7927200007': 520, '7983500003':
521, '7992700007': 522, '80-0233937': 523, '80-0312140': 524, '80-
0860209': 525, '80122': 526, '804963': 527, '8100': 528, '817824':
529, '83-0171636': 530, '83-0368862': 531, '84-0927358': 532, '84-
1040263': 533, '84-1412422': 534, '84-1496821': 535, '84-1564935':
536, '84-1594306': 537, '85-0260899': 538, '852218': 539, '852320':
540, '857': 541, '86-0415227': 542, '86-0431588': 543, '86-0634557':
544, '86-0860478': 545, '8663': 546, '87-0623581': 547, '87-0675992':
548, '87-0682600': 549, '87-0691650': 550, '8796': 551, '8797': 552,
'88-0209429': 553, '88-0508228': 554, '8854': 555, '90-0790926': 556,
'91-1374387': 557, '91-1395192': 558, '91-1441009': 559, '91-1465333':
560, '91-1529683': 561, '91-1569077': 562, '91-1780488': 563, '91-
1841798': 564, '91-1913382': 565, '91-2006136': 566, '915878': 567,
'9289': 568, '93-1231049': 569, '93-1248952': 570, '93-1296762': 571,
'93-1301081': 572, '934329': 573, '9366': 574, '936855': 575, '9373':
576, '94-3195577': 577, '9483': 578, '9486': 579, '95-3821253': 580,
'95-3990375': 581, '95-4196389': 582, '95-4234730': 583, '95-4267987':
584, '95-4462959': 585, '95-4482547': 586, '95-4523866': 587, '95-
4623407': 588, '95-4762204': 589, '95-4769926': 590, '95-4866828':
591, '972590': 592}

Mapping for purchaser_type_name:
{'Affiliate institution': 0, 'Commercial bank, savings bank or savings
association': 1, 'Fannie Mae (FNMA)': 2, 'Farmer Mac (FAMC)': 3,
'Freddie Mac (FHLMC)': 4, 'Ginnie Mae (GNMA)': 5, 'Life insurance
company, credit union, mortgage bank, or finance company': 6, 'Loan
was not originated or was not sold in calendar year covered by
register': 7, 'Other type of purchaser': 8, 'Private securitization':
9}

Mapping for property_type_name:
{'Manufactured housing': 0, 'Multifamily dwelling': 1, 'One-to-four
family dwelling (other than manufactured housing)': 2}

Mapping for preapproval_name:
{'Not applicable': 0, 'Preapproval was not requested': 1, 'Preapproval
was requested': 2}

Mapping for owner_occupancy_name:
{'Not applicable': 0, 'Not owner-occupied as a principal dwelling': 1,
'Owner-occupied as a principal dwelling': 2}

Mapping for msamd_name:
```

```
{'Bellingham - WA': 0, 'Bremerton, Silverdale - WA': 1, 'Kennewick, Richland - WA': 2, 'Lewiston - ID, WA': 3, 'Longview - WA': 4, 'Mount Vernon, Anacortes - WA': 5, 'Olympia, Tumwater - WA': 6, 'Portland, Vancouver, Hillsboro - OR, WA': 7, 'Seattle, Bellevue, Everett - WA': 8, 'Spokane, Spokane Valley - WA': 9, 'Tacoma, Lakewood - WA': 10, 'Walla Walla - WA': 11, 'Wenatchee - WA': 12, 'Yakima - WA': 13}

Mapping for loan_type_name:
{'Conventional': 0, 'FHA-insured': 1, 'FSA/RHS-guaranteed': 2, 'VA-guaranteed': 3}

Mapping for loan_purpose_name:
{'Home improvement': 0, 'Home purchase': 1, 'Refinancing': 2}

Mapping for lien_status_name:
{'Not applicable': 0, 'Not secured by a lien': 1, 'Secured by a first lien': 2, 'Secured by a subordinate lien': 3}

Mapping for hoepa_status_name:
{'HOEPA loan': 0, 'Not a HOEPA loan': 1}

Mapping for county_name:
{'Adams County': 0, 'Asotin County': 1, 'Benton County': 2, 'Chelan County': 3, 'Clallam County': 4, 'Clark County': 5, 'Columbia County': 6, 'Cowlitz County': 7, 'Douglas County': 8, 'Ferry County': 9, 'Franklin County': 10, 'Garfield County': 11, 'Grant County': 12, 'Grays Harbor County': 13, 'Island County': 14, 'Jefferson County': 15, 'King County': 16, 'Kitsap County': 17, 'Kittitas County': 18, 'Klickitat County': 19, 'Lewis County': 20, 'Lincoln County': 21, 'Mason County': 22, 'Okanogan County': 23, 'Pacific County': 24, 'Pend Oreille County': 25, 'Pierce County': 26, 'San Juan County': 27, 'Skagit County': 28, 'Skamania County': 29, 'Snohomish County': 30, 'Spokane County': 31, 'Stevens County': 32, 'Thurston County': 33, 'Wahkiakum County': 34, 'Walla Walla County': 35, 'Whatcom County': 36, 'Whitman County': 37, 'Yakima County': 38}

Mapping for co_applicant_sex_name:
{'Female': 0, 'Information not provided by applicant in mail, Internet, or telephone application': 1, 'Male': 2, 'No co-applicant': 3, 'Not applicable': 4}

Mapping for co_applicant_ethnicity_name:
{'Hispanic or Latino': 0, 'Information not provided by applicant in mail, Internet, or telephone application': 1, 'No co-applicant': 2, 'Not Hispanic or Latino': 3, 'Not applicable': 4}

Mapping for applicant_sex_name:
{'Female': 0, 'Information not provided by applicant in mail, Internet, or telephone application': 1, 'Male': 2, 'Not applicable': 3}
```

```
Mapping for applicant_ethnicity_name:
{'Hispanic or Latino': 0, 'Information not provided by applicant in
mail, Internet, or telephone application': 1, 'Not Hispanic or
Latino': 2, 'Not applicable': 3}

Mapping for agency_name:
{'Consumer Financial Protection Bureau': 0, 'Department of Housing and
Urban Development': 1, 'Federal Deposit Insurance Corporation': 2,
'Federal Reserve System': 3, 'National Credit Union Administration':
4, 'Office of the Comptroller of the Currency': 5}

Mapping for agency_abbr:
{'CFPB': 0, 'FDIC': 1, 'FRS': 2, 'HUD': 3, 'NCUA': 4, 'OCC': 5}

Mapping for action_taken_name:
{'Application approved but not accepted': 0, 'Application denied by
financial institution': 1, 'Application withdrawn by applicant': 2,
'File closed for incompleteness': 3, 'Loan originated': 4, 'Loan
purchased by the institution': 5, 'Preapproval request approved but
not accepted': 6, 'Preapproval request denied by financial
institution': 7}
       state_name  state_abbr  respondent_id  purchaser_type_name  \
0               0           0            317                    4
1               0           0            490                    6
2               0           0            489                    7
3               0           0            318                    7
4               0           0            234                    4
...           ...         ...            ...                  ...
59995           0           0            472                    8
59996           0           0            488                    4
59997           0           0             55                    5
59998           0           0            116                    5
59999           0           0             47                    2

       property_type_name  preapproval_name  owner_occupancy_name
msamd_name  \
0                       2                 0                     2
7
1                       2                 0                     2
11
2                       2                 0                     2
7
3                       2                 0                     2
7
4                       2                 0                     2
1
...                   ...               ...                   ...
...
59995                   2                 0                     2
```

```
7
59996                    2                   0                       1
5
59997                    2                   0                       2
0
59998                    2                   0                       2
1
59999                    2                   0                       2
7

       loan_type_name  loan_purpose_name  lien_status_name
hoepa_status_name  \
0                   0                  2                 2
1
1                   1                  1                 2
1
2                   0                  2                 2
1
3                   0                  2                 2
1
4                   0                  0                 2
1
...               ...                ...               ...
...
59995               0                  2                 2
1
59996               0                  2                 2
1
59997               1                  0                 2
1
59998               3                  2                 2
1
59999               0                  2                 2
1

       county_name  co_applicant_sex_name  co_applicant_ethnicity_name
\
0                5                      2                            3

1               35                      3                            2

2                5                      0                            3

3                5                      0                            1

4               17                      2                            3

...            ...                    ...                          ...

59995            5                      0                            0
```

```
59996           28                       3                            2

59997           36                       0                            3

59998           17                       0                            3

59999            5                       0                            3


       applicant_sex_name  applicant_ethnicity_name  agency_name
agency_abbr  \
0                       0                         2            0
0
1                       2                         0            1
3
2                       2                         2            1
3
3                       2                         1            4
4
4                       0                         2            2
1
...                   ...                       ...          ...
...
59995                   2                         2            1
3
59996                   0                         1            1
3
59997                   2                         2            1
3
59998                   2                         2            0
0
59999                   2                         2            0
0

       action_taken_name
0                      4
1                      4
2                      4
3                      4
4                      4
...                  ...
59995                  4
59996                  4
59997                  4
59998                  4
59999                  4

[60000 rows x 20 columns]
time: 781 ms (started: 2024-03-16 13:28:15 +00:00)
```

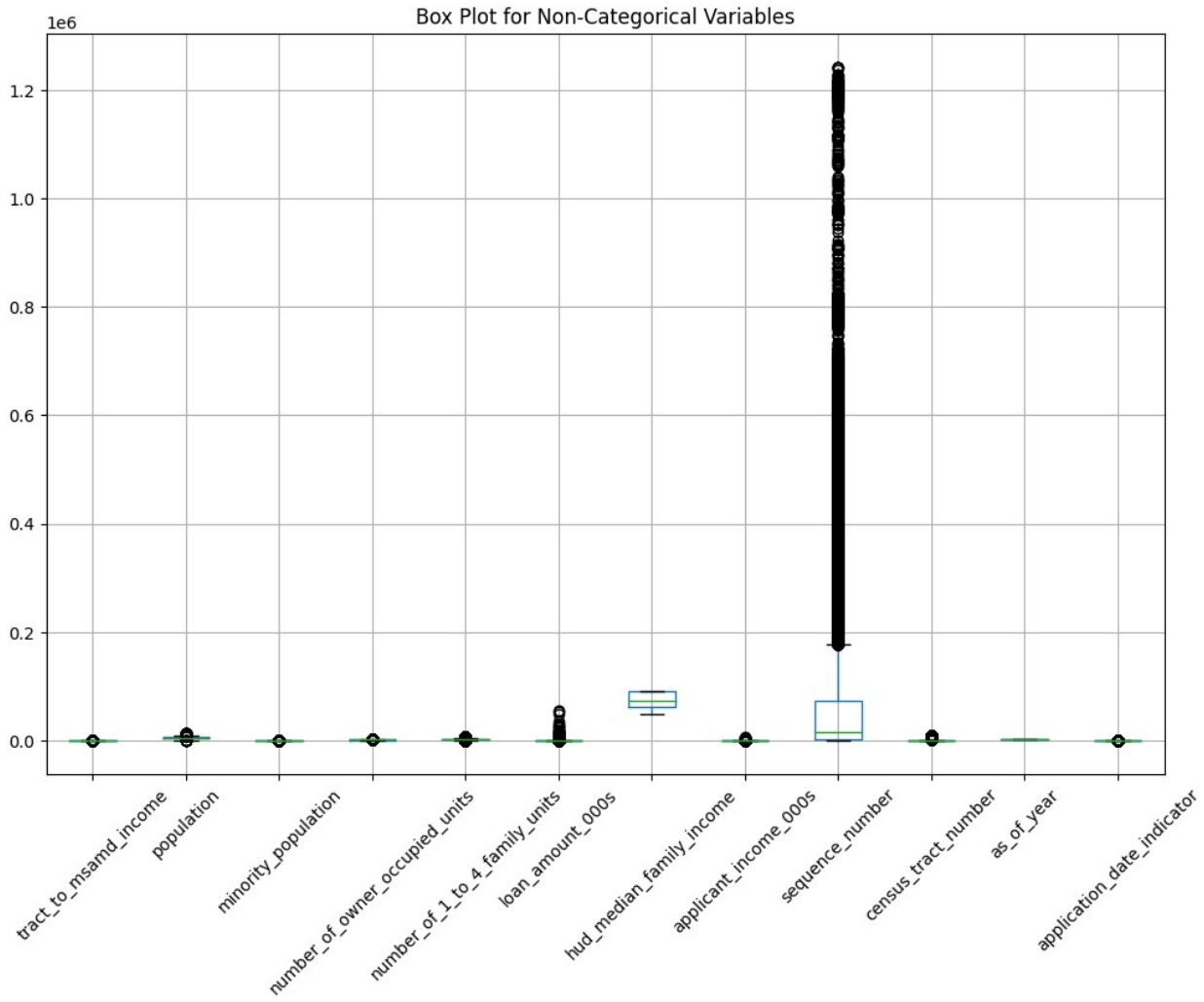
```
print(imputed_data_non_categorical.columns)
```

```
Index(['tract_to_msamd_income', 'population', 'minority_population',
       'number_of_owner_occupied_units',
'number_of_1_to_4_family_units',
       'loan_amount_000s', 'hud_median_family_income',
'applicant_income_000s',
       'sequence_number', 'census_tract_number', 'as_of_year',
       'application_date_indicator'],
      dtype='object')
time: 3.19 ms (started: 2024-03-16 13:28:16 +00:00)
```

```
def identify_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (column < lower_bound) | (column > upper_bound)
    return outliers

# Apply the function to each column to get a DataFrame of True/False
values
outliers = imputed_data_non_categorical.apply(identify_outliers)

# Display the number of outliers for each column
outlier_counts = outliers.sum()
print(outlier_counts)
```

```
tract_to_msamd_income             1309
population                         553
minority_population               2641
number_of_owner_occupied_units     478
number_of_1_to_4_family_units     2150
loan_amount_000s                  2467
hud_median_family_income             0
applicant_income_000s             3765
sequence_number                   7898
census_tract_number               9391
as_of_year                           0
application_date_indicator         776
dtype: int64
time: 55.1 ms (started: 2024-03-16 13:28:16 +00:00)
```

```
# Create a box plot for non-categorical variables
imputed_data_non_categorical.boxplot(rot=45, figsize=(12, 8))
plt.title('Box Plot for Non-Categorical Variables')
plt.show()
```




```
time: 1.12 s (started: 2024-03-16 13:28:16 +00:00)

# Iterate through each column and print count of unique values
for column in imputed_data_non_categorical.columns:
    unique_count = imputed_data_non_categorical[column].nunique()
    print(f"Count of unique values in {column} column:
{unique_count}")

Count of unique values in tract_to_msamd_income column: 1328
Count of unique values in population column: 1284
Count of unique values in minority_population column: 1217
Count of unique values in number_of_owner_occupied_units column: 997
Count of unique values in number_of_1_to_4_family_units column: 1052
Count of unique values in loan_amount_000s column: 1344
Count of unique values in hud_median_family_income column: 16
Count of unique values in applicant_income_000s column: 808
Count of unique values in sequence_number column: 39340
Count of unique values in census_tract_number column: 1109
Count of unique values in as_of_year column: 1
```

```
Count of unique values in application_date_indicator column: 2
time: 28.3 ms (started: 2024-03-16 13:28:17 +00:00)
```

```
# Initialize the StandardScaler
scaler = StandardScaler()

# Apply Standard Scaling to your dataset
scaled_data = scaler.fit_transform(imputed_data_non_categorical)

def identify_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (column < lower_bound) | (column > upper_bound)
    return outliers

# Apply the function to each column in the scaled dataset
outliers_scaled = pd.DataFrame(scaled_data,
columns=imputed_data_non_categorical.columns).apply(identify_outliers)

# Display the number of outliers for each column in the scaled dataset
outlier_counts_scaled = outliers_scaled.sum()
print(outlier_counts_scaled)
```

```
tract_to_msamd_income              1309
population                          553
minority_population                2641
number_of_owner_occupied_units      478
number_of_1_to_4_family_units      2150
loan_amount_000s                   2467
hud_median_family_income              0
applicant_income_000s              3765
sequence_number                    7898
census_tract_number                9391
as_of_year                            0
application_date_indicator          776
dtype: int64
time: 52.8 ms (started: 2024-03-16 13:28:17 +00:00)
```

```
# Initialize the RobustScaler
scaler = RobustScaler()

# Apply Robust Scaling to your dataset
scaled_data_robust =
scaler.fit_transform(imputed_data_non_categorical)

# Check for outliers in the scaled dataset
outliers_robust = pd.DataFrame(scaled_data_robust,
columns=imputed_data_non_categorical.columns).apply(identify_outliers)
```

```
# Display the number of outliers for each column in the scaled dataset
outlier_counts_robust = outliers_robust.sum()
print(outlier_counts_robust)
```

```
tract_to_msamd_income              1309
population                          553
minority_population                2641
number_of_owner_occupied_units      478
number_of_1_to_4_family_units      2150
loan_amount_000s                   2467
hud_median_family_income              0
applicant_income_000s              3765
sequence_number                    7898
census_tract_number                9391
as_of_year                            0
application_date_indicator          776
dtype: int64
time: 101 ms (started: 2024-03-16 13:28:17 +00:00)
```

```
# Define columns to exclude from normalization
columns_to_exclude = ['hud_median_family_income', 'as_of_year',
'application_date_indicator']

# Create a copy of the DataFrame with excluded columns
data_to_scale =
imputed_data_non_categorical.drop(columns=columns_to_exclude)

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Apply Min-Max Scaling to the selected columns
scaled_data = scaler.fit_transform(data_to_scale)

# Create a DataFrame with scaled data and original column names
scaled_df = pd.DataFrame(scaled_data, columns=data_to_scale.columns)

# Add back the excluded columns to the scaled DataFrame
scaled_df[columns_to_exclude] =
imputed_data_non_categorical[columns_to_exclude]

# Display the scaled DataFrame
print(scaled_df)
```

```
       tract_to_msamd_income  population  minority_population  \
0                   0.442799    0.640752            0.234501
1                   0.285162    0.372631            0.236658
2                   0.317084    0.385008            0.105445
3                   0.543502    0.381682            0.070620
4                   0.610556    0.393363            0.091213
...                      ...         ...                 ...
```

```
59995                0.394915    0.299373             0.149434
59996                0.445679    0.369923             0.060162
59997                0.418734    0.609964             0.105553
59998                0.336419    0.179392             0.322803
59999                0.442799    0.640752             0.234501

       number_of_owner_occupied_units
number_of_1_to_4_family_units  \
0                            0.724346                          0.448858

1                            0.420188                          0.298329

2                            0.375922                          0.308728

3                            0.506372                          0.305660

4                            0.566734                          0.354074

...                               ...                               ...

59995                        0.370557                          0.240709

59996                        0.556673                          0.383396

59997                        0.783032                          0.558814

59998                        0.195171                          0.116263

59999                        0.724346                          0.448858


       loan_amount_000s  applicant_income_000s  sequence_number  \
0              0.004109               0.018669         0.096625
1              0.004346               0.006656         0.042368
2              0.004364               0.018831         0.005001
3              0.006364               0.050974         0.000158
4              0.007564               0.018344         0.026241
...                 ...                    ...              ...
59995          0.002927               0.012013         0.024452
59996          0.002564               0.007955         0.268165
59997          0.004546               0.014123         0.020504
59998          0.004655               0.018153         0.289037
59999          0.005309               0.013149         0.032247

       census_tract_number  hud_median_family_income  as_of_year  \
0                 0.042258                   73300.0      2016.0
1                 0.943728                   57900.0      2016.0
2                 0.042333                   73300.0      2016.0
3                 0.041421                   73300.0      2016.0
4                 0.092866                   78100.0      2016.0
...                    ...                       ...         ...
```

```
59995              0.041926                    73300.0      2016.0
59996              0.963715                    61400.0      2016.0
59997              0.000724                    69900.0      2016.0
59998              0.082002                    78100.0      2016.0
59999              0.042258                    73300.0      2016.0
```

```
       application_date_indicator
0                             0.0
1                             0.0
2                             0.0
3                             0.0
4                             0.0
...                           ...
59995                         0.0
59996                         0.0
59997                         0.0
59998                         0.0
59999                         0.0

[60000 rows x 12 columns]
time: 41.5 ms (started: 2024-03-16 13:28:17 +00:00)
```

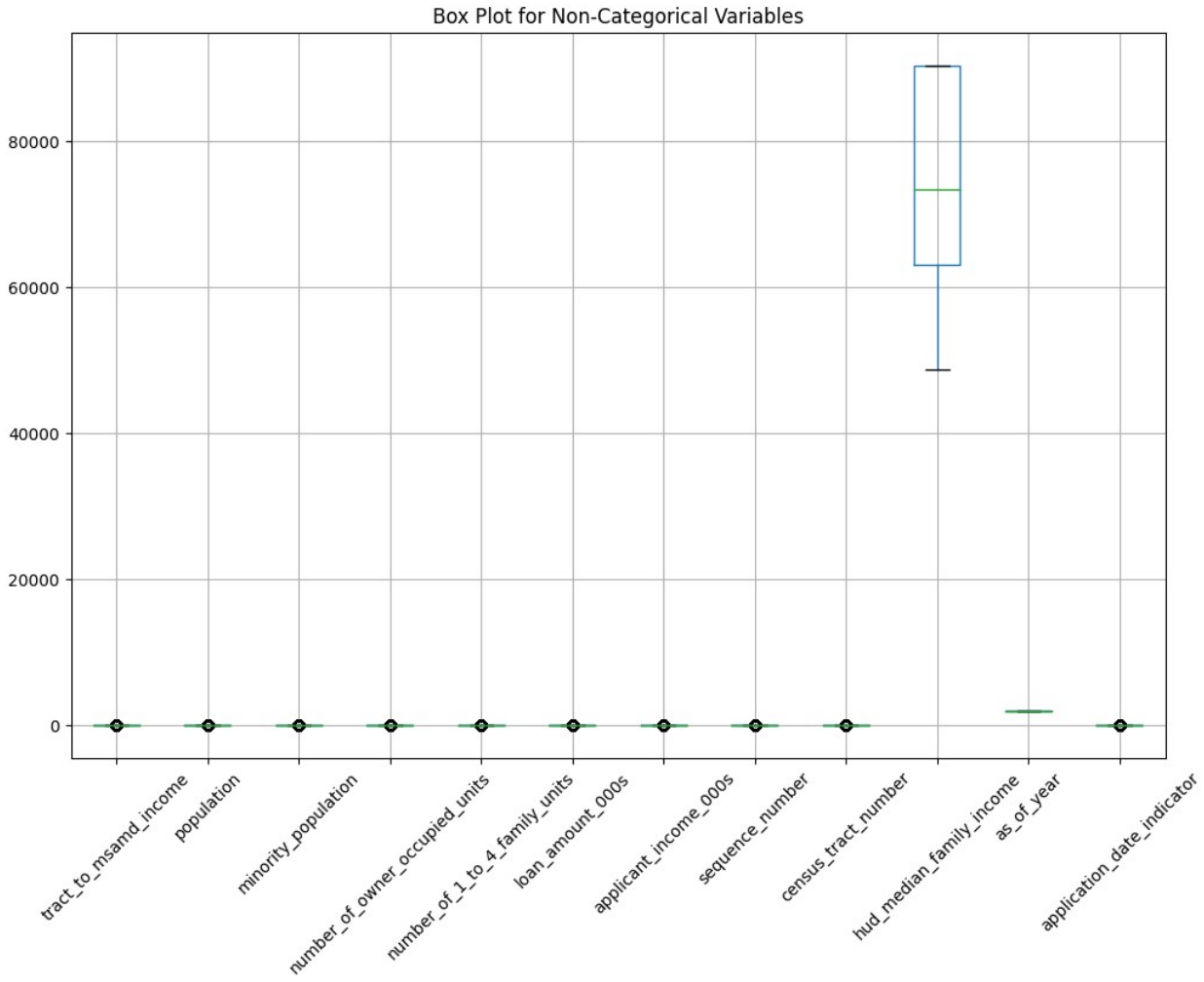
```
def identify_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (column < lower_bound) | (column > upper_bound)
    return outliers

# Apply the function to each column in the scaled dataset
scaled_outliers = scaled_df.apply(identify_outliers)

# Display the number of outliers for each column in the scaled dataset
scaled_outlier_counts = scaled_outliers.sum()
print(scaled_outlier_counts)
```

```
tract_to_msamd_income             1309
population                         553
minority_population               2641
number_of_owner_occupied_units     478
number_of_1_to_4_family_units     2150
loan_amount_000s                  2467
applicant_income_000s             3765
sequence_number                   7898
census_tract_number               9391
hud_median_family_income             0
as_of_year                           0
application_date_indicator         776
```

```
dtype: int64
time: 56.5 ms (started: 2024-03-16 13:28:17 +00:00)
```

```
def identify_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers1 = (column < lower_bound) | (column > upper_bound)
    return outliers1

# Apply the function to each column to get a DataFrame of True/False
values
outliers1 = scaled_df.apply(identify_outliers)

# Display the number of outliers for each column
outlier_counts1 = outliers1.sum()
print(outlier_counts1)


# Create a box plot for non-categorical variables
scaled_df.boxplot(rot=45, figsize=(12, 8))
plt.title('Box Plot for Non-Categorical Variables')
plt.show()
```

```
tract_to_msamd_income              1309
population                          553
minority_population                2641
number_of_owner_occupied_units      478
number_of_1_to_4_family_units      2150
loan_amount_000s                   2467
applicant_income_000s              3765
sequence_number                    7898
census_tract_number                9391
hud_median_family_income              0
as_of_year                            0
application_date_indicator          776
dtype: int64
```




```
time: 1.13 s (started: 2024-03-16 13:28:17 +00:00)
```

```
# Calculate standard deviation for non-categorical columns
std_deviation_non_categorical1 = scaled_df.std()

# Creating a DataFrame to display the results
dispersion_non_categorical_df1 = pd.DataFrame({
    'Variable': scaled_df.columns,
    'Standard Deviation': std_deviation_non_categorical1.values
})

print(dispersion_non_categorical_df1)
```

```
                         Variable  Standard Deviation
0            tract_to_msamd_income            0.116026
1                       population            0.132618
2              minority_population            0.155273
3   number_of_owner_occupied_units            0.173640
4    number_of_1_to_4_family_units            0.125768
```

```
5                  loan_amount_000s             0.010999
6             applicant_income_000s             0.018857
7                   sequence_number             0.121228
8               census_tract_number             0.344020
9         hud_median_family_income         12798.211781
10                      as_of_year             0.000000
11      application_date_indicator             0.225976
time: 17.1 ms (started: 2024-03-16 13:28:18 +00:00)
```

```
# Assuming encoded_data_categorical and scaled_df have the same number
of rows
combined_data = pd.concat([encoded_data_categorical, scaled_df],
axis=1)

# Display the combined DataFrame
print(combined_data)


list(combined_data.columns)
```

```
       state_name  state_abbr  respondent_id  purchaser_type_name  \
0               0           0            317                    4
1               0           0            490                    6
2               0           0            489                    7
3               0           0            318                    7
4               0           0            234                    4
...           ...         ...            ...                  ...
59995           0           0            472                    8
59996           0           0            488                    4
59997           0           0             55                    5
59998           0           0            116                    5
59999           0           0             47                    2

       property_type_name  preapproval_name  owner_occupancy_name
msamd_name  \
0                       2                 0                     2
7
1                       2                 0                     2
11
2                       2                 0                     2
7
3                       2                 0                     2
7
4                       2                 0                     2
1
...                   ...               ...                   ...
...
59995                   2                 0                     2
7
59996                   2                 0                     1
```

```
5
59997                    2                  0                       2
0
59998                    2                  0                       2
1
59999                    2                  0                       2
7

       loan_type_name  loan_purpose_name  ...  minority_population  \
0                   0                  2  ...             0.234501
1                   1                  1  ...             0.236658
2                   0                  2  ...             0.105445
3                   0                  2  ...             0.070620
4                   0                  0  ...             0.091213
...               ...                ...  ...                  ...
59995               0                  2  ...             0.149434
59996               0                  2  ...             0.060162
59997               1                  0  ...             0.105553
59998               3                  2  ...             0.322803
59999               0                  2  ...             0.234501

       number_of_owner_occupied_units
number_of_1_to_4_family_units  \
0                            0.724346                          0.448858

1                            0.420188                          0.298329

2                            0.375922                          0.308728

3                            0.506372                          0.305660

4                            0.566734                          0.354074

...                               ...                               ...

59995                        0.370557                          0.240709

59996                        0.556673                          0.383396

59997                        0.783032                          0.558814

59998                        0.195171                          0.116263

59999                        0.724346                          0.448858


       loan_amount_000s  applicant_income_000s  sequence_number  \
0              0.004109               0.018669         0.096625
1              0.004346               0.006656         0.042368
2              0.004364               0.018831         0.005001
3              0.006364               0.050974         0.000158
```

```
4              0.007564                0.018344         0.026241
...                 ...                     ...              ...
59995          0.002927                0.012013         0.024452
59996          0.002564                0.007955         0.268165
59997          0.004546                0.014123         0.020504
59998          0.004655                0.018153         0.289037
59999          0.005309                0.013149         0.032247

       census_tract_number  hud_median_family_income  as_of_year  \
0                 0.042258                   73300.0      2016.0
1                 0.943728                   57900.0      2016.0
2                 0.042333                   73300.0      2016.0
3                 0.041421                   73300.0      2016.0
4                 0.092866                   78100.0      2016.0
...                    ...                       ...         ...
59995             0.041926                   73300.0      2016.0
59996             0.963715                   61400.0      2016.0
59997             0.000724                   69900.0      2016.0
59998             0.082002                   78100.0      2016.0
59999             0.042258                   73300.0      2016.0

       application_date_indicator
0                             0.0
1                             0.0
2                             0.0
3                             0.0
4                             0.0
...                           ...
59995                         0.0
59996                         0.0
59997                         0.0
59998                         0.0
59999                         0.0

[60000 rows x 32 columns]

['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
```

```
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name',
 'tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']
```

```
time: 44.3 ms (started: 2024-03-16 13:28:18 +00:00)
```

```
df_ppd_subset = combined_data.copy()
```

```
time: 22.5 ms (started: 2024-03-16 13:28:18 +00:00)
```

```
# Create K-Means Clusters [K=2]
km_2cluster = kmclus(n_clusters=2, init='random', random_state=333)
df_ppd_subset['Cluster_Label'] =
km_2cluster.fit_predict(df_ppd_subset)
km_2cluster_model = df_ppd_subset['Cluster_Label'].values
km_2cluster_model
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
```

```
array([0, 0, 0, ..., 0, 1, 0], dtype=int32)
```

```
time: 296 ms (started: 2024-03-16 13:28:18 +00:00)
```

```
# K-Means Clustering Model Evaluation [K=2]
# ----------------------------------------------------
```

```
sscore_km_2cluster = sscore(df_ppd_subset, km_2cluster_model)
dbscore_km_2cluster = dbscore(df_ppd_subset, km_2cluster_model);
%memit
print(f"Davies-Bouldin Index for 2 clusters: {dbscore_km_2cluster}")
print(f"Silhouette Score for 2 clusters: {sscore_km_2cluster}")
```

```
peak memory: 406.48 MiB, increment: 0.19 MiB
Davies-Bouldin Index for 2 clusters: 0.4619936534703312
Silhouette Score for 2 clusters: 0.663051131384813
time: 1min 17s (started: 2024-03-16 13:28:19 +00:00)
```

```
list(scaled_df.columns)
```

```
['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']
```

```
time: 5.75 ms (started: 2024-03-16 13:29:37 +00:00)
```

```
# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_2cluster_model

# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
group in cluster_groups])

# Print ANOVA results for non-categorical variables
for column, result in anova_results_non_cat.items():
```

```
    print(f"Variable: {column}")
    print(f"F-value: {result.statistic}")
    print(f"P-value: {result.pvalue}")
    print()
```

```
Variable: tract_to_msamd_income
F-value: 27.519663767225108
P-value: 1.5603202366422437e-07

Variable: population
F-value: 0.030971142300885723
P-value: 0.860305378609647

Variable: minority_population
F-value: 4314.228447264084
P-value: 0.0

Variable: number_of_owner_occupied_units
F-value: 52.91842156268415
P-value: 3.519255050690632e-13

Variable: number_of_1_to_4_family_units
F-value: 2261.8846751490787
P-value: 0.0

Variable: loan_amount_000s
F-value: 924.7921291408038
P-value: 1.3729320983717837e-201

Variable: applicant_income_000s
F-value: 1584.4166238495632
P-value: 0.0

Variable: sequence_number
F-value: 142.3008030500535
P-value: 9.103498523541926e-33

Variable: census_tract_number
F-value: 5364.32086304639
P-value: 0.0

Variable: hud_median_family_income
F-value: 146150.09501056315
P-value: 0.0

Variable: as_of_year
F-value: nan
P-value: nan

Variable: application_date_indicator
F-value: 374.6253681685772
```

```
P-value: 3.3066791591424276e-83

time: 155 ms (started: 2024-03-16 13:29:37 +00:00)

/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning: Each of the input arrays is constant;the F
statistic is not defined or infinite
  warnings.warn(stats.ConstantInputWarning(msg))
```

```
list(encoded_data_categorical.columns)
```

```
['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

time: 3.41 ms (started: 2024-03-16 13:29:37 +00:00)
```

```
from scipy.stats import chi2_contingency

# Extracting categorical variables
cat_variables = ['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
```

```
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Perform Chi-square test for each categorical variable
chi2_results_cat = {}
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_2cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'Chi-square': chi2, 'P-value':
p_value}

# Print Chi-square test results for categorical variables
for column, result in chi2_results_cat.items():
    print(f"Variable: {column}")
    print(f"Chi-square: {result['Chi-square']}")
    print(f"P-value: {result['P-value']}")
    print()
```

```
Variable: state_name
Chi-square: 0.0
P-value: 1.0

Variable: state_abbr
Chi-square: 0.0
P-value: 1.0

Variable: respondent_id
Chi-square: 13677.950380187096
P-value: 0.0

Variable: purchaser_type_name
Chi-square: 1506.292344864866
P-value: 0.0

Variable: property_type_name
Chi-square: 533.6583222543006
P-value: 1.310894410252237e-116

Variable: preapproval_name
Chi-square: 305.30512741422757
P-value: 5.056312807421688e-67

Variable: owner_occupancy_name
Chi-square: 144.36627239407184
P-value: 4.4798375238593323e-32
```

```
Variable: msamd_name
Chi-square: 35095.2220412316
P-value: 0.0

Variable: loan_type_name
Chi-square: 2162.5530864683587
P-value: 0.0

Variable: loan_purpose_name
Chi-square: 95.30503575818713
P-value: 2.0173183375889264e-21

Variable: lien_status_name
Chi-square: 397.031412952757
P-value: 9.731198995627216e-86

Variable: hoepa_status_name
Chi-square: 0.005827388854829811
P-value: 0.9391507227238625

Variable: county_name
Chi-square: 59567.13592698854
P-value: 0.0

Variable: co_applicant_sex_name
Chi-square: 344.4704836923513
P-value: 2.7404452102196065e-73

Variable: co_applicant_ethnicity_name
Chi-square: 395.1904283867685
P-value: 3.044090809109492e-84

Variable: applicant_sex_name
Chi-square: 266.74790558766233
P-value: 1.5596885828947085e-57

Variable: applicant_ethnicity_name
Chi-square: 417.42422551304344
P-value: 3.7217458435272255e-90

Variable: agency_name
Chi-square: 3064.84069511026
P-value: 0.0

Variable: agency_abbr
Chi-square: 3064.84069511026
P-value: 0.0

Variable: action_taken_name
```

```
Chi-square: 2049.5385487539634
P-value: 0.0

time: 531 ms (started: 2024-03-16 13:29:37 +00:00)
```

```
# Get the cluster centers
cluster_centers = km_2cluster.cluster_centers_

# Convert cluster_centers to a DataFrame
centroids_df = pd.DataFrame(cluster_centers,
columns=df_ppd_subset.columns[:-1])

# Display the centroids of clusters
centroids_df
```

{"type":"dataframe","variable_name":"centroids_df"}

```
time: 68 ms (started: 2024-03-16 13:29:37 +00:00)
```

```
df_ppd_subset['Cluster_Label'] = km_2cluster_model

# Calculate cluster-wise descriptive statistics
cluster_stats = df_ppd_subset.groupby('Cluster_Label').describe()

# Variables of interest
variables_of_interest = ['tract_to_msamd_income',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'application_date_indicator']

# Print descriptive statistics for each variable
for variable in variables_of_interest:
    print(f"Descriptive statistics for variable: {variable}")
    print(cluster_stats[variable])
    print()
```

```
Descriptive statistics for variable: tract_to_msamd_income
                count      mean       std      min       25%
50%  \
Cluster_Label

0               38597.0  0.383058  0.105904  0.01555  0.313053
0.378872
1               21403.0  0.388244  0.132274  0.00000  0.297626
0.372043
```

```
                     75%      max
Cluster_Label
0              0.444568  0.77206
1              0.463450  1.00000

Descriptive statistics for variable: minority_population
                count      mean       std       min       25%
50%  \
Cluster_Label

0              38597.0  0.198672  0.143935  0.000000  0.104043
0.160970
1              21403.0  0.282625  0.160294  0.046469  0.158706
0.243235

                     75%       max
Cluster_Label
0              0.243450  1.000000
1              0.360863  0.961617

Descriptive statistics for variable: number_of_owner_occupied_units
                count      mean       std       min       25%
50%  \
Cluster_Label

0              38597.0  0.467971  0.181445  0.000000  0.339705
0.447686
1              21403.0  0.457211  0.158365  0.000671  0.346747
0.453052

                     75%       max
Cluster_Label
0              0.580483  1.000000
1              0.563045  0.909792

Descriptive statistics for variable: number_of_1_to_4_family_units
                count      mean       std       min       25%
50%  \
Cluster_Label

0              38597.0  0.332594  0.137154  0.002387  0.242755
0.310603
1              21403.0  0.282552  0.093853  0.000000  0.223832
0.283669

                     75%       max
Cluster_Label
0              0.398738  1.000000
1              0.344016  0.554893
```

```
Descriptive statistics for variable: loan_amount_000s
                count      mean       std  min       25%       50%
75%  \
Cluster_Label

0              38597.0  0.004270  0.009418  0.0  0.002746  0.003836
0.005073
1              21403.0  0.007099  0.013193  0.0  0.004164  0.005782
0.007691

                    max
Cluster_Label
0              0.949999
1              1.000000

Descriptive statistics for variable: applicant_income_000s
                count      mean       std       min       25%
50%  \
Cluster_Label

0              38597.0  0.015901  0.013001  0.000000  0.009253
0.013799
1              21403.0  0.022215  0.025814  0.000487  0.012825
0.018153

                    75%       max
Cluster_Label
0              0.018153  0.417045
1              0.024838  1.000000

Descriptive statistics for variable: sequence_number
                count      mean       std  min       25%       50%
75%  \
Cluster_Label

0              38597.0  0.066832  0.126497  0.0  0.002489  0.013954
0.061254
1              21403.0  0.054522  0.110660  0.0  0.002789  0.012049
0.048500

                    max
Cluster_Label
0              1.000000
1              0.999993

Descriptive statistics for variable: census_tract_number
                count      mean       std  min       25%       50%
75%  \
Cluster_Label
```

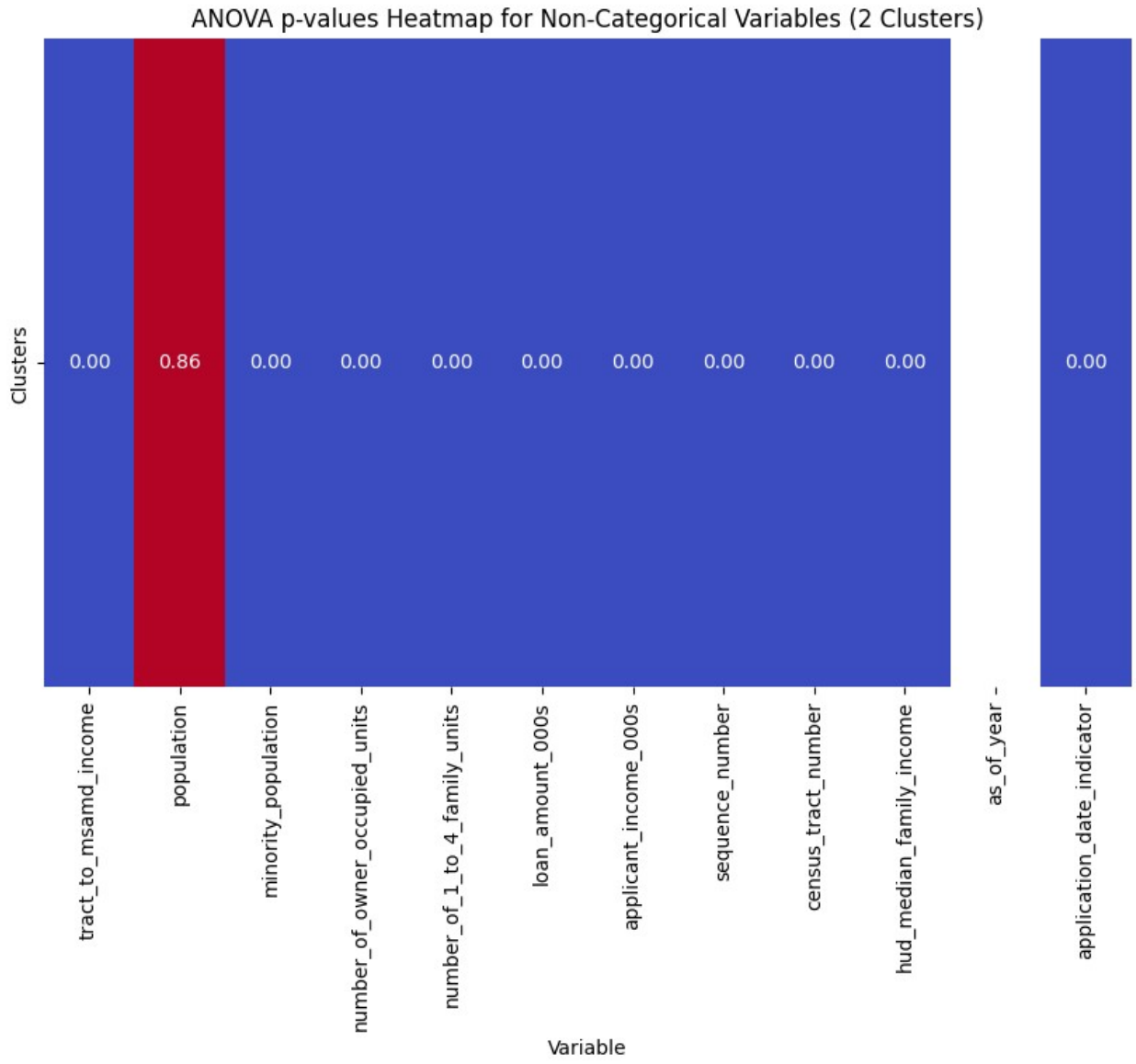
```
0              38597.0  0.252725  0.405494  0.0  0.011072  0.041618
0.074847
1              21403.0  0.046989  0.089629  0.0  0.022453  0.032603
0.053316

                    max
Cluster_Label
0              1.00000
1              0.96351

Descriptive statistics for variable: hud_median_family_income
                 count          mean          std      min      25%
50%  \
Cluster_Label

0              38597.0  65844.945673  7935.884805  48700.0  55600.0
69900.0
1              21403.0  88340.293417  4479.833489  78100.0  90300.0
90300.0

                    75%           max
Cluster_Label
0              73300.0  73869.411136
1              90300.0  90300.000000

Descriptive statistics for variable: application_date_indicator
                 count      mean       std  min  25%  50%  75%  max
Cluster_Label
0              38597.0  0.039122  0.276976  0.0  0.0  0.0  0.0  2.0
1              21403.0  0.001962  0.062618  0.0  0.0  0.0  0.0  2.0

time: 882 ms (started: 2024-03-16 13:29:38 +00:00)
```

```python
# Assuming km_2cluster_model contains the cluster labels

# Grouping the data by cluster labels
cluster_groups = df_with_clusters.groupby(km_2cluster_model)

# Counting the number of variables in each cluster
num_variables_in_clusters = cluster_groups.size()

# Displaying the results
print("Number of Variables in Each Cluster:")
print(num_variables_in_clusters)
```

```
Number of Variables in Each Cluster:
0    38597
1    21403
dtype: int64
time: 19.3 ms (started: 2024-03-16 13:29:38 +00:00)
```

```
# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_2cluster_model

# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
group in cluster_groups])

# Extract p-values for non-categorical variables
non_cat_p_values = [result.pvalue for result in
anova_results_non_cat.values()]

# Plotting heatmap for non-categorical variables
plt.figure(figsize=(10, 6))
sns.heatmap([non_cat_p_values], cmap='coolwarm', annot=True,
fmt='.2f', xticklabels=non_cat_variables, yticklabels=['Clusters'],
cbar=False)
plt.xlabel('Variable')
plt.title('ANOVA p-values Heatmap for Non-Categorical Variables (2
Clusters)')
plt.show()
```

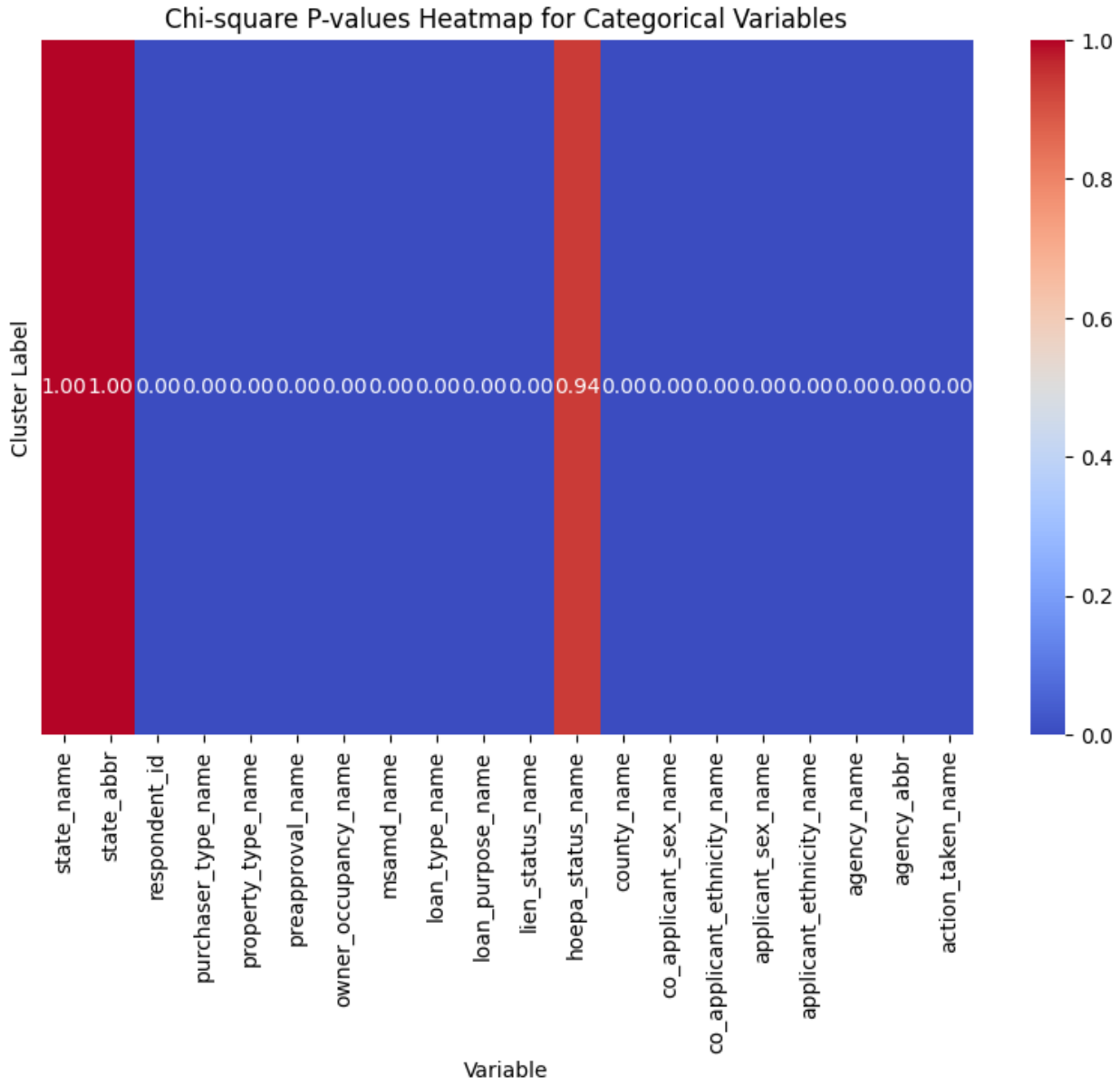
```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning: Each of the input arrays is constant;the F
statistic is not defined or infinite
  warnings.warn(stats.ConstantInputWarning(msg))
```




```
time: 784 ms (started: 2024-03-16 13:29:39 +00:00)
```

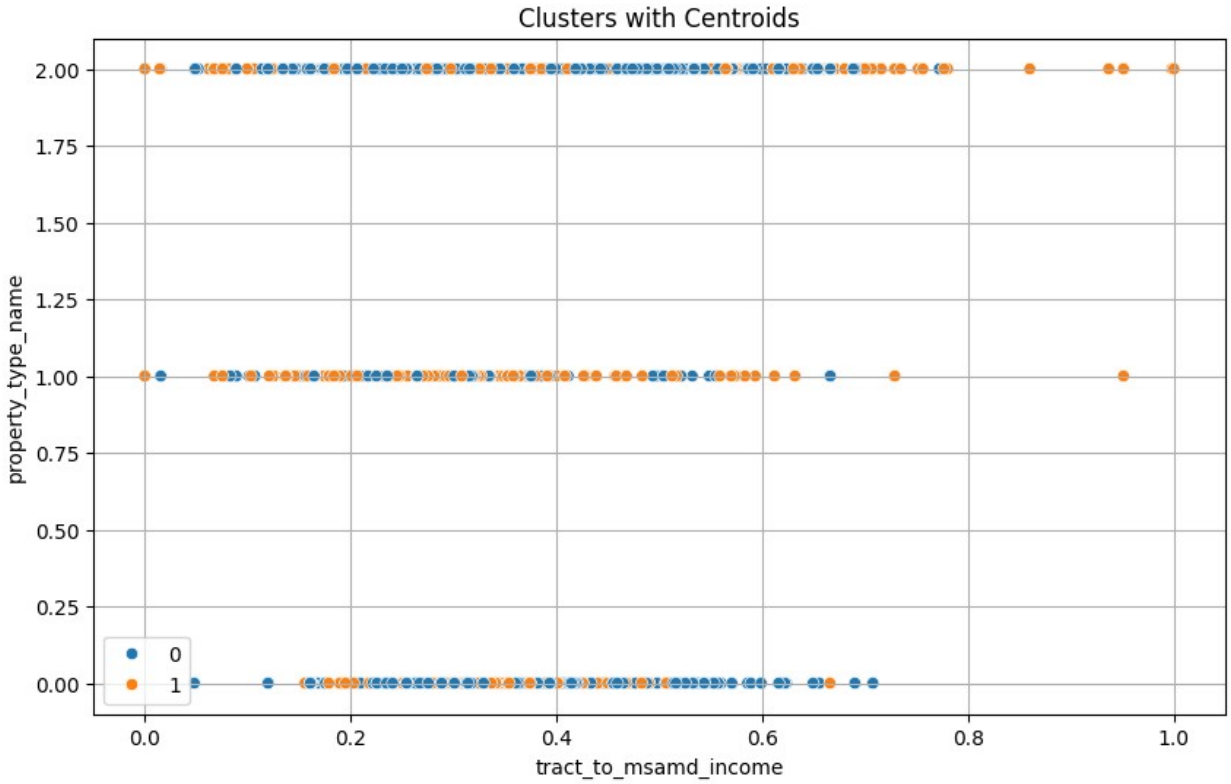
```
cat_variables = ['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
```

```
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Initialize dictionary to store chi-square results
chi2_results_cat = {}

# Calculate chi-square for each categorical variable
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_2cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'P-value': p_value}

# Extract p-values
p_values = [[result['P-value'] for result in
chi2_results_cat.values()]]

# Get variable names
variables = list(chi2_results_cat.keys())

# Plotting heatmap for p-values
plt.figure(figsize=(10, 6))
sns.heatmap(p_values, cmap='coolwarm', annot=True, fmt='.2f',
xticklabels=variables, yticklabels=False)
plt.xlabel('Variable')
plt.ylabel('Cluster Label')
plt.title('Chi-square P-values Heatmap for Categorical Variables')
plt.show()
```




```
time: 1.67 s (started: 2024-03-16 13:29:39 +00:00)

# Assign cluster labels to the DataFrame
df_ppd_subset['Cluster_Label'] = km_2cluster_model

# Plot the scatter plot with clusters and centroids
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_ppd_subset, x='tract_to_msamd_income',
y='property_type_name', hue='Cluster_Label', legend='full')

# Set plot title and labels
plt.title('Clusters with Centroids')
plt.xlabel('tract_to_msamd_income')
```

```
plt.ylabel('property_type_name')
plt.legend()
plt.grid(True)
%memit
plt.show()

peak memory: 423.85 MiB, increment: 0.00 MiB
```

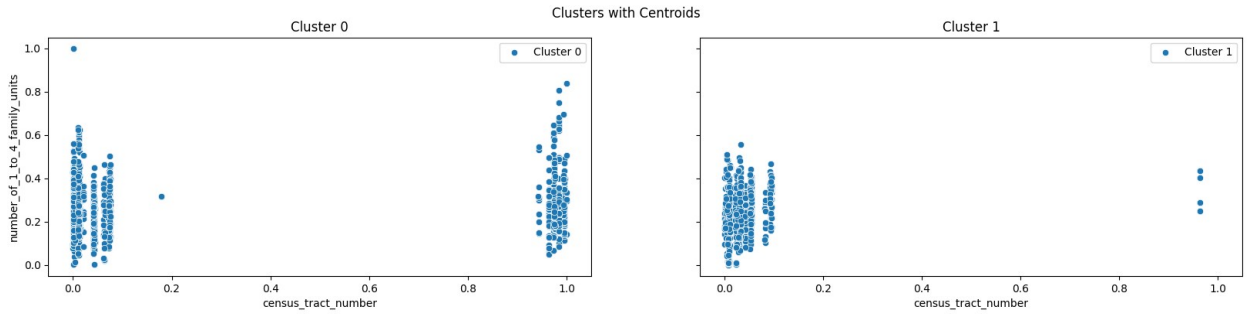



```
time: 6.58 s (started: 2024-03-16 13:29:41 +00:00)

import plotly.graph_objs as go

# Create 3D scatter plot
fig = go.Figure()

# Add traces for each cluster
for cluster_label in df_ppd_subset['Cluster_Label'].unique():
    cluster_data = df_ppd_subset[df_ppd_subset['Cluster_Label'] ==
cluster_label]
    fig.add_trace(go.Scatter3d(
        x=cluster_data['tract_to_msamd_income'],
        y=cluster_data['loan_amount_000s'],
        z=cluster_data['hud_median_family_income'],
        mode='markers',
        marker=dict(
```

```
            size=5,
            color=cluster_label,
            colorscale='Viridis',  # Adjust colorscale if needed
            opacity=0.8
        ),
        name=f'Cluster {cluster_label}'
    ))

# Set layout
fig.update_layout(
    title='Clusters with Centroids (3D)',
    scene=dict(
        xaxis=dict(title='tract_to_msamd_income'),
        yaxis=dict(title='loan_amount_000s'),
        zaxis=dict(title='hud_median_family_income'),
    ),
    margin=dict(l=0, r=0, b=0, t=40)
)

# Show interactive 3D scatter plot
fig.show()
```

```
time: 1.51 s (started: 2024-03-16 13:29:48 +00:00)
```

```
# Create subplots for each cluster
fig, axes = plt.subplots(1, 2, figsize=(20, 4), sharex=True,
sharey=True)
fig.suptitle('Clusters with Centroids')

# Iterate through each cluster
for i in range(2):
    # Filter data points belonging to the current cluster
    cluster_data = df_ppd_subset[df_ppd_subset['Cluster_Label'] == i]

    # Scatter plot of data points
    sns.scatterplot(data=cluster_data, x='census_tract_number',
y='number_of_1_to_4_family_units', ax=axes[i], label=f'Cluster {i}')

    # Set title and labels for each subplot
    axes[i].set_title(f'Cluster {i}')
    axes[i].set_xlabel('census_tract_number')
    axes[i].set_ylabel('number_of_1_to_4_family_units')
    axes[i].legend()

plt.show()
%memit
```




```
peak memory: 425.66 MiB, increment: 0.00 MiB
time: 3.15 s (started: 2024-03-16 13:29:49 +00:00)

#########       KKKKKKKKKK    ===========     333333333333

time: 475 µs (started: 2024-03-16 13:29:52 +00:00)

# Create K-Means Clusters [K=3]
km_3cluster = kmclus(n_clusters=3, init='random', random_state=333)
df_ppd_subset['Cluster_Label'] =
km_3cluster.fit_predict(df_ppd_subset)
km_3cluster_model = df_ppd_subset['Cluster_Label'].values
km_3cluster_model

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning


array([0, 1, 0, ..., 0, 0, 0], dtype=int32)

time: 1.08 s (started: 2024-03-16 13:29:52 +00:00)

# K-Means Clustering Model Evaluation [K=3]
# ----------------------------------------------------

sscore_km_3cluster = sscore(df_ppd_subset, km_3cluster_model)
dbscore_km_3cluster = dbscore(df_ppd_subset, km_3cluster_model);
%memit
print(f"Davies-Bouldin Index for 3 clusters: {dbscore_km_3cluster}")
print(f"Silhouette Score for 3 clusters: {sscore_km_3cluster}")

peak memory: 434.50 MiB, increment: 0.00 MiB
Davies-Bouldin Index for 3 clusters: 0.2801372945964262
Silhouette Score for 3 clusters: 0.8003178033339671
time: 1min 14s (started: 2024-03-16 13:29:53 +00:00)

# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_3cluster_model
```

```python
# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
group in cluster_groups])

# Print ANOVA results for non-categorical variables
for column, result in anova_results_non_cat.items():
    print(f"Variable: {column}")
    print(f"F-value: {result.statistic}")
    print(f"P-value: {result.pvalue}")
    print()
```

```
Variable: tract_to_msamd_income
F-value: 25.29673766271471
P-value: 1.0432682323355863e-11

Variable: population
F-value: 8.168053006844962
P-value: 0.000283885036873319

Variable: minority_population
F-value: 3189.773620360581
P-value: 0.0

Variable: number_of_owner_occupied_units
F-value: 193.6598534295485
P-value: 1.4618712456960466e-84

Variable: number_of_1_to_4_family_units
F-value: 3074.2751445775048
P-value: 0.0
```

```
Variable: loan_amount_000s
F-value: 536.7077920387354
P-value: 9.361471848138305e-232

Variable: applicant_income_000s
F-value: 913.9693822860053
P-value: 0.0

Variable: sequence_number
F-value: 124.06118967363064
P-value: 1.706138677651434e-54

Variable: census_tract_number
F-value: 20446.636928775097
P-value: 0.0

Variable: hud_median_family_income
F-value: 519880.84699270705
P-value: 0.0

Variable: as_of_year
F-value: nan
P-value: nan

Variable: application_date_indicator
F-value: 1015.3399507415708
P-value: 0.0

time: 165 ms (started: 2024-03-16 13:31:08 +00:00)

/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning:

Each of the input arrays is constant;the F statistic is not defined or
infinite
```

```python
from scipy.stats import chi2_contingency

# Extracting categorical variables
cat_variables = ['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
```

```
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Perform Chi-square test for each categorical variable
chi2_results_cat = {}
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_3cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'Chi-square': chi2, 'P-value':
p_value}

# Print Chi-square test results for categorical variables
for column, result in chi2_results_cat.items():
    print(f"Variable: {column}")
    print(f"Chi-square: {result['Chi-square']}")
    print(f"P-value: {result['P-value']}")
    print()
```

```
Variable: state_name
Chi-square: 0.0
P-value: 1.0

Variable: state_abbr
Chi-square: 0.0
P-value: 1.0

Variable: respondent_id
Chi-square: 23221.71658104706
P-value: 0.0

Variable: purchaser_type_name
Chi-square: 2728.4024483581015
P-value: 0.0

Variable: property_type_name
Chi-square: 1193.2917166872726
P-value: 4.533566444037516e-257

Variable: preapproval_name
Chi-square: 297.1963916737516
P-value: 4.3606364753985263e-63
```

```
Variable: owner_occupancy_name
Chi-square: 281.7851951753754
P-value: 9.185061445027928e-60

Variable: msamd_name
Chi-square: 78493.27301942065
P-value: 0.0

Variable: loan_type_name
Chi-square: 3575.3546254802704
P-value: 0.0

Variable: loan_purpose_name
Chi-square: 108.57838291283734
P-value: 1.4626619455705175e-22

Variable: lien_status_name
Chi-square: 2023.1788946193697
P-value: 0.0

Variable: hoepa_status_name
Chi-square: 1.5893148166793993
P-value: 0.45173596939148075

Variable: county_name
Chi-square: 119369.61531546856
P-value: 0.0

Variable: co_applicant_sex_name
Chi-square: 1511.0647773498515
P-value: 0.0

Variable: co_applicant_ethnicity_name
Chi-square: 1499.0663557884432
P-value: 0.0

Variable: applicant_sex_name
Chi-square: 657.0462670876913
P-value: 1.1454120451923734e-138

Variable: applicant_ethnicity_name
Chi-square: 772.5107108026517
P-value: 1.3378219235026382e-163

Variable: agency_name
Chi-square: 3964.8364288058638
P-value: 0.0

Variable: agency_abbr
```

```
Chi-square: 3964.8364288058638
P-value: 0.0

Variable: action_taken_name
Chi-square: 5094.33254943791
P-value: 0.0

time: 460 ms (started: 2024-03-16 13:31:08 +00:00)
```

```
# Get the cluster centers
cluster_centers = km_3cluster.cluster_centers_

# Convert cluster_centers to a DataFrame
centroids_df = pd.DataFrame(cluster_centers,
columns=df_ppd_subset.columns)

# Display the centroids of clusters
centroids_df
```

```
{"type":"dataframe","variable_name":"centroids_df"}

time: 38.8 ms (started: 2024-03-16 13:31:09 +00:00)
```

```
# Calculate cluster-wise descriptive statistics
cluster_stats = df_ppd_subset.groupby('Cluster_Label').describe()

# Print the descriptive statistics for each cluster
print("Cluster-wise Descriptive Statistics:")
cluster_stats
```

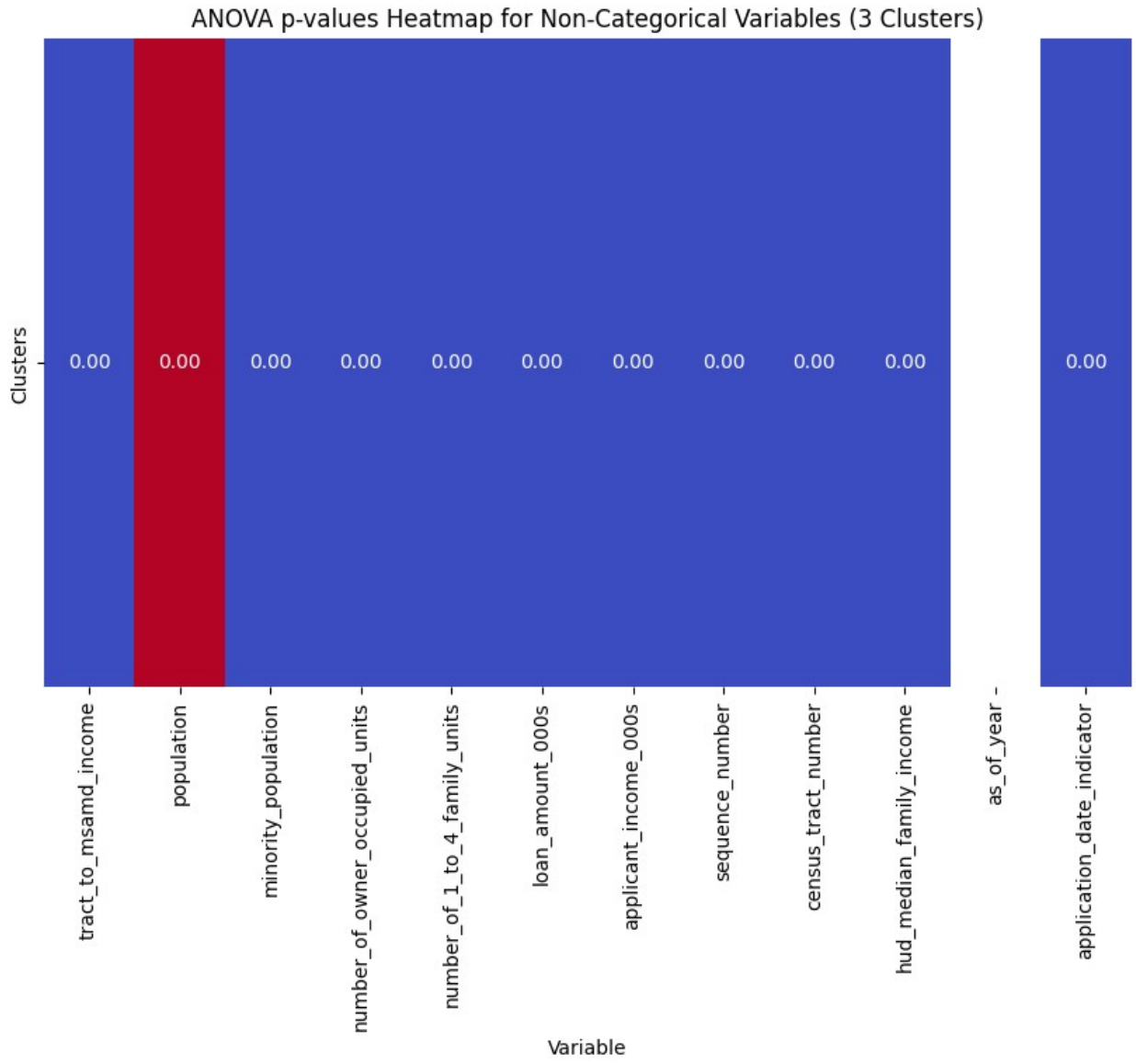
```
Cluster-wise Descriptive Statistics:

{"type":"dataframe","variable_name":"cluster_stats"}

time: 397 ms (started: 2024-03-16 13:31:09 +00:00)
```

```
# Grouping the data by cluster labels
cluster_groups = df_with_clusters.groupby(km_3cluster_model)

# Counting the number of variables in each cluster
num_variables_in_clusters = cluster_groups.size()

# Displaying the results
print("Number of Variables in Each Cluster:")
print(num_variables_in_clusters)
```

```
Number of Variables in Each Cluster:
0    26674
1    15361
2    17965
dtype: int64
time: 6.07 ms (started: 2024-03-16 13:31:09 +00:00)
```

```
# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_3cluster_model

# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
group in cluster_groups])

# Extract p-values for non-categorical variables
non_cat_p_values = [result.pvalue for result in
anova_results_non_cat.values()]

# Plotting heatmap for non-categorical variables
plt.figure(figsize=(10, 6))
sns.heatmap([non_cat_p_values], cmap='coolwarm', annot=True,
fmt='.2f', xticklabels=non_cat_variables, yticklabels=['Clusters'],
cbar=False)
plt.xlabel('Variable')
plt.title('ANOVA p-values Heatmap for Non-Categorical Variables (3
Clusters)')
plt.show()
```

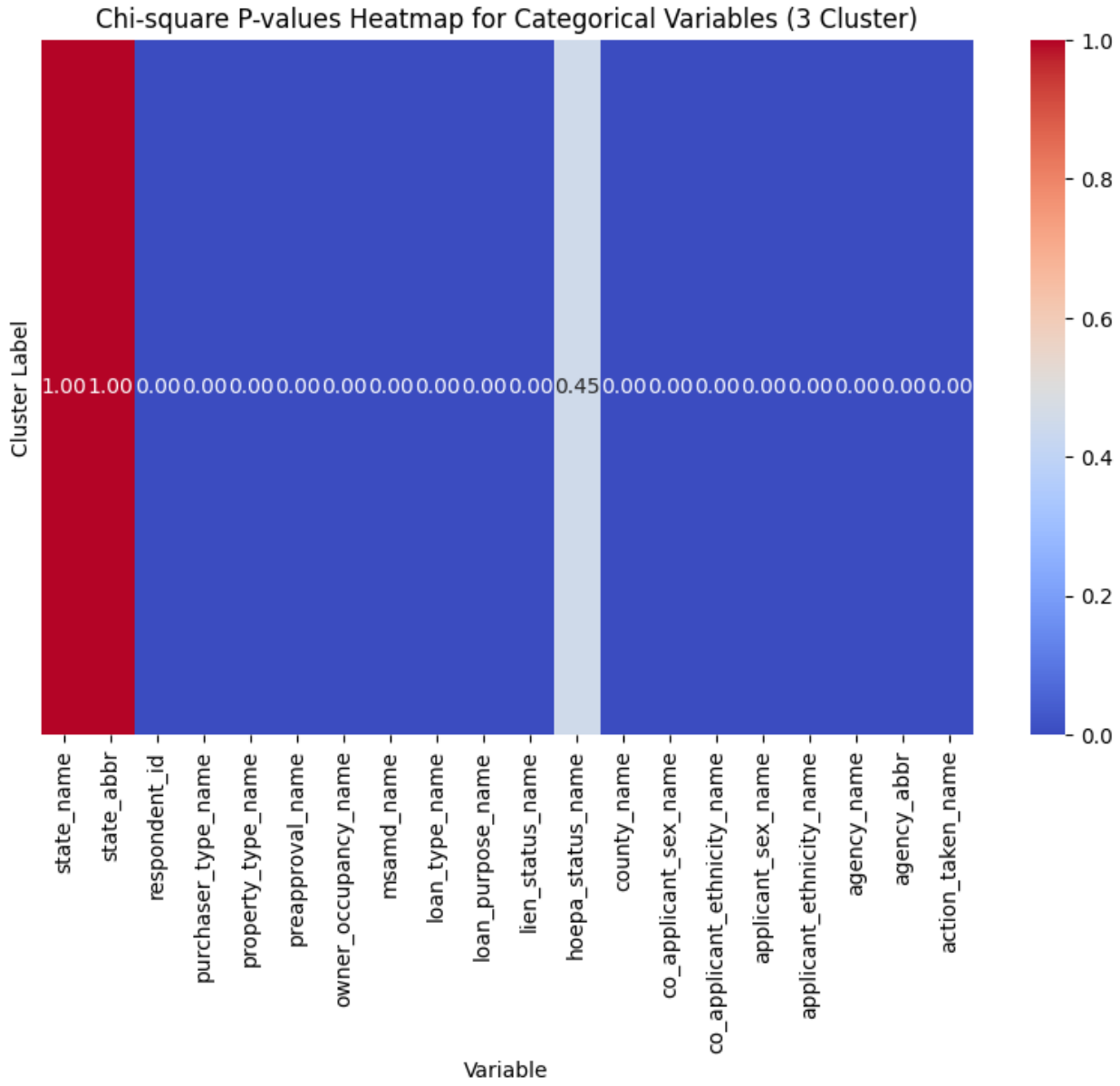
```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning:

Each of the input arrays is constant;the F statistic is not defined or
infinite
```




```
time: 539 ms (started: 2024-03-16 13:31:09 +00:00)
```

```
cat_variables = ['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
```

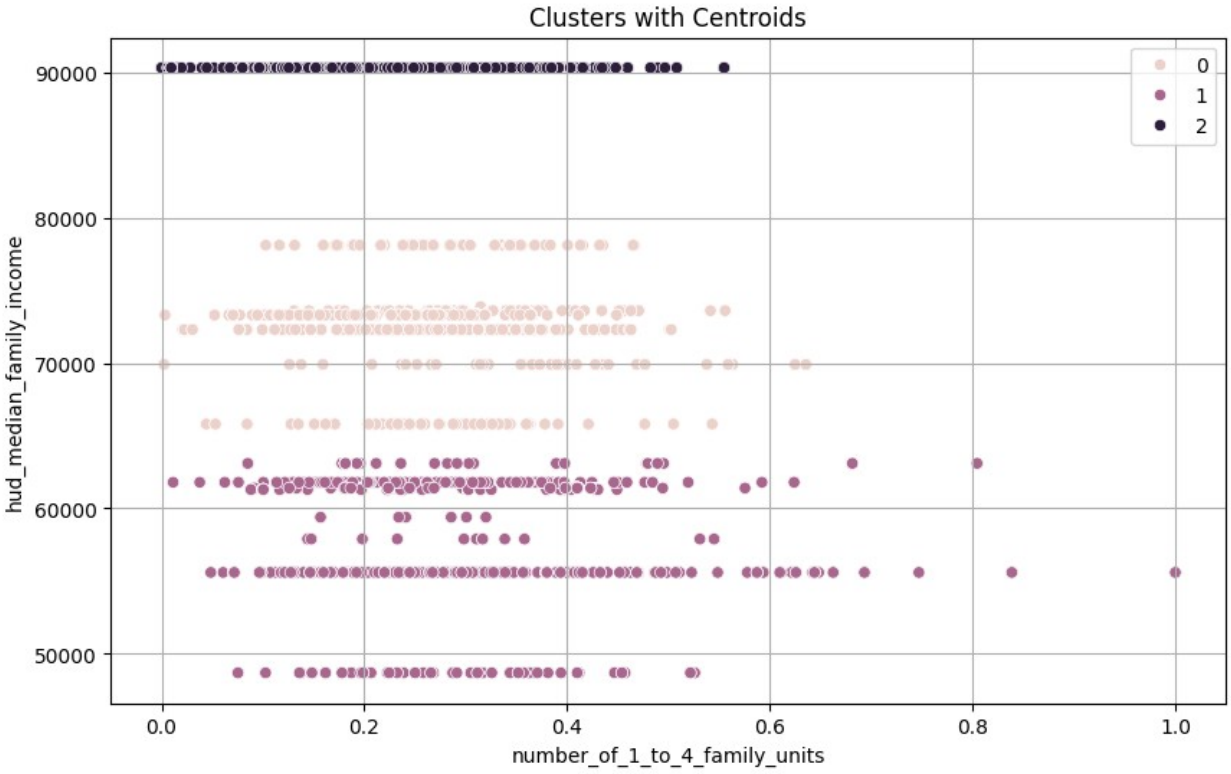
```
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Initialize dictionary to store chi-square results
chi2_results_cat = {}

# Calculate chi-square for each categorical variable
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_3cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'P-value': p_value}

# Extract p-values
p_values = [[result['P-value'] for result in
chi2_results_cat.values()]]

# Get variable names
variables = list(chi2_results_cat.keys())

# Plotting heatmap for p-values
plt.figure(figsize=(10, 6))
sns.heatmap(p_values, cmap='coolwarm', annot=True, fmt='.2f',
xticklabels=variables, yticklabels=False)
plt.xlabel('Variable')
plt.ylabel('Cluster Label')
plt.title('Chi-square P-values Heatmap for Categorical Variables (3
Cluster)')
plt.show()
```




```
time: 1.45 s (started: 2024-03-16 13:31:10 +00:00)

# Assign cluster labels to the DataFrame
df_ppd_subset['Cluster_Label'] = km_3cluster_model

# Plot the scatter plot with clusters and centroids
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_ppd_subset, x='number_of_1_to_4_family_units',
y='hud_median_family_income', hue='Cluster_Label', legend='full')

# Set plot title and labels
plt.title('Clusters with Centroids')
plt.xlabel('number_of_1_to_4_family_units')
```

```
plt.ylabel('hud_median_family_income')
plt.legend()
plt.grid(True)
%memit
plt.show()
```

```
peak memory: 427.11 MiB, increment: -0.98 MiB
```



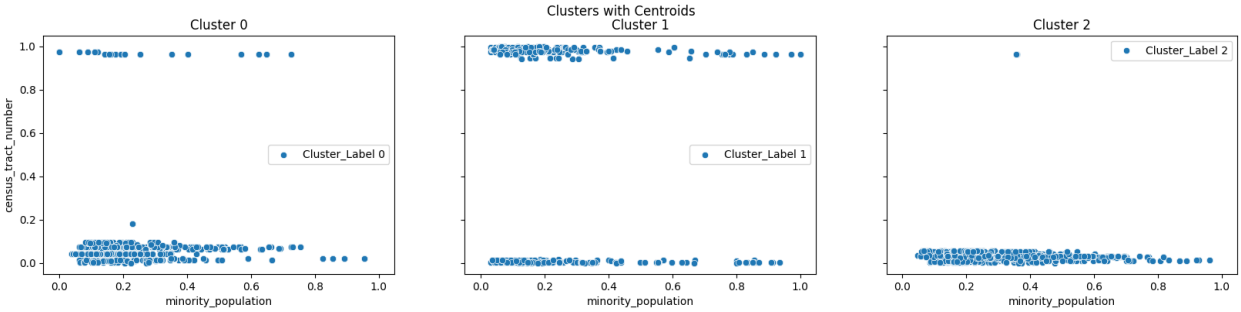

```
time: 5.66 s (started: 2024-03-16 13:31:11 +00:00)
```

```
import plotly.graph_objs as go
import numpy as np

# Create 3D scatter plot
fig = go.Figure()

# Get unique cluster labels
unique_clusters = np.unique(km_3cluster_model)

# Add traces for each cluster
for cluster_label in unique_clusters:
    cluster_data = df_ppd_subset[km_3cluster_model == cluster_label]
    fig.add_trace(go.Scatter3d(
        x=cluster_data['hud_median_family_income'],
        y=cluster_data['applicant_income_000s'],
        z=cluster_data['minority_population'],
```

```
        mode='markers',
        marker=dict(
            size=5,
            color=cluster_label,
            colorscale='Viridis',  # Adjust colorscale if needed
            opacity=0.8
        ),
        name=f'Cluster {cluster_label}'
    ))

# Set layout
fig.update_layout(
    title='Clusters with Centroids (3D)',
    scene=dict(
        xaxis=dict(title='hud_median_family_income'),
        yaxis=dict(title='applicant_income_000s'),
        zaxis=dict(title='minority_population'),
    ),
    margin=dict(l=0, r=0, b=0, t=40)
)

# Show interactive 3D scatter plot
fig.show()
```

```
time: 183 ms (started: 2024-03-16 13:31:17 +00:00)
```

```
# Create subplots for each cluster
fig, axes = plt.subplots(1, 3, figsize=(20, 4), sharex=True,
sharey=True)
fig.suptitle('Clusters with Centroids')

# Iterate through each cluster
for i in range(3):
    # Filter data points belonging to the current cluster
    cluster_data = df_ppd_subset[df_ppd_subset['Cluster_Label'] == i]

    # Scatter plot of data points
    sns.scatterplot(data=cluster_data, x='minority_population',
y='census_tract_number', ax=axes[i], label=f'Cluster_Label {i}')

    # Set title and labels for each subplot
    axes[i].set_title(f'Cluster {i}')
    axes[i].set_xlabel('minority_population')
    axes[i].set_ylabel('census_tract_number')
    axes[i].legend()

plt.show()
%memit
```




```
peak memory: 429.14 MiB, increment: 0.00 MiB
time: 3.47 s (started: 2024-03-16 13:31:17 +00:00)

# K=4

time: 618 µs (started: 2024-03-16 13:31:21 +00:00)

# Create K-Means Clusters [K=4]
km_4cluster = kmclus(n_clusters=4, init='random', random_state=333)
df_ppd_subset['Cluster_Label'] =
km_4cluster.fit_predict(df_ppd_subset)
km_4cluster_model = df_ppd_subset['Cluster_Label'].values
km_4cluster_model

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning


array([3, 1, 3, ..., 3, 3, 3], dtype=int32)

time: 769 ms (started: 2024-03-16 13:31:21 +00:00)

############        KKKKKKKKKKKKKK        ==============
4444444444444444

time: 370 µs (started: 2024-03-16 13:31:22 +00:00)

# K-Means Clustering Model Evaluation [K=4]
# ----------------------------------------------------

sscore_km_4cluster = sscore(df_ppd_subset, km_4cluster_model)
dbscore_km_4cluster = dbscore(df_ppd_subset, km_4cluster_model);
%memit
print(f"Davies-Bouldin Index for 4 clusters: {dbscore_km_4cluster}")
print(f"Silhouette Score for 4 clusters: {sscore_km_4cluster}")

peak memory: 435.36 MiB, increment: 0.00 MiB
Davies-Bouldin Index for 4 clusters: 0.32886818262255224
```

```
Silhouette Score for 4 clusters: 0.8316755139608607
time: 1min 6s (started: 2024-03-16 13:31:22 +00:00)
```

```
# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_4cluster_model

# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
group in cluster_groups])

# Print ANOVA results for non-categorical variables
for column, result in anova_results_non_cat.items():
    print(f"Variable: {column}")
    print(f"F-value: {result.statistic}")
    print(f"P-value: {result.pvalue}")
    print()
```

```
Variable: tract_to_msamd_income
F-value: 116.46075670315373
P-value: 3.3591311435931583e-75

Variable: population
F-value: 160.2340103092766
P-value: 1.8831380726119466e-103

Variable: minority_population
F-value: 2052.5765129599467
P-value: 0.0

Variable: number_of_owner_occupied_units
```

```
F-value: 212.99694244929935
P-value: 1.9128448472234274e-137

Variable: number_of_1_to_4_family_units
F-value: 2377.0990174010044
P-value: 0.0

Variable: loan_amount_000s
F-value: 365.8248021976416
P-value: 1.8134725333189733e-235

Variable: applicant_income_000s
F-value: 616.0724987454291
P-value: 0.0

Variable: sequence_number
F-value: 83.1358491107596
P-value: 1.1358688699169335e-53

Variable: census_tract_number
F-value: 10490.45836348815
P-value: 0.0

Variable: hud_median_family_income
F-value: 928866.2876657086
P-value: 0.0

Variable: as_of_year
F-value: nan
P-value: nan

Variable: application_date_indicator
F-value: 1146.2606066568228
P-value: 0.0

time: 162 ms (started: 2024-03-16 13:32:28 +00:00)

/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning:

Each of the input arrays is constant;the F statistic is not defined or
infinite


from scipy.stats import chi2_contingency

# Extracting categorical variables
cat_variables = ['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
```

```
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Perform Chi-square test for each categorical variable
chi2_results_cat = {}
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_4cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'Chi-square': chi2, 'P-value':
p_value}

# Print Chi-square test results for categorical variables
for column, result in chi2_results_cat.items():
    print(f"Variable: {column}")
    print(f"Chi-square: {result['Chi-square']}")
    print(f"P-value: {result['P-value']}")
    print()

Variable: state_name
Chi-square: 0.0
P-value: 1.0

Variable: state_abbr
Chi-square: 0.0
P-value: 1.0

Variable: respondent_id
Chi-square: 30682.275596241423
P-value: 0.0

Variable: purchaser_type_name
Chi-square: 3280.2325414766296
P-value: 0.0

Variable: property_type_name
```

```
Chi-square: 1293.604763053993
P-value: 2.6251307592996707e-276

Variable: preapproval_name
Chi-square: 231.08688270862203
P-value: 4.488473379152535e-47

Variable: owner_occupancy_name
Chi-square: 323.4276530444764
P-value: 7.769853285841703e-67

Variable: msamd_name
Chi-square: 127258.67917720857
P-value: 0.0

Variable: loan_type_name
Chi-square: 3841.8742390807242
P-value: 0.0

Variable: loan_purpose_name
Chi-square: 197.4432157933294
P-value: 6.642669805903239e-40

Variable: lien_status_name
Chi-square: 3316.5803471847057
P-value: 0.0

Variable: hoepa_status_name
Chi-square: 1.9606176827433834
P-value: 0.5806207543407746

Variable: county_name
Chi-square: 179282.81787812975
P-value: 0.0

Variable: co_applicant_sex_name
Chi-square: 2245.2798800116875
P-value: 0.0

Variable: co_applicant_ethnicity_name
Chi-square: 2385.0869290986975
P-value: 0.0

Variable: applicant_sex_name
Chi-square: 871.9013308628205
P-value: 6.996982061129339e-182

Variable: applicant_ethnicity_name
Chi-square: 1384.0610130073328
P-value: 2.1476458074629814e-292
```

```
Variable: agency_name
Chi-square: 4358.65762161032
P-value: 0.0

Variable: agency_abbr
Chi-square: 4358.65762161032
P-value: 0.0

Variable: action_taken_name
Chi-square: 7990.153372773316
P-value: 0.0

time: 563 ms (started: 2024-03-16 13:32:28 +00:00)

cluster_centers = km_4cluster.cluster_centers_

# Convert centroids to DataFrame for better visualization
centroids_df = pd.DataFrame(cluster_centers,
columns=df_ppd_subset.columns)
%memit
print("Centroids of Clusters:")
centroids_df

peak memory: 438.29 MiB, increment: 0.00 MiB
Centroids of Clusters:

{"type":"dataframe","variable_name":"centroids_df"}

time: 508 ms (started: 2024-03-16 13:32:29 +00:00)

# Calculate cluster-wise descriptive statistics
cluster_stats = df_ppd_subset.groupby('Cluster_Label').describe()

# Print the descriptive statistics for each cluster
print("Cluster-wise Descriptive Statistics:")
cluster_stats

Cluster-wise Descriptive Statistics:

{"type":"dataframe","variable_name":"cluster_stats"}

time: 900 ms (started: 2024-03-16 13:32:30 +00:00)

# Grouping the data by cluster labels
cluster_groups = df_with_clusters.groupby(km_4cluster_model)

# Counting the number of variables in each cluster
num_variables_in_clusters = cluster_groups.size()

# Displaying the results
```

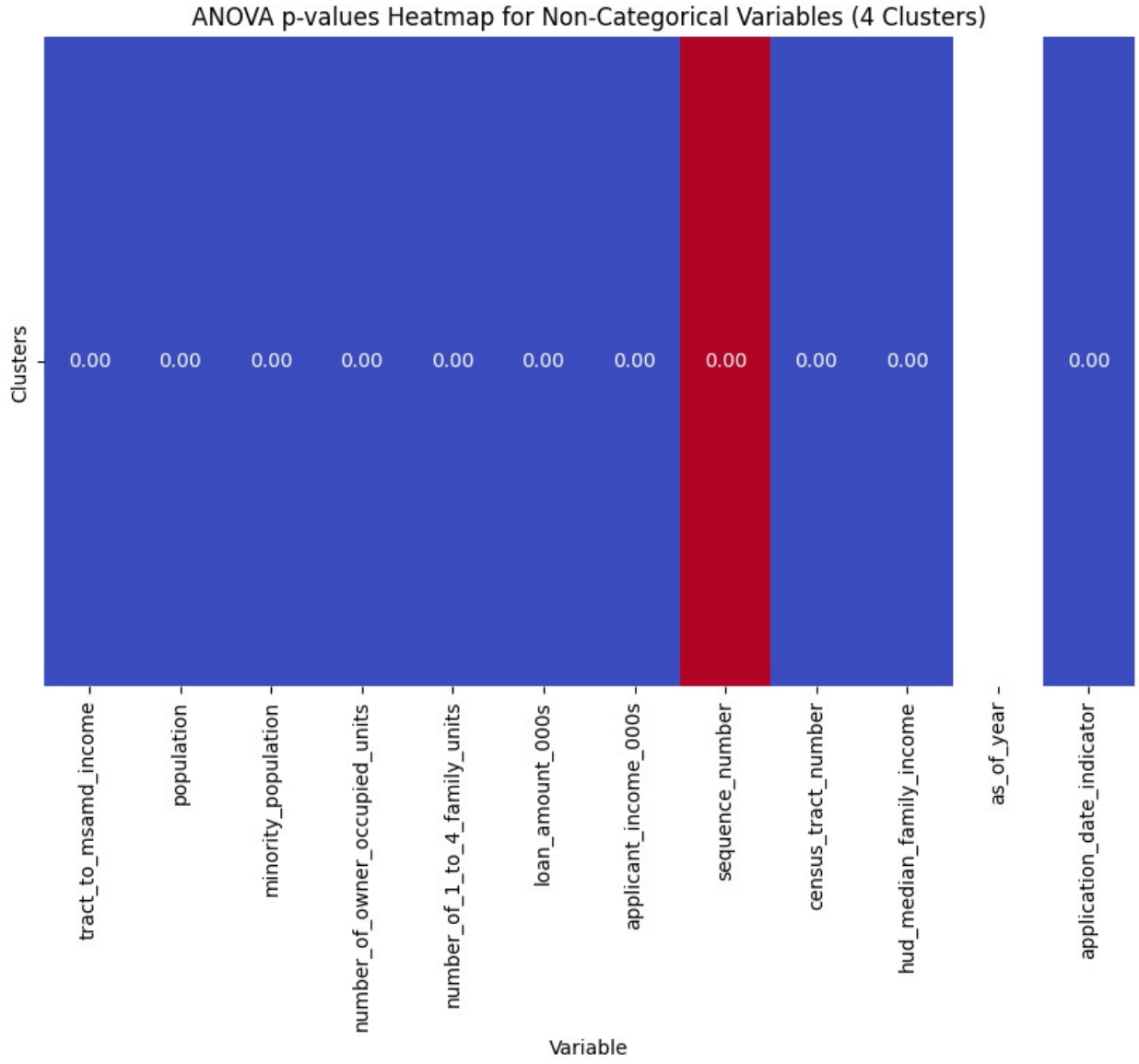
```
print("Number of Variables in Each Cluster:")
print(num_variables_in_clusters)
```

```
Number of Variables in Each Cluster:
0     8935
1    10205
2    17965
3    22895
dtype: int64
time: 15.8 ms (started: 2024-03-16 13:32:31 +00:00)
```

```
# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_4cluster_model

# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
group in cluster_groups])

# Extract p-values for non-categorical variables
non_cat_p_values = [result.pvalue for result in
anova_results_non_cat.values()]

# Plotting heatmap for non-categorical variables
plt.figure(figsize=(10, 6))
sns.heatmap([non_cat_p_values], cmap='coolwarm', annot=True,
fmt='.2f', xticklabels=non_cat_variables, yticklabels=['Clusters'],
cbar=False)
plt.xlabel('Variable')
plt.title('ANOVA p-values Heatmap for Non-Categorical Variables (4
```

```
Clusters)')
plt.show()
```

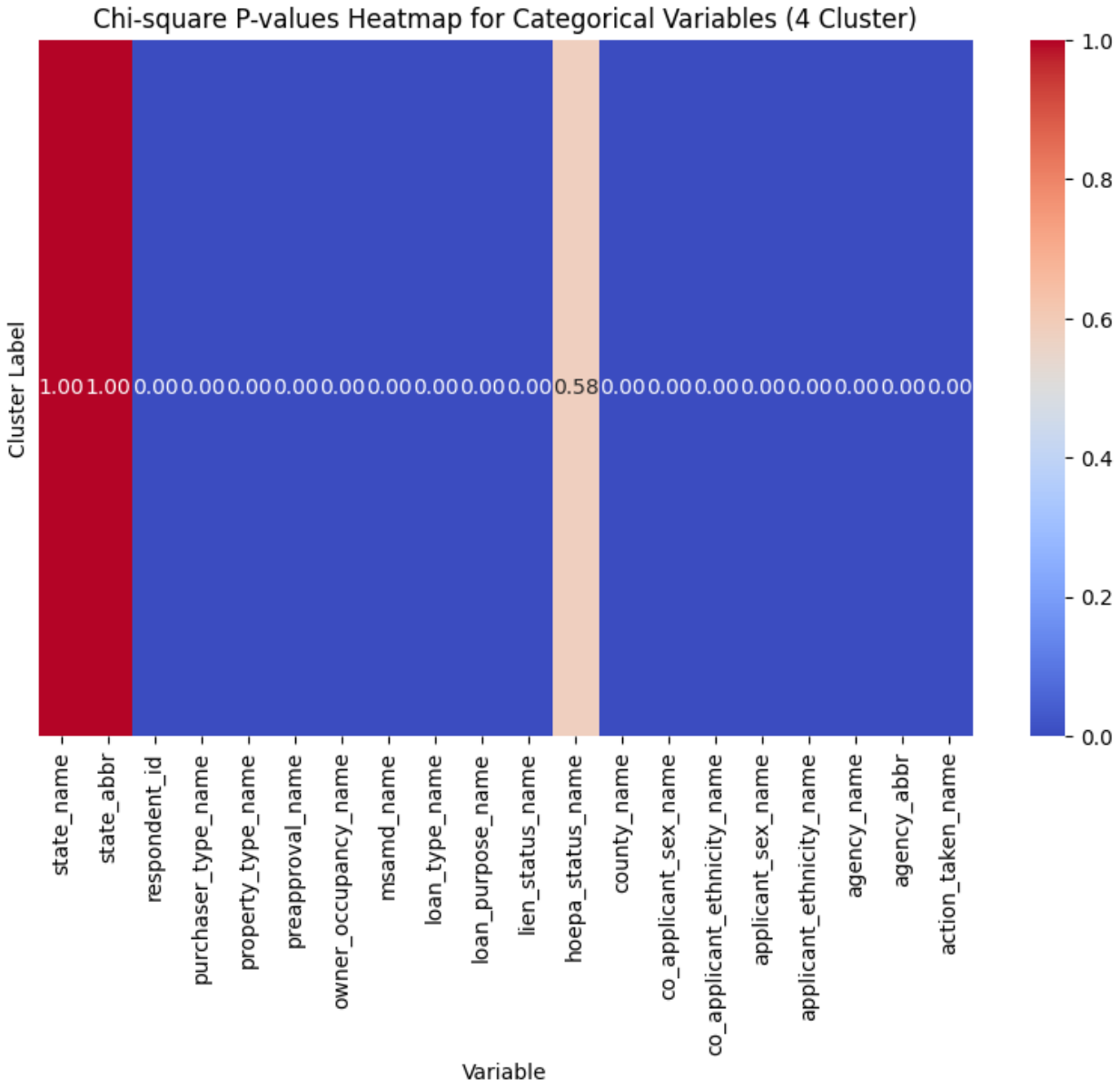
```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning:

Each of the input arrays is constant;the F statistic is not defined or
infinite
```




```
time: 962 ms (started: 2024-03-16 13:32:31 +00:00)
```

```
cat_variables = ['state_name',
 'state_abbr',
```

```
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']


# Initialize dictionary to store chi-square results
chi2_results_cat = {}

# Calculate chi-square for each categorical variable
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_4cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'P-value': p_value}

# Extract p-values
p_values = [[result['P-value'] for result in
chi2_results_cat.values()]]

# Get variable names
variables = list(chi2_results_cat.keys())

# Plotting heatmap for p-values
plt.figure(figsize=(10, 6))
sns.heatmap(p_values, cmap='coolwarm', annot=True, fmt='.2f',
xticklabels=variables, yticklabels=False)
plt.xlabel('Variable')
plt.ylabel('Cluster Label')
plt.title('Chi-square P-values Heatmap for Categorical Variables (4
Cluster)')
plt.show()
```




```
time: 1.43 s (started: 2024-03-16 13:32:32 +00:00)
```

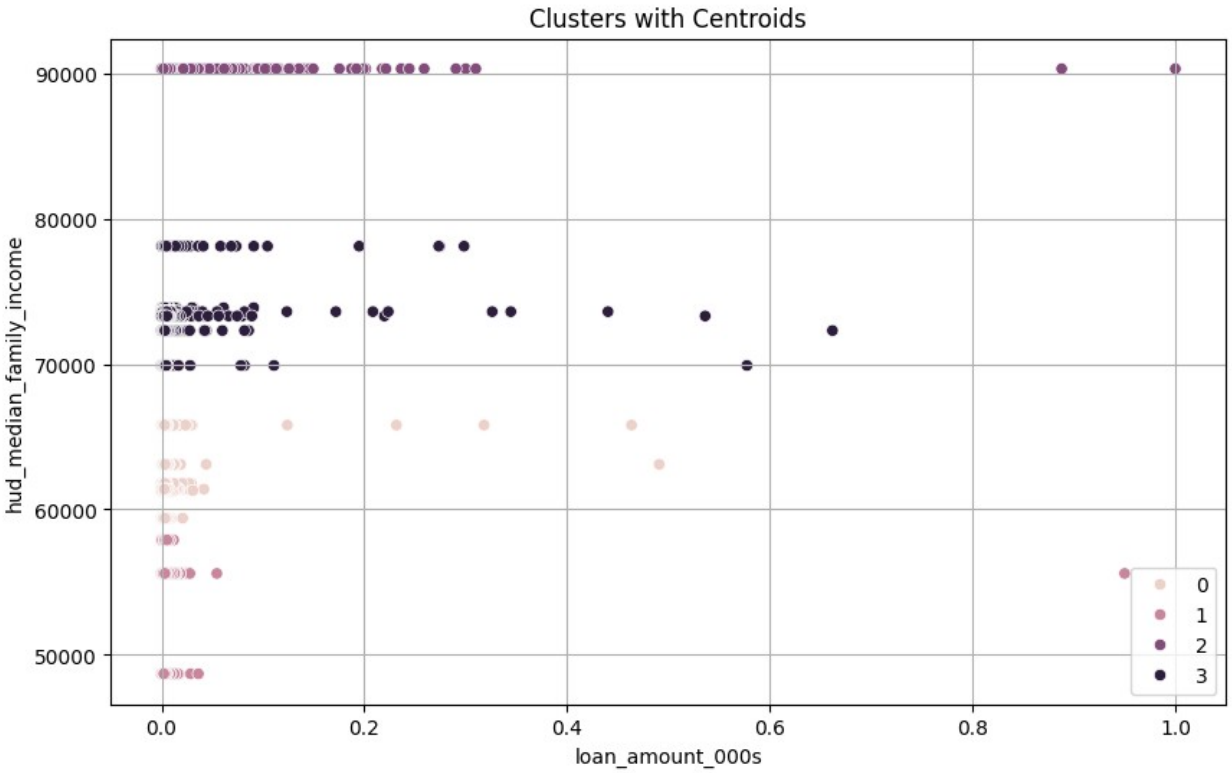
```
import plotly.graph_objs as go
import numpy as np

# Create 3D scatter plot
fig = go.Figure()

# Get unique cluster labels
unique_clusters = np.unique(km_4cluster_model)

# Add traces for each cluster
for cluster_label in unique_clusters:
```

```
    cluster_data = df_ppd_subset[km_4cluster_model == cluster_label]
    fig.add_trace(go.Scatter3d(
        x=cluster_data['loan_amount_000s'],
        y=cluster_data['hud_median_family_income'],
        z=cluster_data['number_of_owner_occupied_units'],
        mode='markers',
        marker=dict(
            size=5,
            color=cluster_label,
            colorscale='Viridis',  # Adjust colorscale if needed
            opacity=0.8
        ),
        name=f'Cluster {cluster_label}'
    ))

# Set layout
fig.update_layout(
    title='Clusters with Centroids (3D)',
    scene=dict(
        xaxis=dict(title='loan_amount_000s'),
        yaxis=dict(title='hud_median_family_income'),
        zaxis=dict(title='number_of_owner_occupied_units'),
    ),
    margin=dict(l=0, r=0, b=0, t=40)
)

# Show interactive 3D scatter plot
fig.show()
```

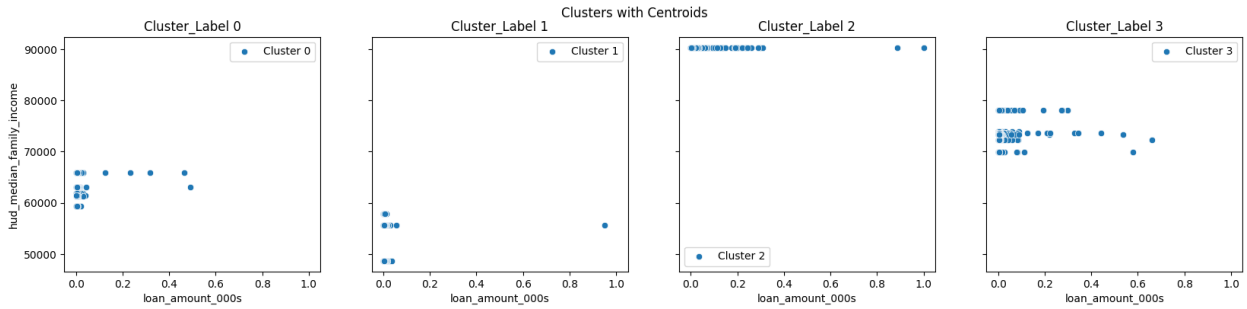
```
time: 304 ms (started: 2024-03-16 13:32:33 +00:00)
```

```
# Assign cluster labels to the DataFrame
df_ppd_subset['Cluster_Label'] = km_4cluster_model

# Plot the scatter plot with clusters and centroids
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_ppd_subset, x='loan_amount_000s',
y='hud_median_family_income', hue='Cluster_Label', legend='full')

# Set plot title and labels
plt.title('Clusters with Centroids')
plt.xlabel('loan_amount_000s')
plt.ylabel('hud_median_family_income')
plt.legend()
plt.grid(True)
%memit
plt.show()
```

```
peak memory: 442.75 MiB, increment: 0.00 MiB
```




```
time: 8.76 s (started: 2024-03-16 13:32:33 +00:00)

# Create subplots for each cluster
fig, axes = plt.subplots(1, 4, figsize=(20, 4), sharex=True,
sharey=True)
fig.suptitle('Clusters with Centroids')

# Iterate through each cluster
for i in range(4):
    # Filter data points belonging to the current cluster
    cluster_data = df_ppd_subset[df_ppd_subset['Cluster_Label'] == i]

    # Scatter plot of data points
    sns.scatterplot(data=cluster_data, x='loan_amount_000s',
y='hud_median_family_income', ax=axes[i], label=f'Cluster {i}')

    # Set title and labels for each subplot
    axes[i].set_title(f'Cluster_Label {i}')
    axes[i].set_xlabel('loan_amount_000s')
    axes[i].set_ylabel('hud_median_family_income')
    axes[i].legend()

plt.show()
%memit
```




```
peak memory: 443.86 MiB, increment: 0.00 MiB
time: 5.03 s (started: 2024-03-16 13:32:42 +00:00)

############           KKKKKKKKKK           ==============
5555555555555555

time: 1.28 ms (started: 2024-03-16 13:32:47 +00:00)
```

```
# Create K-Means Clusters [K=5]
km_5cluster = kmclus(n_clusters=5, init='random', random_state=333)
df_ppd_subset['Cluster_Label'] =
km_5cluster.fit_predict(df_ppd_subset)
km_5cluster_model = df_ppd_subset['Cluster_Label'].values
km_5cluster_model
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning


array([3, 1, 3, ..., 3, 4, 3], dtype=int32)

time: 853 ms (started: 2024-03-16 13:32:47 +00:00)
```

```
# Create subsets for each cluster label
dfk0 = df_ppd_subset[df_ppd_subset['Cluster_Label'] == 0]
dfk1 = df_ppd_subset[df_ppd_subset['Cluster_Label'] == 1]
dfk2 = df_ppd_subset[df_ppd_subset['Cluster_Label'] == 2]
dfk3 = df_ppd_subset[df_ppd_subset['Cluster_Label'] == 3]
dfk4 = df_ppd_subset[df_ppd_subset['Cluster_Label'] == 4]

# Find the number of rows for each subset
num_rows_dfk0 = dfk0.shape[0]
num_rows_dfk1 = dfk1.shape[0]
num_rows_dfk2 = dfk2.shape[0]
num_rows_dfk3 = dfk3.shape[0]
num_rows_dfk4 = dfk4.shape[0]

print("Number of rows in dfk0:", num_rows_dfk0)
```

```
print("Number of rows in dfk1:", num_rows_dfk1)
print("Number of rows in dfk2:", num_rows_dfk2)
print("Number of rows in dfk3:", num_rows_dfk3)
print("Number of rows in dfk4:", num_rows_dfk4)
```

```
Number of rows in dfk0: 8935
Number of rows in dfk1: 10205
Number of rows in dfk2: 17965
Number of rows in dfk3: 19457
Number of rows in dfk4: 3438
time: 35.5 ms (started: 2024-03-16 13:32:48 +00:00)
```

```
# K-Means Clustering Model Evaluation [K=5]
# ----------------------------------------------------

sscore_km_5cluster = sscore(df_ppd_subset, km_5cluster_model)
dbscore_km_5cluster = dbscore(df_ppd_subset, km_5cluster_model);
%memit
print(f"Davies-Bouldin Index for 5 clusters: {dbscore_km_5cluster}")
print(f"Silhouette Score for 5 clusters: {sscore_km_5cluster}")
```

```
peak memory: 452.38 MiB, increment: 0.00 MiB
Davies-Bouldin Index for 5 clusters: 0.2879640274935218
Silhouette Score for 5 clusters: 0.8465997294820591
time: 1min 7s (started: 2024-03-16 13:32:48 +00:00)
```

```
# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_5cluster_model

# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
```

```
group in cluster_groups])

# Print ANOVA results for non-categorical variables
for column, result in anova_results_non_cat.items():
    print(f"Variable: {column}")
    print(f"F-value: {result.statistic}")
    print(f"P-value: {result.pvalue}")
    print()
```

```
Variable: tract_to_msamd_income
F-value: 93.49487585041904
P-value: 2.0648710483951816e-79

Variable: population
F-value: 120.34126670279981
P-value: 1.8612114647932083e-102

Variable: minority_population
F-value: 1543.6424637678217
P-value: 0.0

Variable: number_of_owner_occupied_units
F-value: 184.41512936737612
P-value: 2.2830062373195167e-157

Variable: number_of_1_to_4_family_units
F-value: 1808.5304628875501
P-value: 0.0

Variable: loan_amount_000s
F-value: 276.42057634528965
P-value: 6.68692455301139e-236

Variable: applicant_income_000s
F-value: 466.1257290083396
P-value: 0.0

Variable: sequence_number
F-value: 65.07731024940496
P-value: 5.161290729595577e-55

Variable: census_tract_number
F-value: 7928.803505668927
P-value: 0.0

Variable: hud_median_family_income
F-value: 1143184.9185680407
P-value: 0.0

Variable: as_of_year
F-value: nan
```

```
P-value: nan

Variable: application_date_indicator
F-value: 859.6826684826774
P-value: 0.0

time: 185 ms (started: 2024-03-16 13:33:55 +00:00)

/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning:

Each of the input arrays is constant;the F statistic is not defined or
infinite


from scipy.stats import chi2_contingency

# Extracting categorical variables
cat_variables = ['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Perform Chi-square test for each categorical variable
chi2_results_cat = {}
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_5cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'Chi-square': chi2, 'P-value':
p_value}

# Print Chi-square test results for categorical variables
for column, result in chi2_results_cat.items():
```

```
    print(f"Variable: {column}")
    print(f"Chi-square: {result['Chi-square']}")
    print(f"P-value: {result['P-value']}")
    print()
```

```
Variable: state_name
Chi-square: 0.0
P-value: 1.0

Variable: state_abbr
Chi-square: 0.0
P-value: 1.0

Variable: respondent_id
Chi-square: 37667.00324602582
P-value: 0.0

Variable: purchaser_type_name
Chi-square: 3421.7380065506586
P-value: 0.0

Variable: property_type_name
Chi-square: 1313.264544866733
P-value: 3.191747745479637e-278

Variable: preapproval_name
Chi-square: 494.6907302308296
P-value: 9.689713005617645e-102

Variable: owner_occupancy_name
Chi-square: 326.87732492857003
P-value: 7.751748869377117e-66

Variable: msamd_name
Chi-square: 187202.44702894494
P-value: 0.0

Variable: loan_type_name
Chi-square: 4193.389893736484
P-value: 0.0

Variable: loan_purpose_name
Chi-square: 209.97632564398754
P-value: 5.034516395326935e-41

Variable: lien_status_name
Chi-square: 3332.3871806545253
P-value: 0.0

Variable: hoepa_status_name
Chi-square: 2.423712104723065
```

```
P-value: 0.6583463972387893

Variable: county_name
Chi-square: 239169.27522539254
P-value: 0.0

Variable: co_applicant_sex_name
Chi-square: 2320.8209893840603
P-value: 0.0

Variable: co_applicant_ethnicity_name
Chi-square: 2472.609477291185
P-value: 0.0

Variable: applicant_sex_name
Chi-square: 988.6046170530095
P-value: 5.277439994457898e-204

Variable: applicant_ethnicity_name
Chi-square: 1517.7897607925352
P-value: 0.0

Variable: agency_name
Chi-square: 4859.037635138346
P-value: 0.0

Variable: agency_abbr
Chi-square: 4859.037635138346
P-value: 0.0

Variable: action_taken_name
Chi-square: 8254.23051445065
P-value: 0.0

time: 421 ms (started: 2024-03-16 13:33:55 +00:00)

cluster_centers = km_5cluster.cluster_centers_

# Convert centroids to DataFrame for better visualization
centroids_df = pd.DataFrame(cluster_centers,
columns=df_ppd_subset.columns)
%memit
print("Centroids of Clusters:")
centroids_df

peak memory: 452.40 MiB, increment: 0.00 MiB
Centroids of Clusters:

{"type":"dataframe","variable_name":"centroids_df"}

time: 376 ms (started: 2024-03-16 13:33:56 +00:00)
```

```
# Calculate cluster-wise descriptive statistics
cluster_stats = df_ppd_subset.groupby('Cluster_Label').describe()

# Print the descriptive statistics for each cluster
print("Cluster-wise Descriptive Statistics:")
cluster_stats
```

```
Cluster-wise Descriptive Statistics:
```

{"type":"dataframe","variable_name":"cluster_stats"}

```
time: 922 ms (started: 2024-03-16 13:33:56 +00:00)
```

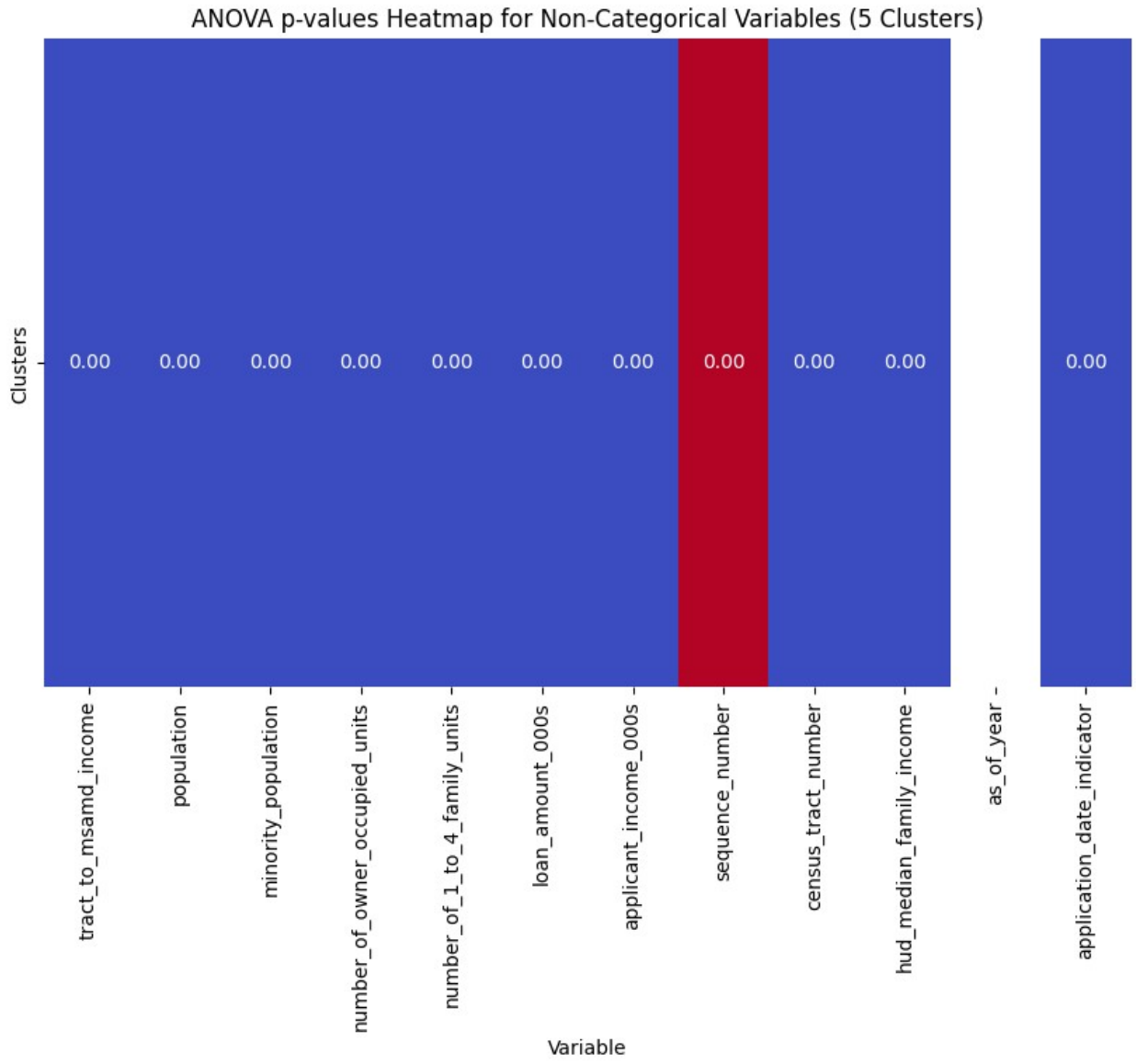
```
# Grouping the data by cluster labels
cluster_groups = df_with_clusters.groupby(km_5cluster_model)

# Counting the number of variables in each cluster
num_variables_in_clusters = cluster_groups.size()

# Displaying the results
print("Number of Variables in Each Cluster:")
print(num_variables_in_clusters)
```

```
Number of Variables in Each Cluster:
0     8935
1    10205
2    17965
3    19457
4     3438
dtype: int64
time: 20.6 ms (started: 2024-03-16 13:33:57 +00:00)
```

```
# Joining cluster labels with the original dataset
df_with_clusters = df_ppd_subset.copy()
df_with_clusters['Cluster_Label'] = km_5cluster_model

# Extracting non-categorical variables
non_cat_variables = ['tract_to_msamd_income',
 'population',
 'minority_population',
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Grouping variables by cluster label
```

```
cluster_groups = df_with_clusters.groupby('Cluster_Label')

# Perform ANOVA for each non-categorical variable
anova_results_non_cat = {}
for column in non_cat_variables:
    anova_results_non_cat[column] = f_oneway(*[group[column] for name,
group in cluster_groups])

# Extract p-values for non-categorical variables
non_cat_p_values = [result.pvalue for result in
anova_results_non_cat.values()]

# Plotting heatmap for non-categorical variables
plt.figure(figsize=(10, 6))
sns.heatmap([non_cat_p_values], cmap='coolwarm', annot=True,
fmt='.2f', xticklabels=non_cat_variables, yticklabels=['Clusters'],
cbar=False)
plt.xlabel('Variable')
plt.title('ANOVA p-values Heatmap for Non-Categorical Variables (5
Clusters)')
plt.show()
```

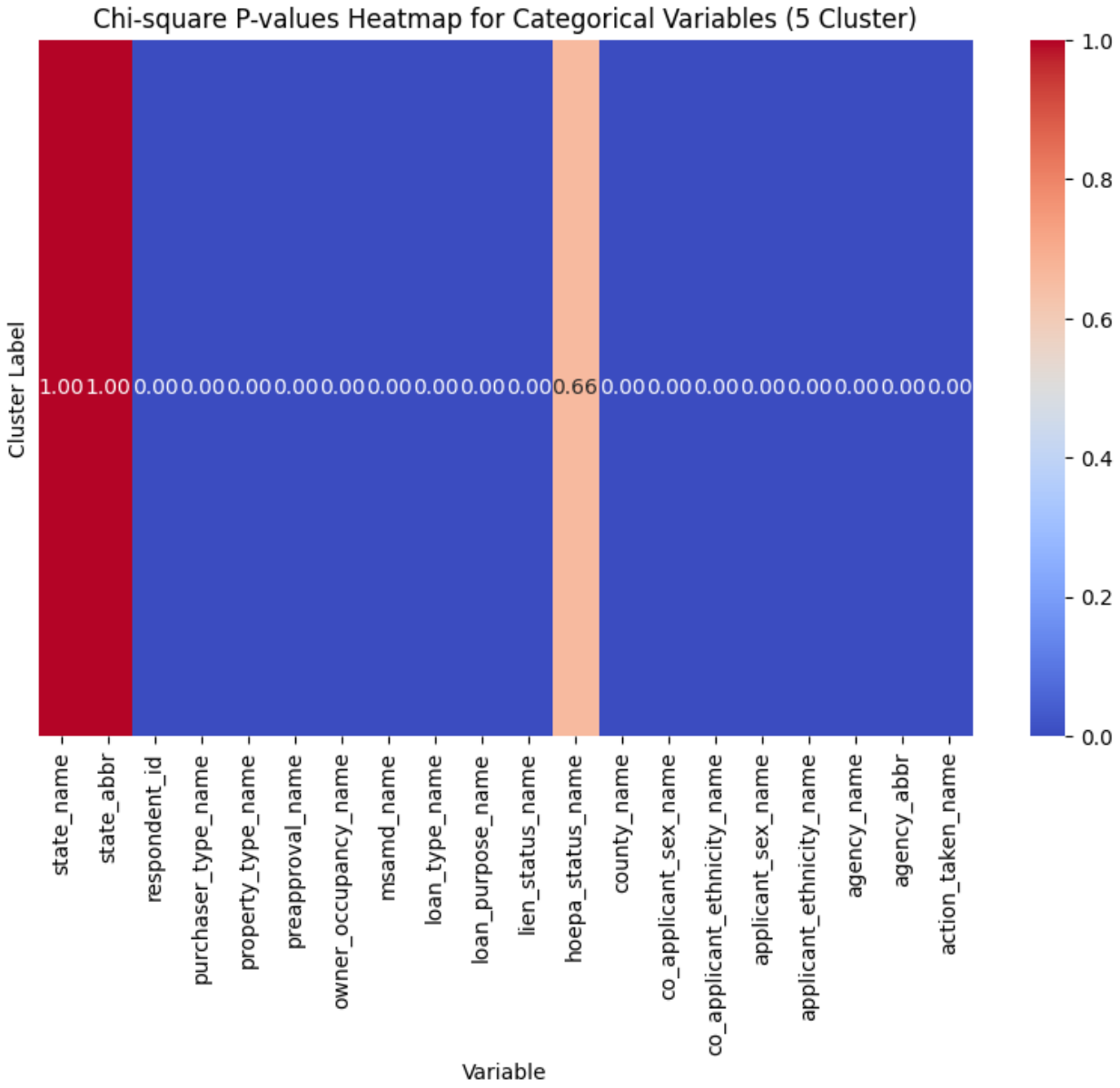
```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:4167:
ConstantInputWarning:

Each of the input arrays is constant;the F statistic is not defined or
infinite
```




```
time: 521 ms (started: 2024-03-16 13:33:57 +00:00)
```

```
cat_variables = ['state_name',
 'state_abbr',
 'respondent_id',
 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
```

```
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Initialize dictionary to store chi-square results
chi2_results_cat = {}

# Calculate chi-square for each categorical variable
for column in cat_variables:
    contingency_table = pd.crosstab(df_with_clusters[column],
km_5cluster_model)
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    chi2_results_cat[column] = {'P-value': p_value}

# Extract p-values
p_values = [[result['P-value'] for result in
chi2_results_cat.values()]]

# Get variable names
variables = list(chi2_results_cat.keys())

# Plotting heatmap for p-values
plt.figure(figsize=(10, 6))
sns.heatmap(p_values, cmap='coolwarm', annot=True, fmt='.2f',
xticklabels=variables, yticklabels=False)
plt.xlabel('Variable')
plt.ylabel('Cluster Label')
plt.title('Chi-square P-values Heatmap for Categorical Variables (5
Cluster)')
plt.show()
```




```
time: 997 ms (started: 2024-03-16 13:33:58 +00:00)
```

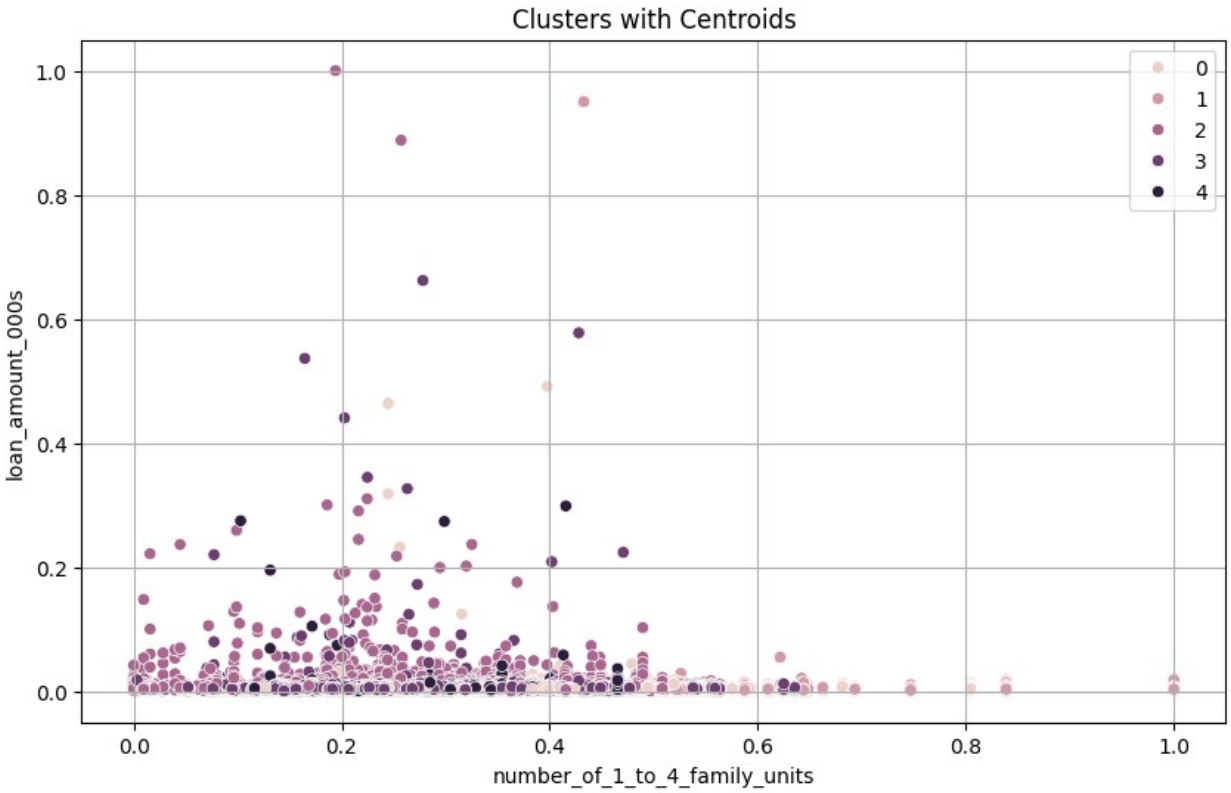
```
import plotly.graph_objs as go
import numpy as np

# Create 3D scatter plot
fig = go.Figure()

# Get unique cluster labels
unique_clusters = np.unique(km_5cluster_model)

# Add traces for each cluster
for cluster_label in unique_clusters:
```

```
    cluster_data = df_ppd_subset[km_5cluster_model == cluster_label]
    fig.add_trace(go.Scatter3d(
        x=cluster_data['number_of_owner_occupied_units'],
        y=cluster_data['number_of_1_to_4_family_units'],
        z=cluster_data['loan_amount_000s'],
        mode='markers',
        marker=dict(
            size=5,
            color=cluster_label,
            colorscale='Viridis',  # Adjust colorscale if needed
            opacity=0.8
        ),
        name=f'Cluster {cluster_label}'
    ))

# Set layout
fig.update_layout(
    title='Clusters with Centroids (3D)',
    scene=dict(
        xaxis=dict(title='number_of_owner_occupied_units'),
        yaxis=dict(title='number_of_1_to_4_family_units'),
        zaxis=dict(title='loan_amount_000s'),
    ),
    margin=dict(l=0, r=0, b=0, t=40)
)

# Show interactive 3D scatter plot
fig.show()
```

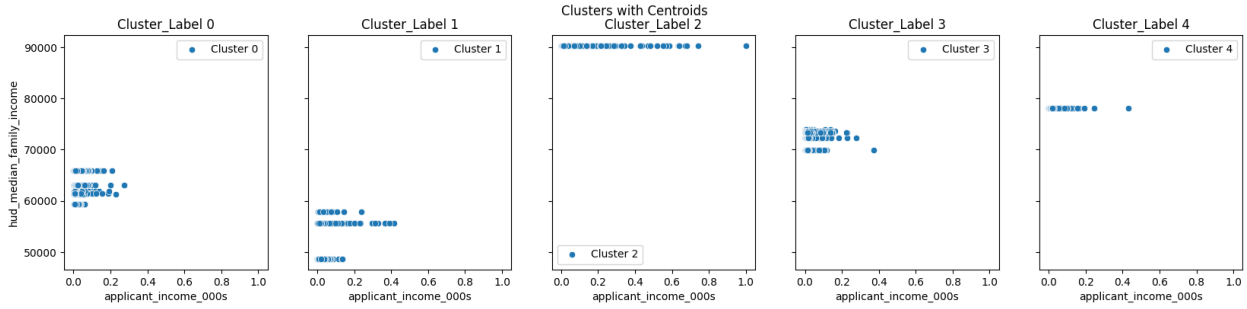
```
time: 397 ms (started: 2024-03-16 13:33:59 +00:00)
```

```
# Assign cluster labels to the DataFrame
df_ppd_subset['Cluster_Label'] = km_5cluster_model

# Plot the scatter plot with clusters and centroids
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_ppd_subset, x='number_of_1_to_4_family_units',
y='loan_amount_000s', hue='Cluster_Label', legend='full')

# Set plot title and labels
plt.title('Clusters with Centroids')
plt.xlabel('number_of_1_to_4_family_units')
plt.ylabel('loan_amount_000s')
plt.legend()
plt.grid(True)
%memit
plt.show()
```

```
peak memory: 441.18 MiB, increment: -0.98 MiB
```




```
time: 8.58 s (started: 2024-03-16 13:33:59 +00:00)

# Create subplots for each cluster
fig, axes = plt.subplots(1, 5, figsize=(20, 4), sharex=True,
sharey=True)
fig.suptitle('Clusters with Centroids')

# Iterate through each cluster
for i in range(5):
    # Filter data points belonging to the current cluster
    cluster_data = df_ppd_subset[df_ppd_subset['Cluster_Label'] == i]

    # Scatter plot of data points
    sns.scatterplot(data=cluster_data, x='applicant_income_000s',
y='hud_median_family_income', ax=axes[i], label=f'Cluster {i}')

    # Set title and labels for each subplot
    axes[i].set_title(f'Cluster_Label {i}')
    axes[i].set_xlabel('applicant_income_000s')
    axes[i].set_ylabel('hud_median_family_income')
    axes[i].legend()

plt.show()
%memit
```

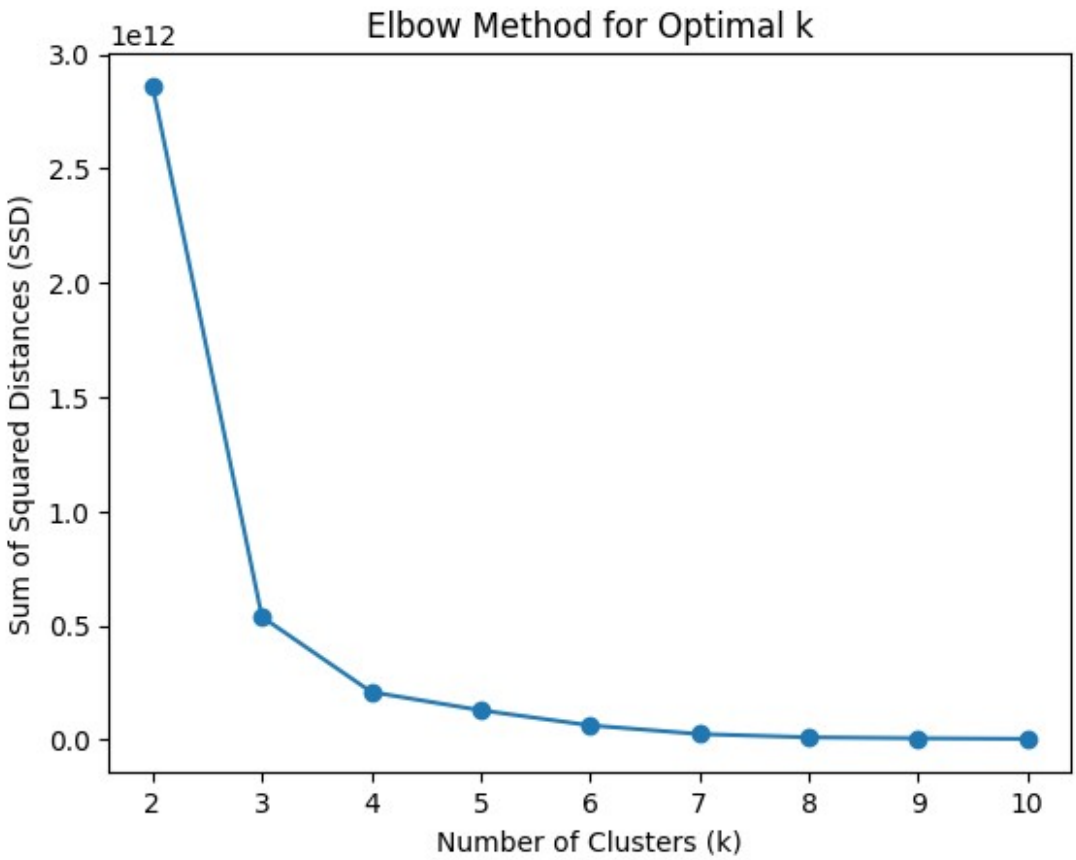



```
peak memory: 444.18 MiB, increment: 0.00 MiB
time: 3.2 s (started: 2024-03-16 13:34:08 +00:00)
```

```
# Example code for Elbow Method
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Assuming 'data' is your feature matrix
ssd = []
k_values = range(2, 11)

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

# Plotting the Elbow curve
plt.plot(k_values, ssd, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Sum of Squared Distances (SSD)')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning
```




```
time: 12.3 s (started: 2024-03-16 13:34:11 +00:00)
```

```
import numpy as np

def cramers_v(confusion_matrix):
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum()
    if n == 0:
        return np.nan
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    phi2corr = max(0, phi2 - ((k - 1) * (r - 1)) / (n - 1))
    rcorr = r - ((r - 1)**2) / (n - 1)
    kcorr = k - ((k - 1)**2) / (n - 1)
    return np.sqrt(phi2corr / min((kcorr - 1), (rcorr - 1)))

# Calculate Cramér's V for each significant categorical variable
cramers_v_results = {}
for column in chi2_results_cat.keys():
    contingency_table = pd.crosstab(df_with_clusters[column],
km_5cluster_model)
    cramers_v_value = cramers_v(contingency_table.values)
    cramers_v_results[column] = cramers_v_value
```

```python
# Print Cramér's V results for significant categorical variables
for column, result in cramers_v_results.items():
    print(f"Variable: {column}")
    print(f"Cramér's V: {result}")
    print()
```

```
<ipython-input-90-82ce08ede275>:13: RuntimeWarning:

invalid value encountered in scalar divide

<ipython-input-90-82ce08ede275>:13: RuntimeWarning:

invalid value encountered in scalar divide


Variable: state_name
Cramér's V: nan

Variable: state_abbr
Cramér's V: nan

Variable: respondent_id
Cramér's V: 0.3835216054272289

Variable: purchaser_type_name
Cramér's V: 0.11877785885809031

Variable: property_type_name
Cramér's V: 0.1042955704764287

Variable: preapproval_name
Cramér's V: 0.06368587106038266

Variable: owner_occupancy_name
Cramér's V: 0.051549961312968254

Variable: msamd_name
Cramér's V: 0.8830886230971905

Variable: loan_type_name
Cramér's V: 0.15241753284165638

Variable: loan_purpose_name
Cramér's V: 0.04102671133794212

Variable: lien_status_name
Cramér's V: 0.13582163571449382

Variable: hoepa_status_name
Cramér's V: 0.0
```

```
Variable: county_name
Cramér's V: 0.997983818557962

Variable: co_applicant_sex_name
Cramér's V: 0.0980003059143487

Variable: co_applicant_ethnicity_name
Cramér's V: 0.10117585759080602

Variable: applicant_sex_name
Cramér's V: 0.07366037871075352

Variable: applicant_ethnicity_name
Cramér's V: 0.09146538067872924

Variable: agency_name
Cramér's V: 0.14199999899019988

Variable: agency_abbr
Cramér's V: 0.14199999899019988

Variable: action_taken_name
Cramér's V: 0.18514384620273197

time: 503 ms (started: 2024-03-16 13:34:23 +00:00)
```

```
# Aggregate statistics across clusters
cluster_agg_stats =
df_ppd_subset.groupby('Cluster_Label').agg(['mean', 'std', 'min',
'max'])

# Print the aggregated statistics
print(cluster_agg_stats)
```

```
              state_name                state_abbr
respondent_id  \
                    mean  std min max       mean  std min max
mean
Cluster_Label

0                    0.0  0.0   0   0        0.0  0.0   0   0
289.874314
1                    0.0  0.0   0   0        0.0  0.0   0   0
329.375796
2                    0.0  0.0   0   0        0.0  0.0   0   0
278.063290
3                    0.0  0.0   0   0        0.0  0.0   0   0
318.292440
4                    0.0  0.0   0   0        0.0  0.0   0   0
328.508726
```

```
                              ... hud_median_family_income
\
                        std  ...                        min            max

Cluster_Label                ...

0               163.739784  ...                    59400.0  65800.000000

1               168.389260  ...                    48700.0  57900.000000

2               159.300937  ...                    90300.0  90300.000000

3               162.183633  ...                    69900.0  73869.411136

4               163.740242  ...                    78100.0  78100.000000


               as_of_year
application_date_indicator  \
                      mean  std     min     max
mean
Cluster_Label

0                   2016.0  0.0  2016.0  2016.0
0.002238
1                   2016.0  0.0  2016.0  2016.0
0.142087
2                   2016.0  0.0  2016.0  2016.0
0.002004
3                   2016.0  0.0  2016.0  2016.0
0.002056
4                   2016.0  0.0  2016.0  2016.0
0.001745


                      std  min  max
Cluster_Label
0               0.066875  0.0  2.0
1               0.513821  0.0  2.0
2               0.063277  0.0  2.0
3               0.064091  0.0  2.0
4               0.059062  0.0  2.0

[5 rows x 128 columns]
time: 213 ms (started: 2024-03-16 13:34:24 +00:00)
```

```
# Select relevant columns
relevant_columns = ['tract_to_msamd_income',
 'population',
 'minority_population',
```

```
 'number_of_owner_occupied_units',
 'number_of_1_to_4_family_units',
 'loan_amount_000s',
 'applicant_income_000s',
 'sequence_number',
 'census_tract_number',
 'hud_median_family_income',
 'as_of_year',
 'application_date_indicator']

# Aggregate statistics across clusters for relevant columns
#cluster_agg_stats_relevant = df_ppd_subset.groupby('Cluster_Label')
[relevant_columns].agg(['mean', 'std', 'min', 'max'])
cluster_agg_stats_relevant = df_ppd_subset.groupby('Cluster_Label')
[relevant_columns].agg(['mean'])
# Print the aggregated statistics for relevant columns
print(cluster_agg_stats_relevant)
```

```
              tract_to_msamd_income population minority_population  \
                               mean       mean                 mean
Cluster_Label
0                          0.396764   0.426652             0.204515
1                          0.391195   0.394937             0.195526
2                          0.389233   0.403022             0.300940
3                          0.372497   0.391808             0.197639
4                          0.383081   0.389814             0.186917

              number_of_owner_occupied_units
number_of_1_to_4_family_units  \
                                        mean
mean
Cluster_Label

0                                    0.490269
0.328579
1                                    0.490023
0.396458
2                                    0.453287
0.274998
3                                    0.446166
0.300941
4                                    0.477716
0.322025

              loan_amount_000s applicant_income_000s
sequence_number  \
                          mean                  mean             mean

Cluster_Label
```

```
0                       0.003932              0.015264        0.064634

1                       0.003738              0.015908        0.072378

2                       0.007448              0.023106        0.051118

3                       0.004705              0.016189        0.064932

4                       0.005278              0.017556        0.072308


              census_tract_number hud_median_family_income as_of_year
\
                             mean                      mean       mean

Cluster_Label

0                         0.285113             63375.153889     2016.0

1                         0.586288             54609.681529     2016.0

2                         0.031471             90300.000000     2016.0

3                         0.062902             72871.900507     2016.0

4                         0.128075             78100.000000     2016.0


              application_date_indicator
                                    mean
Cluster_Label
0                                 0.002238
1                                 0.142087
2                                 0.002004
3                                 0.002056
4                                 0.001745
time: 47.5 ms (started: 2024-03-16 13:34:24 +00:00)
```

```
# Define the non-categorical variables
non_cat_variables = ['tract_to_msamd_income', 'population',
'minority_population',
                     'number_of_owner_occupied_units',
'number_of_1_to_4_family_units',
                     'loan_amount_000s', 'applicant_income_000s',
'sequence_number',
                     'census_tract_number',
'hud_median_family_income', 'as_of_year',
                     'application_date_indicator']

# Create a dictionary to store the average values for each non-
categorical variable in each cluster
```

```
cluster_profiles_non_cat = {}

# Calculate the average values for each non-categorical variable in
each cluster
for variable in non_cat_variables:
    cluster_profiles_non_cat[variable] =
df_ppd_subset.groupby('Cluster_Label')[variable].mean()

# Print the cluster profiling for non-categorical variables
for variable, averages in cluster_profiles_non_cat.items():
    print(f"Variable: {variable}")
    for cluster, avg_value in averages.items():
        print(f"Cluster {cluster}: Average {variable} -
{avg_value:.2f}")
    print()
```

```
Variable: tract_to_msamd_income
Cluster 0: Average tract_to_msamd_income - 0.40
Cluster 1: Average tract_to_msamd_income - 0.39
Cluster 2: Average tract_to_msamd_income - 0.39
Cluster 3: Average tract_to_msamd_income - 0.37
Cluster 4: Average tract_to_msamd_income - 0.38

Variable: population
Cluster 0: Average population - 0.43
Cluster 1: Average population - 0.39
Cluster 2: Average population - 0.40
Cluster 3: Average population - 0.39
Cluster 4: Average population - 0.39

Variable: minority_population
Cluster 0: Average minority_population - 0.20
Cluster 1: Average minority_population - 0.20
Cluster 2: Average minority_population - 0.30
Cluster 3: Average minority_population - 0.20
Cluster 4: Average minority_population - 0.19

Variable: number_of_owner_occupied_units
Cluster 0: Average number_of_owner_occupied_units - 0.49
Cluster 1: Average number_of_owner_occupied_units - 0.49
Cluster 2: Average number_of_owner_occupied_units - 0.45
Cluster 3: Average number_of_owner_occupied_units - 0.45
Cluster 4: Average number_of_owner_occupied_units - 0.48

Variable: number_of_1_to_4_family_units
Cluster 0: Average number_of_1_to_4_family_units - 0.33
Cluster 1: Average number_of_1_to_4_family_units - 0.40
Cluster 2: Average number_of_1_to_4_family_units - 0.27
Cluster 3: Average number_of_1_to_4_family_units - 0.30
Cluster 4: Average number_of_1_to_4_family_units - 0.32
```

```
Variable: loan_amount_000s
Cluster 0: Average loan_amount_000s - 0.00
Cluster 1: Average loan_amount_000s - 0.00
Cluster 2: Average loan_amount_000s - 0.01
Cluster 3: Average loan_amount_000s - 0.00
Cluster 4: Average loan_amount_000s - 0.01

Variable: applicant_income_000s
Cluster 0: Average applicant_income_000s - 0.02
Cluster 1: Average applicant_income_000s - 0.02
Cluster 2: Average applicant_income_000s - 0.02
Cluster 3: Average applicant_income_000s - 0.02
Cluster 4: Average applicant_income_000s - 0.02

Variable: sequence_number
Cluster 0: Average sequence_number - 0.06
Cluster 1: Average sequence_number - 0.07
Cluster 2: Average sequence_number - 0.05
Cluster 3: Average sequence_number - 0.06
Cluster 4: Average sequence_number - 0.07

Variable: census_tract_number
Cluster 0: Average census_tract_number - 0.29
Cluster 1: Average census_tract_number - 0.59
Cluster 2: Average census_tract_number - 0.03
Cluster 3: Average census_tract_number - 0.06
Cluster 4: Average census_tract_number - 0.13

Variable: hud_median_family_income
Cluster 0: Average hud_median_family_income - 63375.15
Cluster 1: Average hud_median_family_income - 54609.68
Cluster 2: Average hud_median_family_income - 90300.00
Cluster 3: Average hud_median_family_income - 72871.90
Cluster 4: Average hud_median_family_income - 78100.00

Variable: as_of_year
Cluster 0: Average as_of_year - 2016.00
Cluster 1: Average as_of_year - 2016.00
Cluster 2: Average as_of_year - 2016.00
Cluster 3: Average as_of_year - 2016.00
Cluster 4: Average as_of_year - 2016.00

Variable: application_date_indicator
Cluster 0: Average application_date_indicator - 0.00
Cluster 1: Average application_date_indicator - 0.14
Cluster 2: Average application_date_indicator - 0.00
Cluster 3: Average application_date_indicator - 0.00
Cluster 4: Average application_date_indicator - 0.00
```

```
time: 61.7 ms (started: 2024-03-16 13:34:24 +00:00)
```

```
# Select relevant columns
relevant_columns = [ 'purchaser_type_name',
 'property_type_name',
 'preapproval_name',
 'owner_occupancy_name',
 'msamd_name',
 'loan_type_name',
 'loan_purpose_name',
 'lien_status_name',
 'hoepa_status_name',
 'county_name',
 'co_applicant_sex_name',
 'co_applicant_ethnicity_name',
 'applicant_sex_name',
 'applicant_ethnicity_name',
 'agency_name',
 'agency_abbr',
 'action_taken_name']

# Create a dictionary to store frequencies for each column and cluster
cluster_freq_stats_relevant = {}

# Iterate over relevant columns
for column in relevant_columns:
    # Group the data by 'Cluster_Label' and get value counts for each
cluster
    cluster_freq_stats_relevant[column] =
df_ppd_subset.groupby('Cluster_Label')[column].value_counts()

# Print the frequency statistics for relevant columns
for column, value_counts in cluster_freq_stats_relevant.items():
    print(f"Variable: {column}")
    print(value_counts)
    print()
```

```
Variable: purchaser_type_name
Cluster_Label  purchaser_type_name
0              7                     2022
               2                     1945
               5                     1709
               4                     1314
               6                      720
               8                      642
               1                      481
               0                       53
               9                       49
1              7                     3581
```

```
                5                          1989
                2                          1869
                4                          1210
                6                           594
                1                           479
                8                           305
                0                           121
                9                            57
2               2                          6070
                7                          4849
                4                          2733
                5                          1401
                6                          1209
                1                           956
                8                           559
                0                           100
                9                            88
3               7                          4851
                2                          3913
                5                          3777
                4                          2556
                6                          2077
                1                          1214
                8                           687
                0                           231
                9                           150
                3                             1
4               5                           905
                7                           809
                2                           606
                4                           476
                6                           265
                1                           188
                8                           102
                0                            48
                9                            39
Name: purchaser_type_name, dtype: int64

Variable: property_type_name
Cluster_Label  property_type_name
0              2                          8491
               0                           418
               1                            26
1              2                          9345
               0                           844
               1                            16
2              2                         17574
               1                           203
               0                           188
```

```
3              2                       18930
               0                         478
               1                          49
4              2                        3290
               0                         133
               1                          15
Name: property_type_name, dtype: int64

Variable: preapproval_name
Cluster_Label  preapproval_name
0              0                       6938
               1                       1656
               2                        341
1              0                       8073
               1                       1831
               2                        301
2              0                      14733
               1                       2368
               2                        864
3              0                      15505
               1                       3307
               2                        645
4              0                       2583
               1                        536
               2                        319
Name: preapproval_name, dtype: int64

Variable: owner_occupancy_name
Cluster_Label  owner_occupancy_name
0              2                        8143
               1                         769
               0                          23
1              2                        8833
               1                        1332
               0                          40
2              2                       16186
               1                        1577
               0                         202
3              2                       17632
               1                        1767
               0                          58
4              2                        3146
               1                         279
               0                          13
Name: owner_occupancy_name, dtype: int64

Variable: msamd_name
Cluster_Label  msamd_name
0              2              3779
               5              1659
```

```
                4                1364
                9                1353
                12                588
                3                 192
1               8                8259
                13               1585
                11                361
2               8               17965
3               7                8773
                10               4609
                6                4446
                0                1507
                8                 122
4               1                3438
Name: msamd_name, dtype: int64

Variable: loan_type_name
Cluster_Label  loan_type_name
0              0                   6083
               1                   1475
               3                   1294
               2                     83
1              0                   6431
               3                   1854
               1                   1611
               2                    309
2              0                  15688
               1                   1299
               3                    942
               2                     36
3              0                  12643
               3                   3810
               1                   2840
               2                    164
4              0                   2072
               3                   1043
               1                    272
               2                     51
Name: loan_type_name, dtype: int64

Variable: loan_purpose_name
Cluster_Label  loan_purpose_name
0              1                    4654
               2                    3786
               0                     495
1              1                    4979
               2                    4720
               0                     506
2              2                    9075
```

```
                1                       7843
                0                       1047
3               2                       9404
                1                       9014
                0                       1039
4               1                       1617
                2                       1591
                0                        230
Name: loan_purpose_name, dtype: int64

Variable: lien_status_name
Cluster_Label  lien_status_name
0               2                       8516
                3                        283
                1                        126
                0                         10
1               2                       9123
                0                        725
                3                        207
                1                        150
2               2                      17406
                3                        390
                1                        151
                0                         18
3               2                      18703
                3                        517
                1                        217
                0                         20
4               2                       3298
                3                         74
                1                         63
                0                          3
Name: lien_status_name, dtype: int64

Variable: hoepa_status_name
Cluster_Label  hoepa_status_name
0               1                       8935
1               1                      10205
2               1                      17963
                0                          2
3               1                      19455
                0                          2
4               1                       3438
Name: hoepa_status_name, dtype: int64

Variable: county_name
Cluster_Label  county_name
0               2                2743
                28               1659
                7                1364
```

```
                31               1248
                10               1036
                3                 403
                1                 192
                8                 185
                32                 77
                25                 28
1               38               1585
                14               1453
                22                888
                4                 849
                20                836
                13                774
                12                725
                18                664
                15                416
                35                340
                23                316
                37                309
                24                279
                19                213
                27                196
                0                 127
                21                 84
                34                 58
                9                  51
                6                  21
                11                 21
2               16              12914
                30               5051
3               5                8609
                26               4611
                33               4447
                36               1510
                29                164
                16                 93
                28                  4
                30                  4
                31                  4
                17                  3
                20                  2
                27                  2
                0                   1
                2                   1
                4                   1
                35                  1
4               17               3438
Name: county_name, dtype: int64
```

```
Variable: co_applicant_sex_name
Cluster_Label  co_applicant_sex_name
0              3                       3914
               0                       3699
               2                        776
               1                        540
               4                          6
1              3                       4181
               0                       3863
               1                        891
               2                        829
               4                        441
2              3                       8609
               0                       5923
               1                       1835
               2                       1576
               4                         22
3              3                       8848
               0                       7510
               2                       1657
               1                       1421
               4                         21
4              0                       1555
               3                       1435
               2                        292
               1                        149
               4                          7
Name: co_applicant_sex_name, dtype: int64

Variable: co_applicant_ethnicity_name
Cluster_Label  co_applicant_ethnicity_name
0              2                            3914
               3                            3855
               1                             802
               0                             355
               4                               9
1              2                            4181
               3                            3903
               1                            1301
               4                             441
               0                             379
2              2                            8609
               3                            6412
               1                            2635
               0                             287
               4                              22
3              2                            8848
               3                            7920
               1                            2277
```

```
                0                                   387
                4                                    25
4               3                                  1659
                2                                  1435
                1                                   270
                0                                    65
                4                                     9
Name: co_applicant_ethnicity_name, dtype: int64

Variable: applicant_sex_name
Cluster_Label  applicant_sex_name
0              2                          5914
               0                          2047
               1                           814
               3                           160
1              2                          6061
               0                          2260
               1                          1347
               3                           537
2              2                         10547
               0                          4019
               1                          3070
               3                           329
3              2                         12117
               0                          4622
               1                          2265
               3                           453
4              2                          2431
               0                           745
               1                           227
               3                            35
Name: applicant_sex_name, dtype: int64

Variable: applicant_ethnicity_name
Cluster_Label  applicant_ethnicity_name
0              2                               6882
               1                               1160
               0                                728
               3                                165
1              2                               6916
               1                               1974
               0                                774
               3                                541
2              2                              12724
               1                               4324
               0                                584
               3                                333
3              2                              14608
               1                               3615
```

```
                0                                777
                3                                457
4               2                               2884
                1                                385
                0                                133
                3                                 36
Name: applicant_ethnicity_name, dtype: int64

Variable: agency_name
Cluster_Label  agency_name
0              1                4077
               0                2225
               2                1584
               4                 892
               5                 113
               3                  44
1              1                5284
               0                2882
               2                 978
               4                 872
               5                 118
               3                  71
2              1                5546
               2                5192
               0                4714
               4                2223
               5                 235
               3                  55
3              1               11004
               0                4591
               2                1823
               4                1566
               5                 411
               3                  62
4              1                1603
               0                1200
               2                 411
               4                 104
               3                  74
               5                  46
Name: agency_name, dtype: int64

Variable: agency_abbr
Cluster_Label  agency_abbr
0              3                4077
               0                2225
               1                1584
               4                 892
               5                 113
```

```
                2                    44
1               3                  5284
                0                  2882
                1                   978
                4                   872
                5                   118
                2                    71
2               3                  5546
                1                  5192
                0                  4714
                4                  2223
                5                   235
                2                    55
3               3                 11004
                0                  4591
                1                  1823
                4                  1566
                5                   411
                2                    62
4               3                  1603
                0                  1200
                1                   411
                4                   104
                2                    74
                5                    46
Name: agency_abbr, dtype: int64

Variable: action_taken_name
Cluster_Label  action_taken_name
0              4                      8732
               2                       141
               3                        41
               5                        10
               1                         8
               0                         2
               6                         1
1              4                      7678
               2                      1261
               5                       725
               3                       386
               1                        82
               0                        73
2              4                     17824
               2                        66
               1                        35
               5                        18
               3                        11
               6                         7
               0                         3
```

```
                7                          1
3               4                      18155
                2                       1192
                7                         34
                1                         33
                5                         20
                6                          9
                3                          8
                0                          6
4               4                       3426
                2                          6
                1                          3
                5                          3
Name: action_taken_name, dtype: int64

time: 277 ms (started: 2024-03-16 13:34:24 +00:00)
```

```
# Define a function to get the dominant category for each cluster
def get_dominant_category(cluster_data):
    return cluster_data.idxmax()

# Create a dictionary to store the dominant categories for each
cluster and variable
cluster_profiles = {}

# Iterate over relevant columns
for column in relevant_columns:
    # Group the data by 'Cluster_Label' and get value counts for each
cluster
    cluster_freq = df_ppd_subset.groupby('Cluster_Label')
[column].value_counts()

    # Find the dominant category for each cluster
    dominant_categories =
cluster_freq.groupby(level=0).apply(get_dominant_category)

    # Store the dominant categories in the dictionary
    cluster_profiles[column] = dominant_categories

# Print the cluster profiling
for column, dominant_categories in cluster_profiles.items():
    print(f"Variable: {column}")
    for cluster, category in dominant_categories.items():
        print(f"Cluster {cluster}: Dominant category - {category}")
    print()
```

```
Variable: purchaser_type_name
Cluster 0: Dominant category - (0, 7)
Cluster 1: Dominant category - (1, 7)
Cluster 2: Dominant category - (2, 2)
```

```
Cluster 3: Dominant category - (3, 7)
Cluster 4: Dominant category - (4, 5)

Variable: property_type_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 2)
Cluster 2: Dominant category - (2, 2)
Cluster 3: Dominant category - (3, 2)
Cluster 4: Dominant category - (4, 2)

Variable: preapproval_name
Cluster 0: Dominant category - (0, 0)
Cluster 1: Dominant category - (1, 0)
Cluster 2: Dominant category - (2, 0)
Cluster 3: Dominant category - (3, 0)
Cluster 4: Dominant category - (4, 0)

Variable: owner_occupancy_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 2)
Cluster 2: Dominant category - (2, 2)
Cluster 3: Dominant category - (3, 2)
Cluster 4: Dominant category - (4, 2)

Variable: msamd_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 8)
Cluster 2: Dominant category - (2, 8)
Cluster 3: Dominant category - (3, 7)
Cluster 4: Dominant category - (4, 1)

Variable: loan_type_name
Cluster 0: Dominant category - (0, 0)
Cluster 1: Dominant category - (1, 0)
Cluster 2: Dominant category - (2, 0)
Cluster 3: Dominant category - (3, 0)
Cluster 4: Dominant category - (4, 0)

Variable: loan_purpose_name
Cluster 0: Dominant category - (0, 1)
Cluster 1: Dominant category - (1, 1)
Cluster 2: Dominant category - (2, 2)
Cluster 3: Dominant category - (3, 2)
Cluster 4: Dominant category - (4, 1)

Variable: lien_status_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 2)
Cluster 2: Dominant category - (2, 2)
Cluster 3: Dominant category - (3, 2)
```

```
Cluster 4: Dominant category - (4, 2)

Variable: hoepa_status_name
Cluster 0: Dominant category - (0, 1)
Cluster 1: Dominant category - (1, 1)
Cluster 2: Dominant category - (2, 1)
Cluster 3: Dominant category - (3, 1)
Cluster 4: Dominant category - (4, 1)

Variable: county_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 38)
Cluster 2: Dominant category - (2, 16)
Cluster 3: Dominant category - (3, 5)
Cluster 4: Dominant category - (4, 17)

Variable: co_applicant_sex_name
Cluster 0: Dominant category - (0, 3)
Cluster 1: Dominant category - (1, 3)
Cluster 2: Dominant category - (2, 3)
Cluster 3: Dominant category - (3, 3)
Cluster 4: Dominant category - (4, 0)

Variable: co_applicant_ethnicity_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 2)
Cluster 2: Dominant category - (2, 2)
Cluster 3: Dominant category - (3, 2)
Cluster 4: Dominant category - (4, 3)

Variable: applicant_sex_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 2)
Cluster 2: Dominant category - (2, 2)
Cluster 3: Dominant category - (3, 2)
Cluster 4: Dominant category - (4, 2)

Variable: applicant_ethnicity_name
Cluster 0: Dominant category - (0, 2)
Cluster 1: Dominant category - (1, 2)
Cluster 2: Dominant category - (2, 2)
Cluster 3: Dominant category - (3, 2)
Cluster 4: Dominant category - (4, 2)

Variable: agency_name
Cluster 0: Dominant category - (0, 1)
Cluster 1: Dominant category - (1, 1)
Cluster 2: Dominant category - (2, 1)
Cluster 3: Dominant category - (3, 1)
Cluster 4: Dominant category - (4, 1)
```

```
Variable: agency_abbr
Cluster 0: Dominant category - (0, 3)
Cluster 1: Dominant category - (1, 3)
Cluster 2: Dominant category - (2, 3)
Cluster 3: Dominant category - (3, 3)
Cluster 4: Dominant category - (4, 3)

Variable: action_taken_name
Cluster 0: Dominant category - (0, 4)
Cluster 1: Dominant category - (1, 4)
Cluster 2: Dominant category - (2, 4)
Cluster 3: Dominant category - (3, 4)
Cluster 4: Dominant category - (4, 4)

time: 248 ms (started: 2024-03-16 13:34:24 +00:00)
```

```
# Filter the dataframe to include only data points from Cluster 4
cluster_4_data = df_ppd_subset[df_ppd_subset['Cluster_Label'] == 4]

# List of categorical variables
categorical_variables = ['purchaser_type_name', 'property_type_name',
'preapproval_name',
                         'owner_occupancy_name', 'msamd_name',
'loan_type_name',
                         'loan_purpose_name', 'lien_status_name',
'hoepa_status_name',
                         'county_name', 'co_applicant_sex_name',
'co_applicant_ethnicity_name',
                         'applicant_sex_name',
'applicant_ethnicity_name', 'agency_name',
                         'action_taken_name']

# List of numerical variables
numerical_variables = ['tract_to_msamd_income', 'population',
'minority_population',
                       'number_of_owner_occupied_units',
'number_of_1_to_4_family_units',
                       'loan_amount_000s', 'applicant_income_000s',
'census_tract_number',
                       'hud_median_family_income',
'application_date_indicator']

# Frequency calculation for categorical variables
categorical_freq =
cluster_4_data[categorical_variables].apply(pd.Series.value_counts)

# Mean calculation for numerical variables
numerical_mean = cluster_4_data[numerical_variables].mean()
```

```python
# Display the frequencies and mean values
print("Frequency of Categorical Variables in Cluster 4:")
print(categorical_freq)
print("\nMean of Numerical Variables in Cluster 4:")
print(numerical_mean)
```

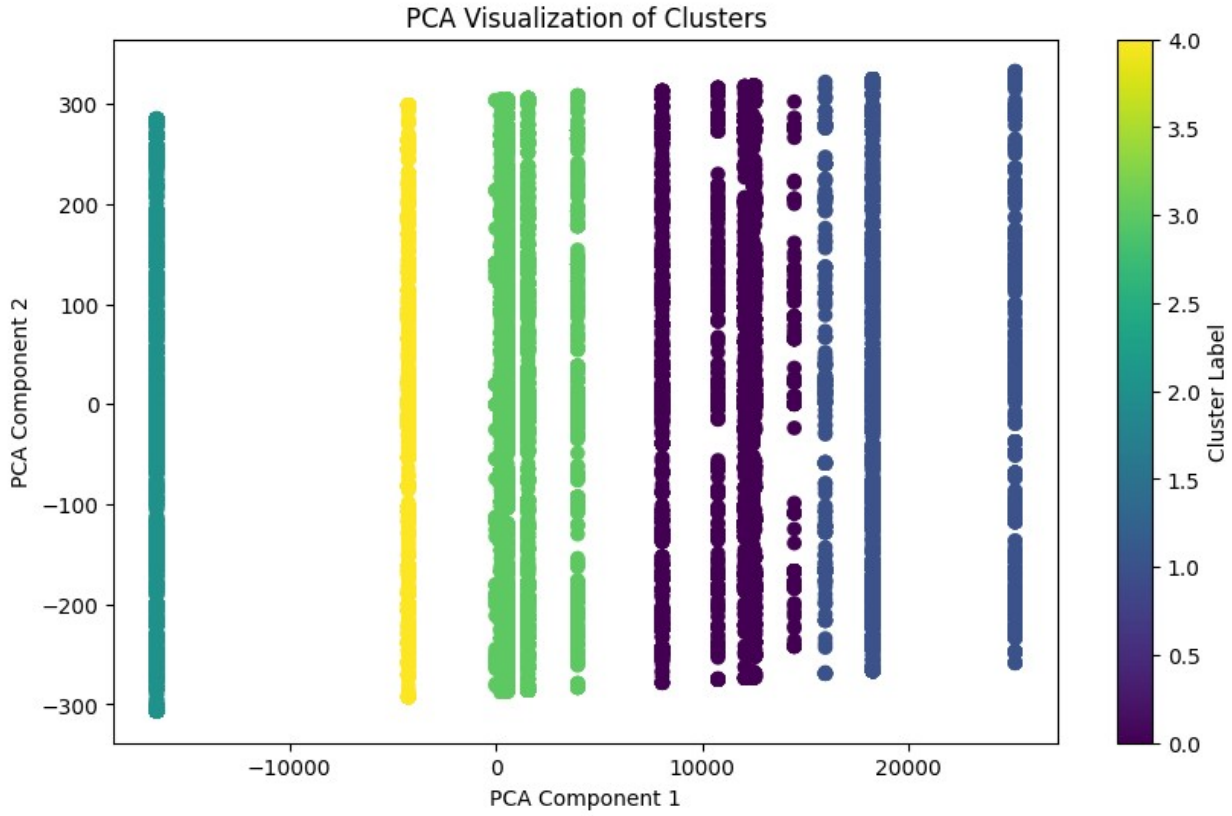
```
Frequency of Categorical Variables in Cluster 4:
    purchaser_type_name  property_type_name  preapproval_name  \
0                  48.0               133.0            2583.0
1                 188.0                15.0             536.0
2                 606.0              3290.0             319.0
3                   NaN                 NaN               NaN
4                 476.0                 NaN               NaN
5                 905.0                 NaN               NaN
6                 265.0                 NaN               NaN
7                 809.0                 NaN               NaN
8                 102.0                 NaN               NaN
9                  39.0                 NaN               NaN
17                  NaN                 NaN               NaN

    owner_occupancy_name  msamd_name  loan_type_name
loan_purpose_name  \
0                   13.0         NaN          2072.0
230.0
1                  279.0      3438.0           272.0
1617.0
2                 3146.0         NaN            51.0
1591.0
3                    NaN         NaN          1043.0
NaN
4                    NaN         NaN             NaN
NaN
5                    NaN         NaN             NaN
NaN
6                    NaN         NaN             NaN
NaN
7                    NaN         NaN             NaN
NaN
8                    NaN         NaN             NaN
NaN
9                    NaN         NaN             NaN
NaN
17                   NaN         NaN             NaN
NaN

    lien_status_name  hoepa_status_name  county_name
co_applicant_sex_name  \
0                 3.0                NaN          NaN
1555.0
1                63.0             3438.0          NaN
```

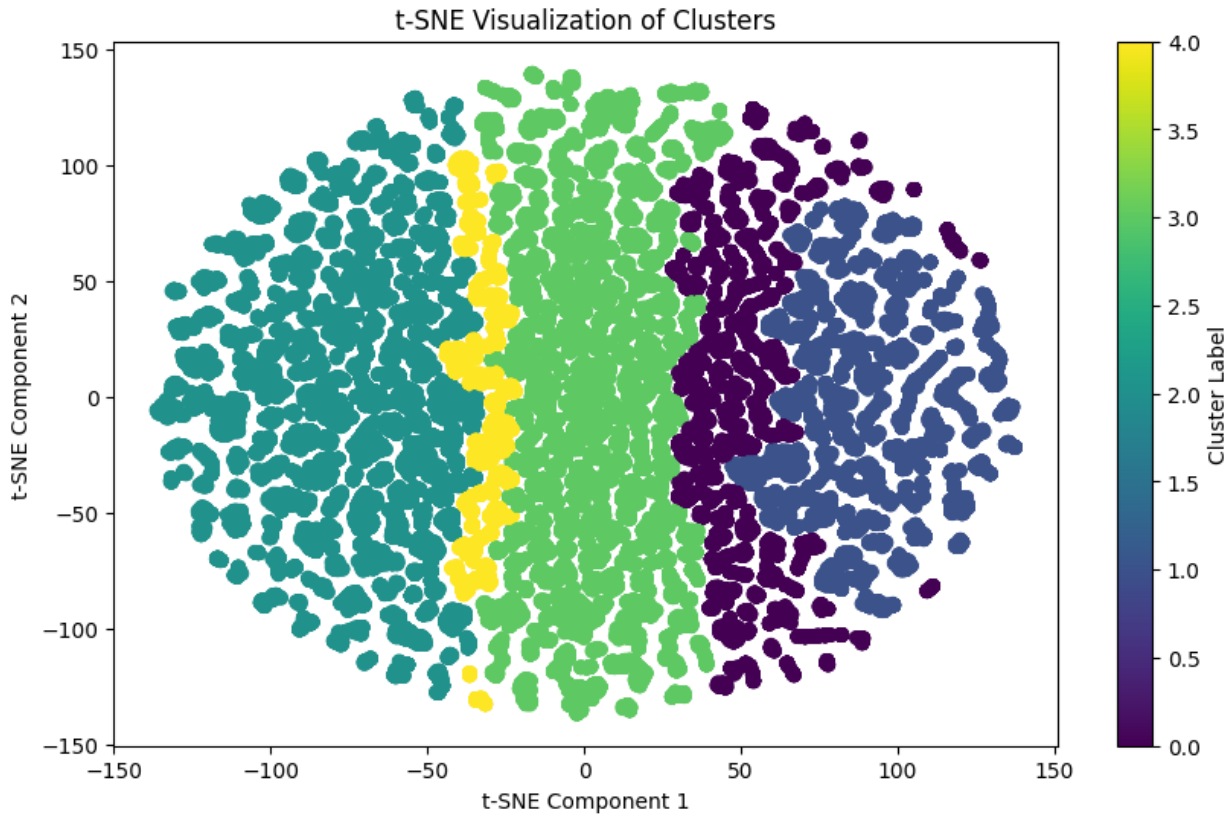
```
149.0
2             3298.0                 NaN           NaN
292.0
3               74.0                 NaN           NaN
1435.0
4                NaN                 NaN           NaN
7.0
5                NaN                 NaN           NaN
NaN
6                NaN                 NaN           NaN
NaN
7                NaN                 NaN           NaN
NaN
8                NaN                 NaN           NaN
NaN
9                NaN                 NaN           NaN
NaN
17               NaN                 NaN        3438.0
NaN

    co_applicant_ethnicity_name  applicant_sex_name
applicant_ethnicity_name  \
0                            65.0               745.0
133.0
1                           270.0               227.0
385.0
2                          1435.0              2431.0
2884.0
3                          1659.0                35.0
36.0
4                             9.0                 NaN
NaN
5                             NaN                 NaN
NaN
6                             NaN                 NaN
NaN
7                             NaN                 NaN
NaN
8                             NaN                 NaN
NaN
9                             NaN                 NaN
NaN
17                            NaN                 NaN
NaN

    agency_name  action_taken_name
0        1200.0                NaN
1        1603.0                3.0
2         411.0                6.0
```

```
3           74.0                  NaN
4          104.0               3426.0
5           46.0                  3.0
6            NaN                  NaN
7            NaN                  NaN
8            NaN                  NaN
9            NaN                  NaN
17           NaN                  NaN

Mean of Numerical Variables in Cluster 4:
tract_to_msamd_income                 0.383081
population                            0.389814
minority_population                   0.186917
number_of_owner_occupied_units        0.477716
number_of_1_to_4_family_units         0.322025
loan_amount_000s                      0.005278
applicant_income_000s                 0.017556
census_tract_number                   0.128075
hud_median_family_income          78100.000000
application_date_indicator            0.001745
dtype: float64
time: 129 ms (started: 2024-03-16 13:34:25 +00:00)
```

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

# Reduce dimensionality using PCA
pca = PCA(n_components=2)
pca_result =
pca.fit_transform(df_ppd_subset.drop(columns=['Cluster_Label']))

# Visualize clusters using PCA
plt.figure(figsize=(10, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1],
c=df_ppd_subset['Cluster_Label'], cmap='viridis')
plt.title('PCA Visualization of Clusters')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster Label')
plt.show()

# Reduce dimensionality using t-SNE
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
tsne_result =
tsne.fit_transform(df_ppd_subset.drop(columns=['Cluster_Label']))

# Visualize clusters using t-SNE
plt.figure(figsize=(10, 6))
plt.scatter(tsne_result[:, 0], tsne_result[:, 1],
```

```
c=df_ppd_subset['Cluster_Label'], cmap='viridis')
plt.title('t-SNE Visualization of Clusters')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.colorbar(label='Cluster Label')
plt.show()
```

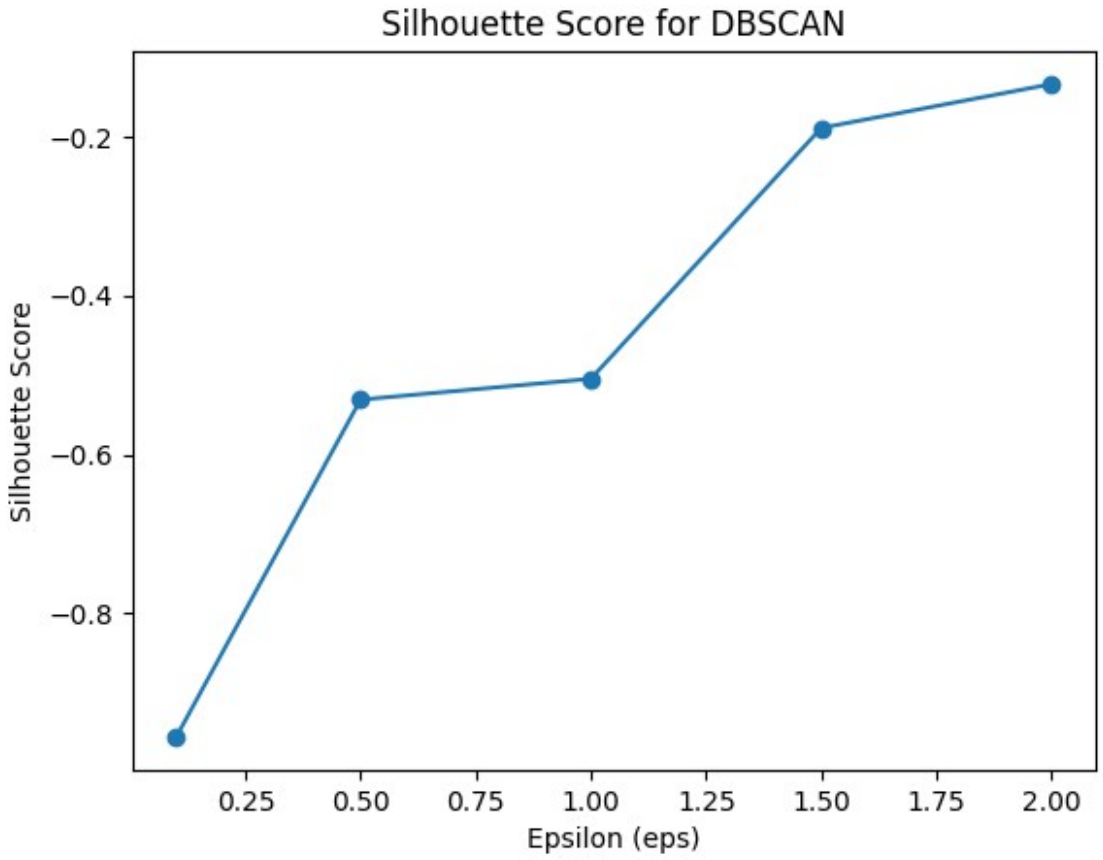





```
time: 15min 41s (started: 2024-03-16 13:34:25 +00:00)
```

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

# Assuming 'data' is your feature matrix
silhouette_scores = []
eps_values = [0.1, 0.5, 1.0, 1.5, 2.0]  # Example values for epsilon

for eps in eps_values:
    dbscan = DBSCAN(eps=eps)
    dbscan.fit(df_ppd_subset)
    if len(set(dbscan.labels_)) > 1:  # Silhouette score requires at
least 2 clusters
        silhouette_scores.append(silhouette_score(df_ppd_subset,
dbscan.labels_))
    else:
        silhouette_scores.append(-1)  # Silhouette score is undefined
for 1 cluster

# Plotting the silhouette scores
plt.plot(eps_values, silhouette_scores, marker='o')
plt.title('Silhouette Score for DBSCAN')
plt.xlabel('Epsilon (eps)')
```

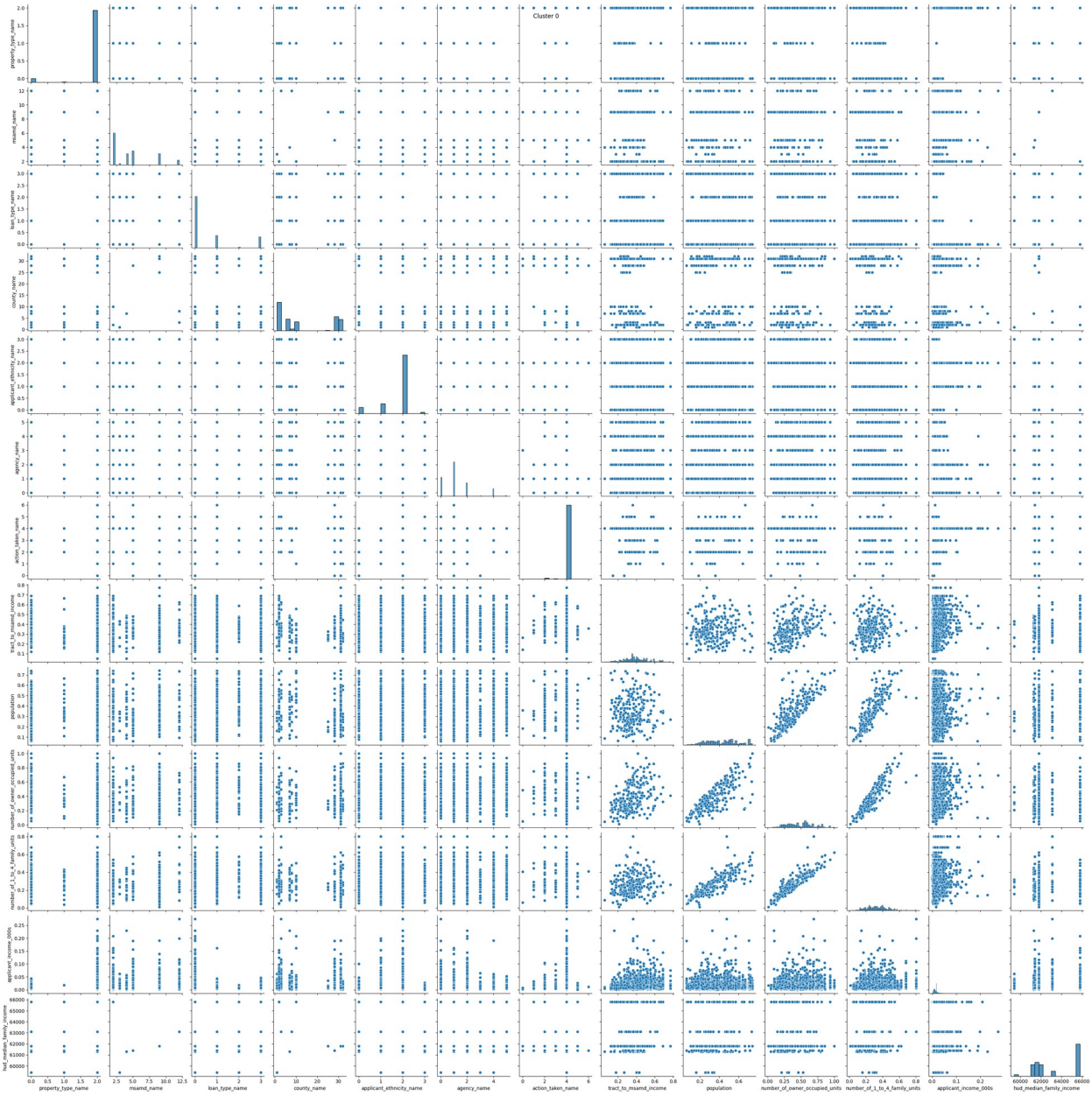
```
plt.ylabel('Silhouette Score')
plt.show()
```

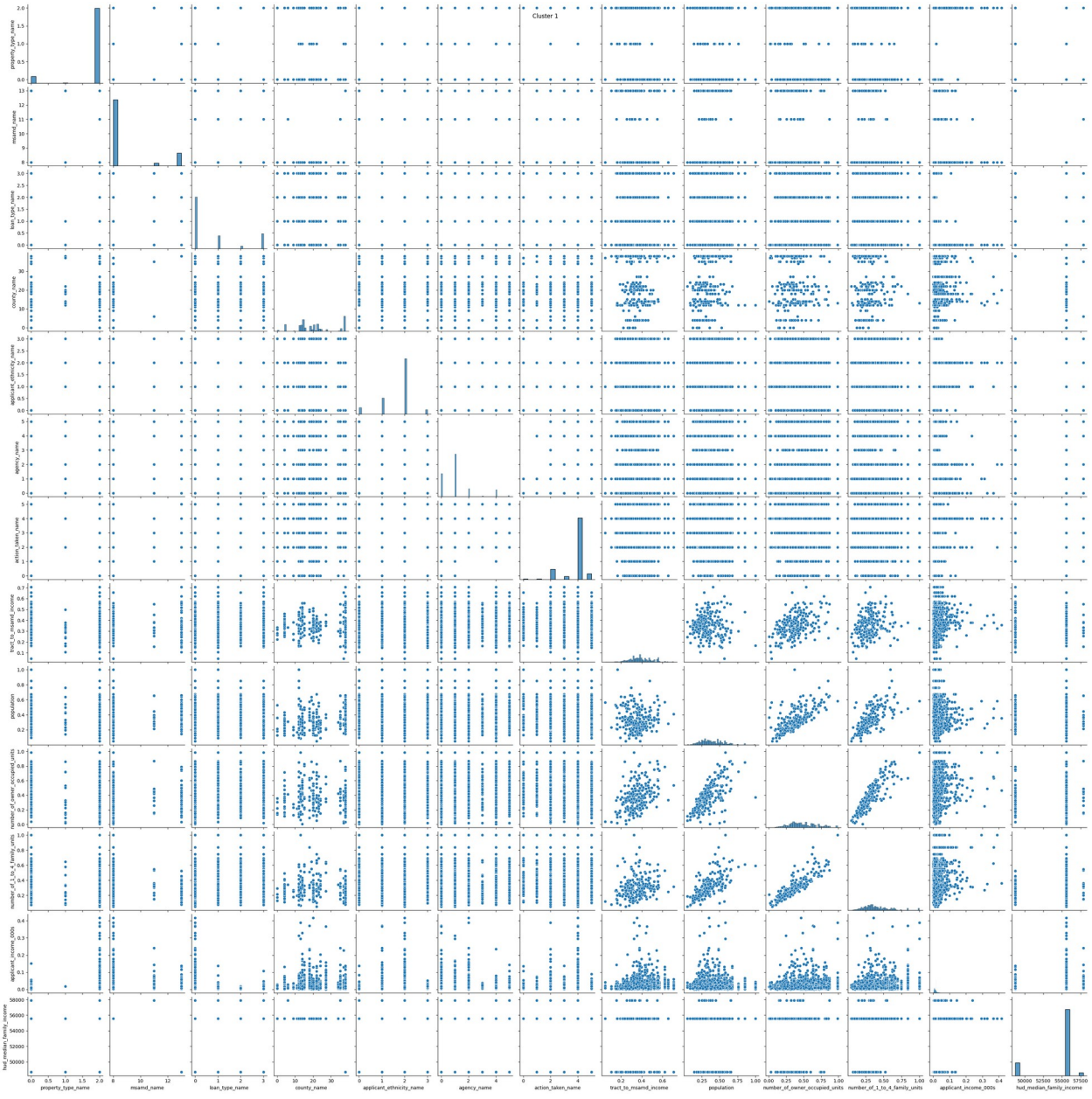


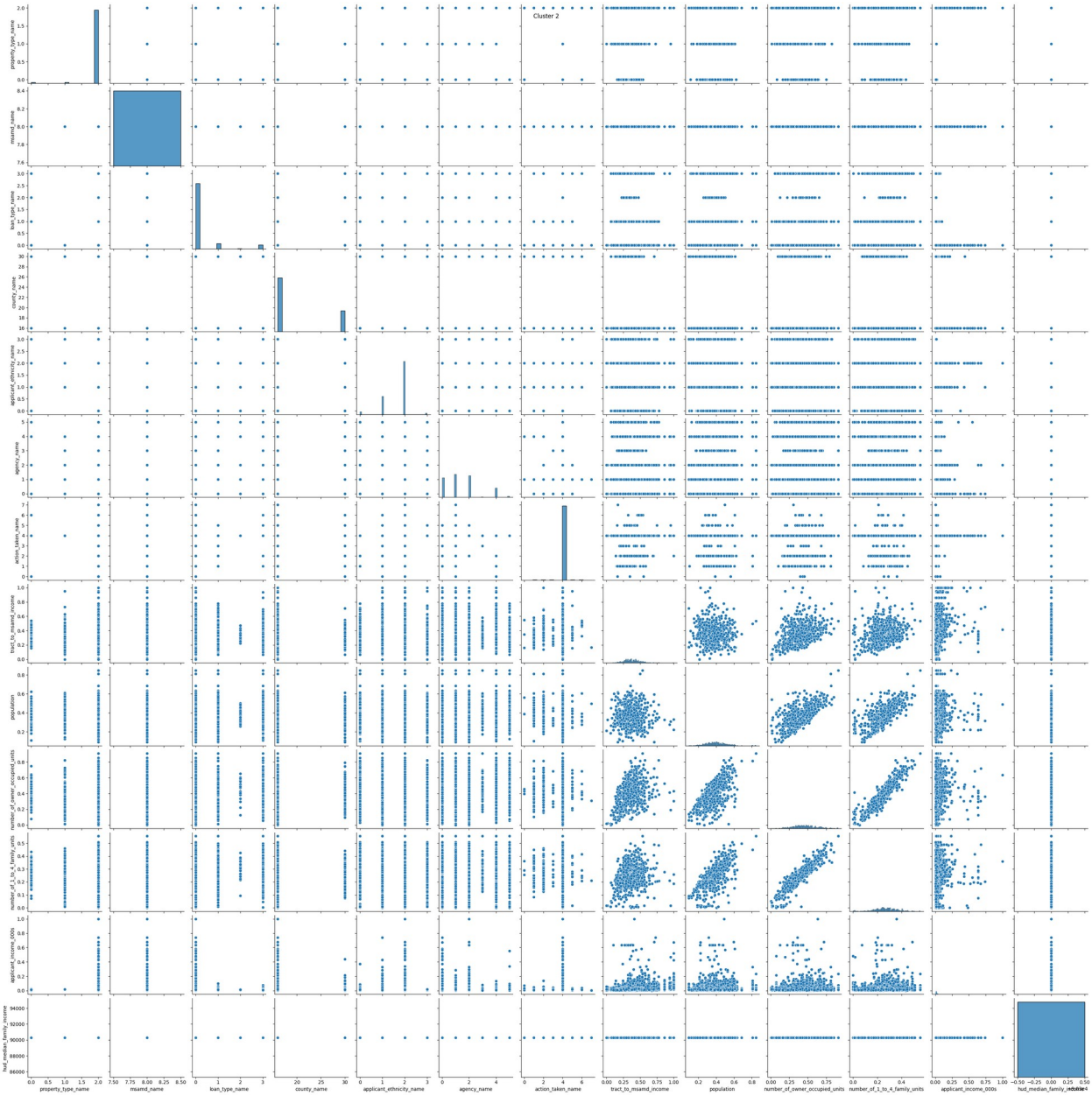

```
time: 5min 59s (started: 2024-03-16 15:21:31 +00:00)
```

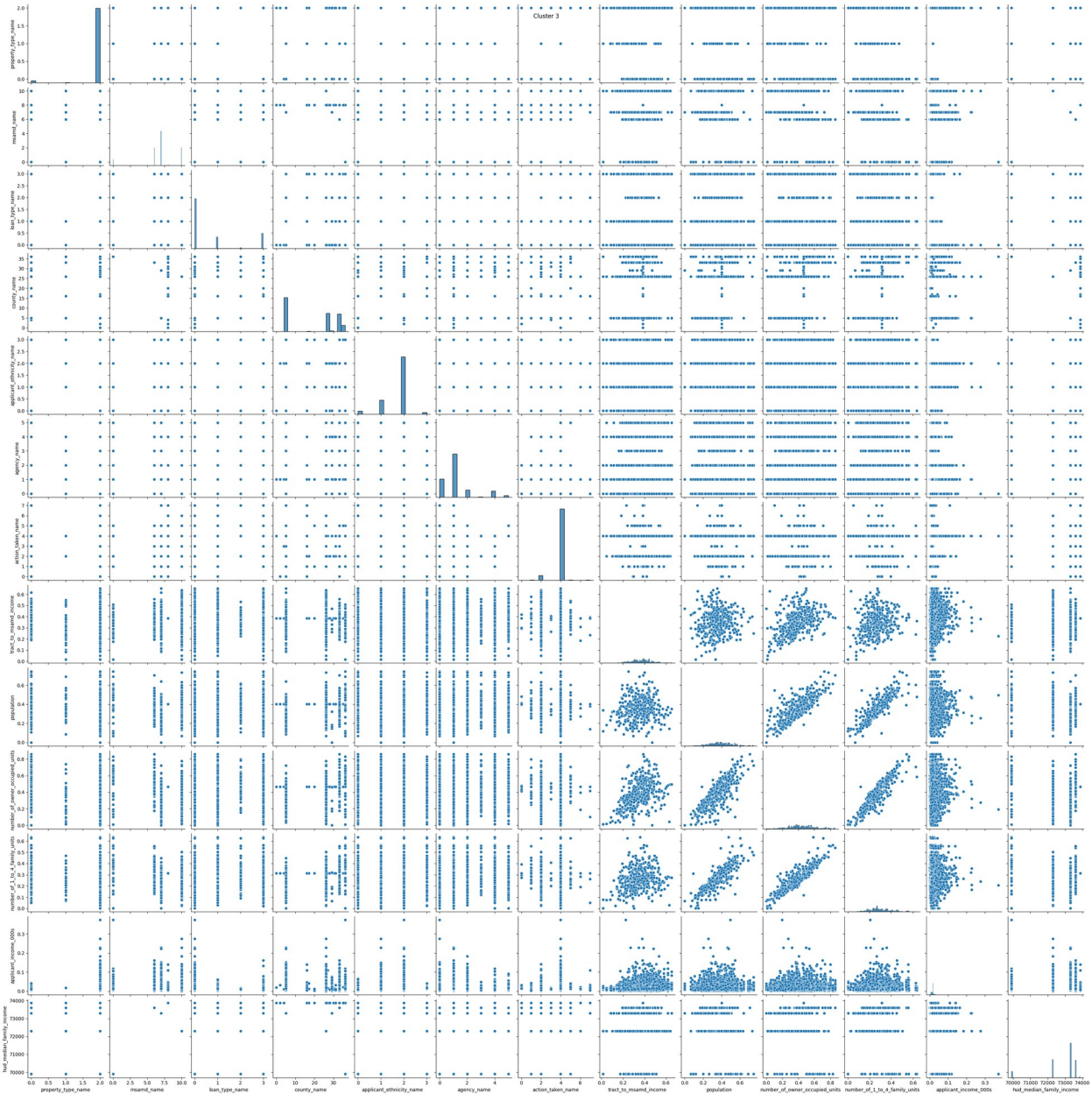
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select top categorical and non-categorical significant variables
top_cat_variables = ['property_type_name', 'msamd_name',
'loan_type_name',
                     'county_name', 'applicant_ethnicity_name',
'agency_name',
                     'action_taken_name']
top_non_cat_variables = ['tract_to_msamd_income', 'population',
                         'number_of_owner_occupied_units',
                         'number_of_1_to_4_family_units',
                         'applicant_income_000s',
'hud_median_family_income']

# Combine all significant variables
significant_variables = top_cat_variables + top_non_cat_variables
```

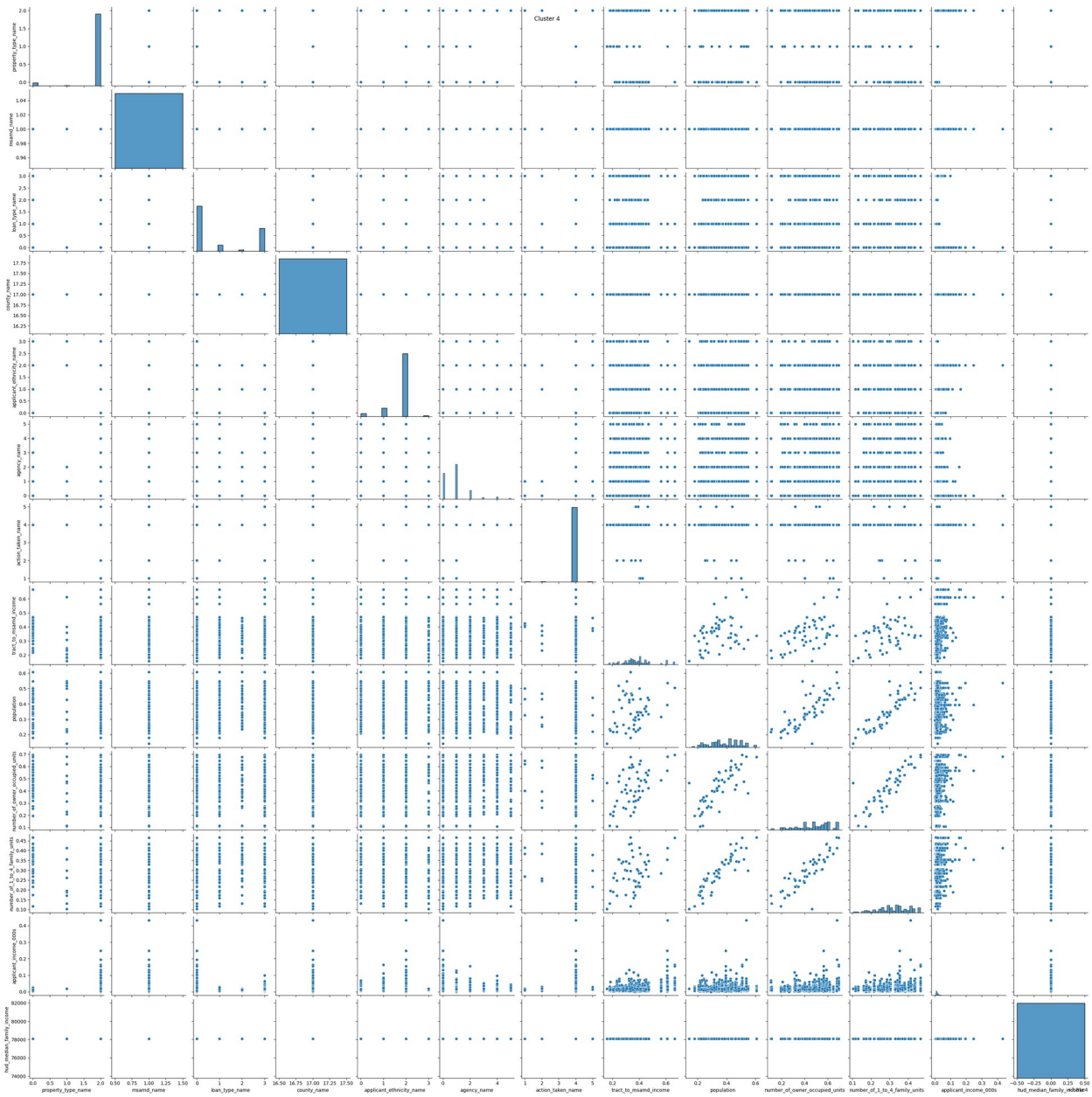
```
# Separate data for each cluster
cluster_data = {}
for cluster_label in range(5):
    cluster_data[cluster_label] =
df_ppd_subset[df_ppd_subset['Cluster_Label'] == cluster_label]

# Create pairplot for all clusters using only significant variables
for cluster_label, data in cluster_data.items():
    # Filter data for significant variables
    filtered_data = data[significant_variables]

    # Plot pairplot
    sns.pairplot(filtered_data)
    plt.suptitle(f'Cluster {cluster_label}')
    plt.show()
```


Cluster 0
property_type_name
msamd_name
loan_type_name
county_name
applicant_ethnicity_name
agency_name
action_taken_name
tract_to_msamd_income
population
number_of_owner_occupied_units
number_of_1_to_4_family_units
applicant_income_000s
hud_median_family_income


Cluster 1
property_type_name
msamd_name
loan_type_name
county_name
applicant_ethnicity_name
agency_name
action_taken_name
tract_to_msamd_income
population
number_of_owner_occupied_units
number_of_1_to_4_family_units
applicant_income_000s
hud_median_family_income


Cluster 2
property_type_name
msamd_name
loan_type_name
county_name
applicant_ethnicity_name
agency_name
action_taken_name
tract_to_msamd_income
population
number_of_owner_occupied_units
number_of_1_to_4_family_units
applicant_income_000s
hud_median_family_income

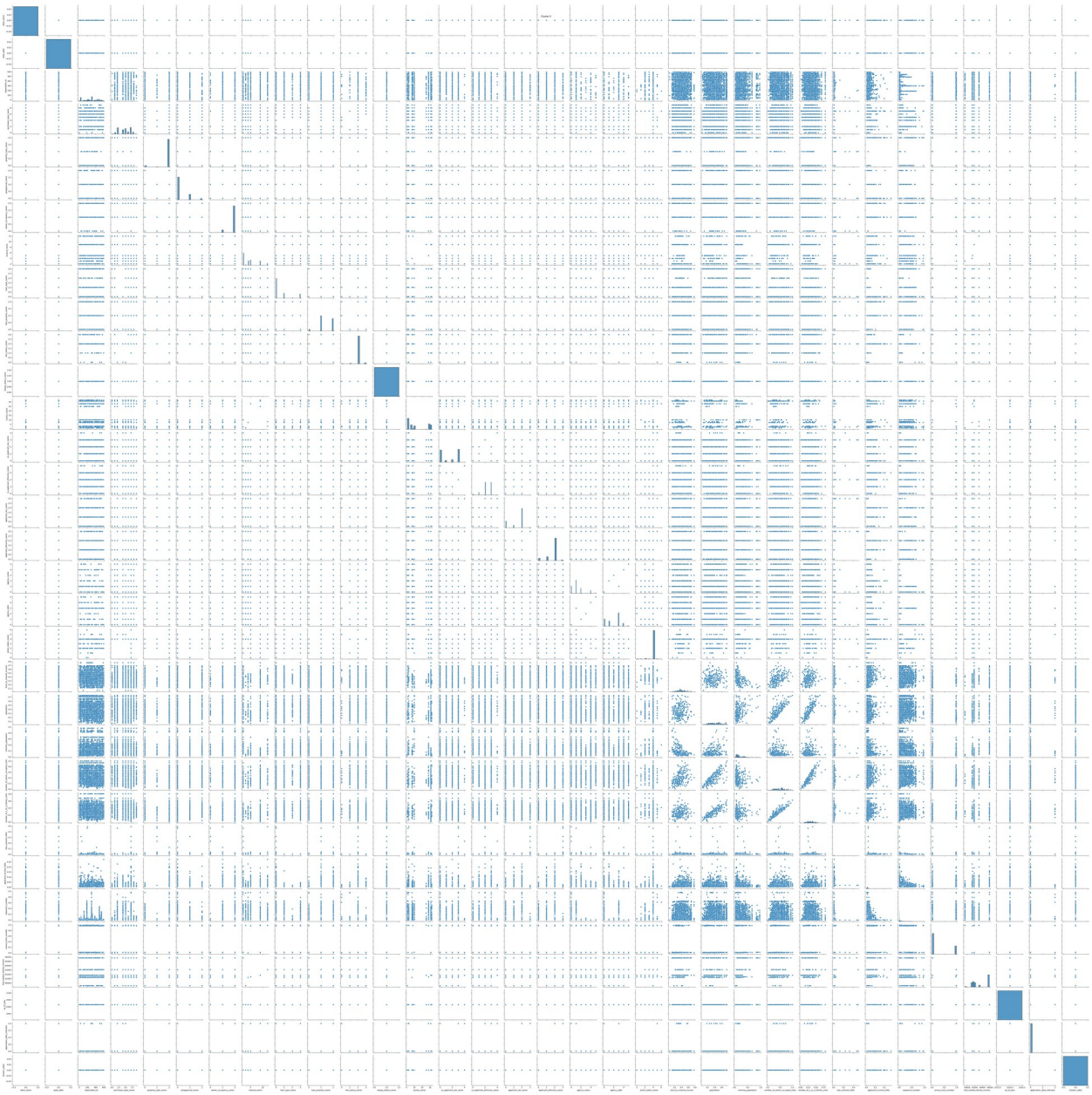

Cluster 3
property_type_name
msamd_name
loan_type_name
county_name
applicant_ethnicity_name
agency_name
action_taken_name
tract_to_msamd_income
population
number_of_owner_occupied_units
number_of_1_to_4_family_units
applicant_income_000s
hud_median_family_income

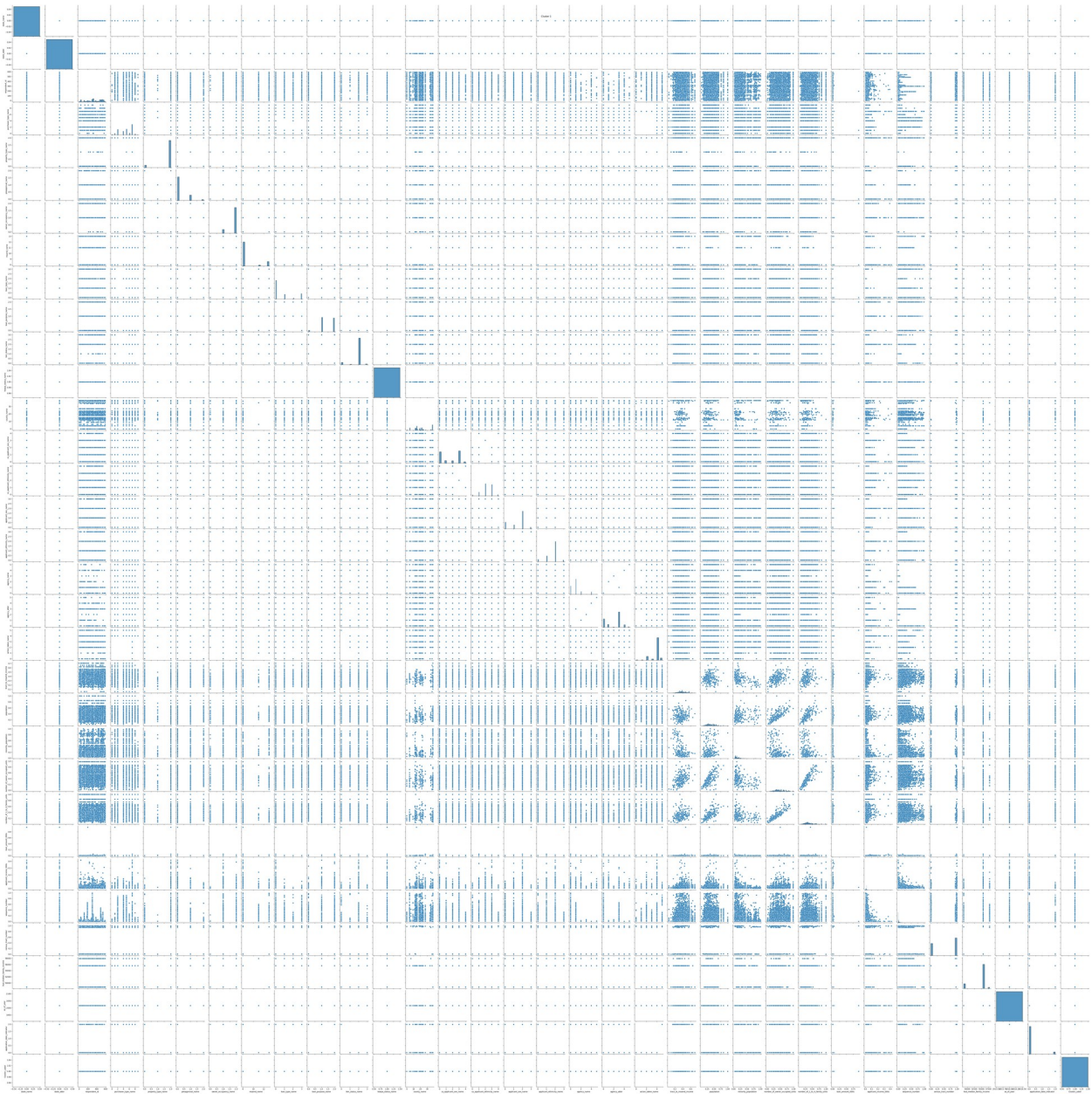


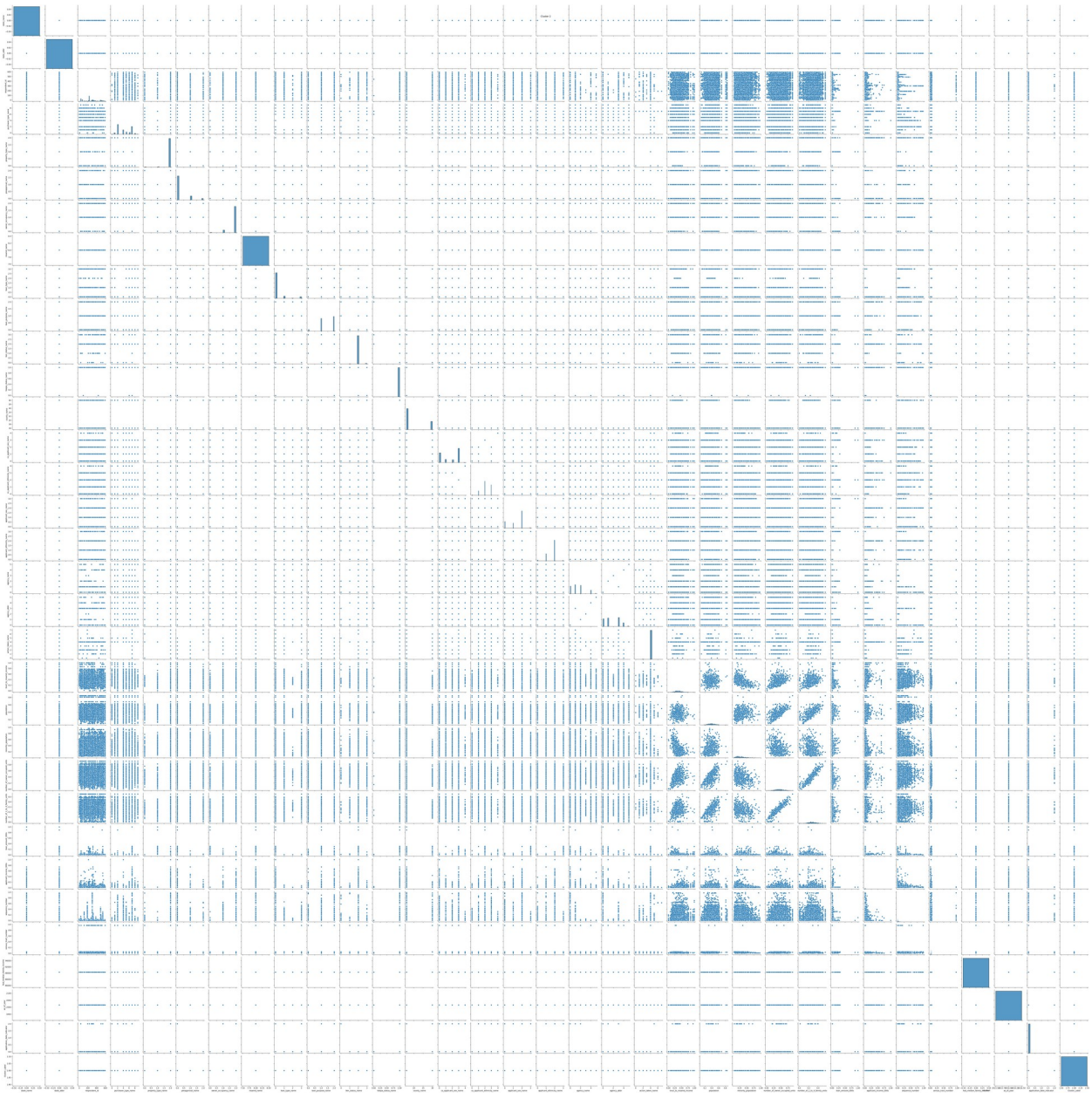
time: 9min 34s (started: 2024-03-16 15:11:56 +00:00)

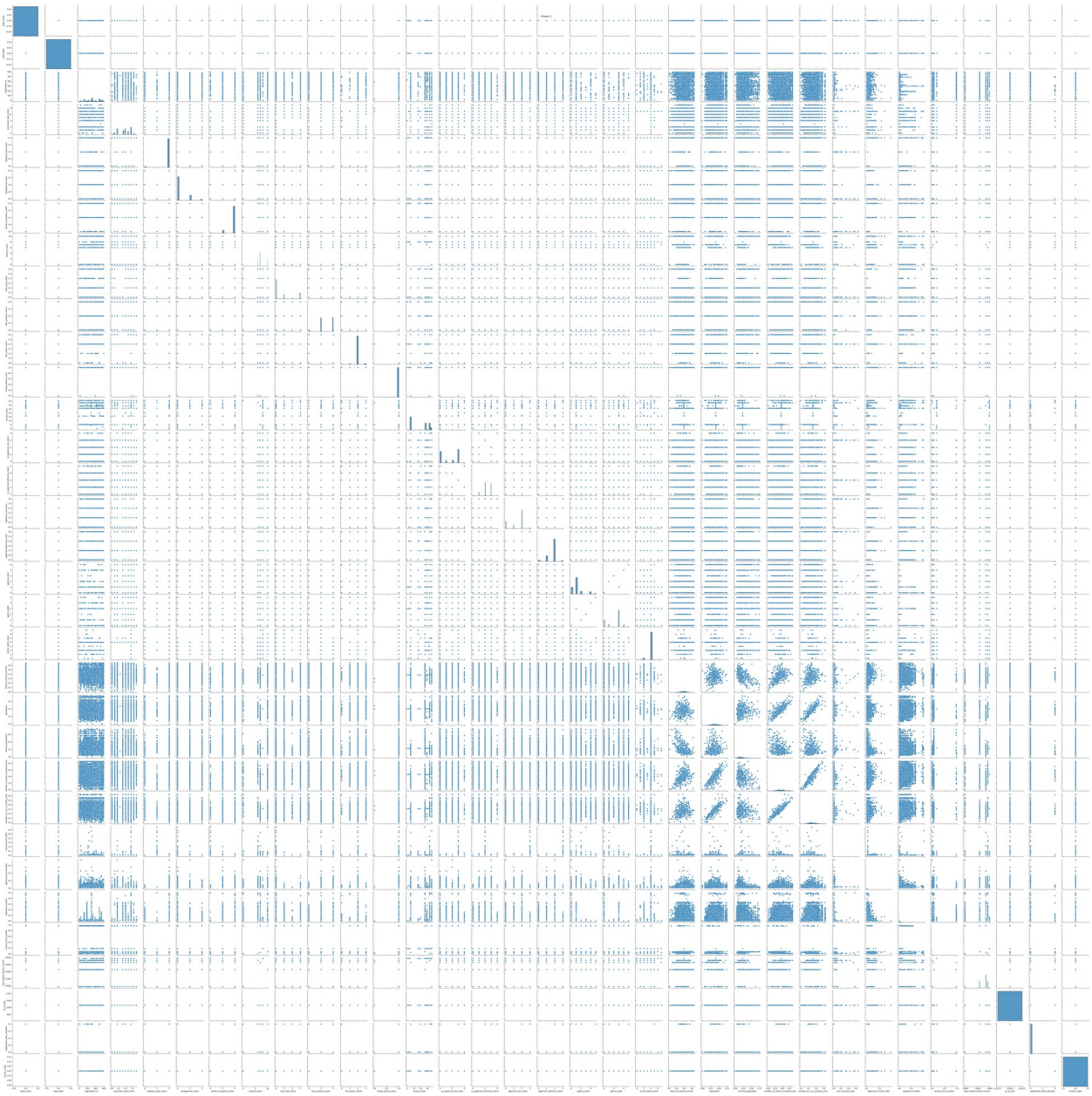
```
'''
import seaborn as sns
import matplotlib.pyplot as plt

# Separate data for each cluster
cluster_data = {}
for cluster_label in range(5):
    cluster_data[cluster_label] =
df_ppd_subset[df_ppd_subset['Cluster_Label'] == cluster_label]
```

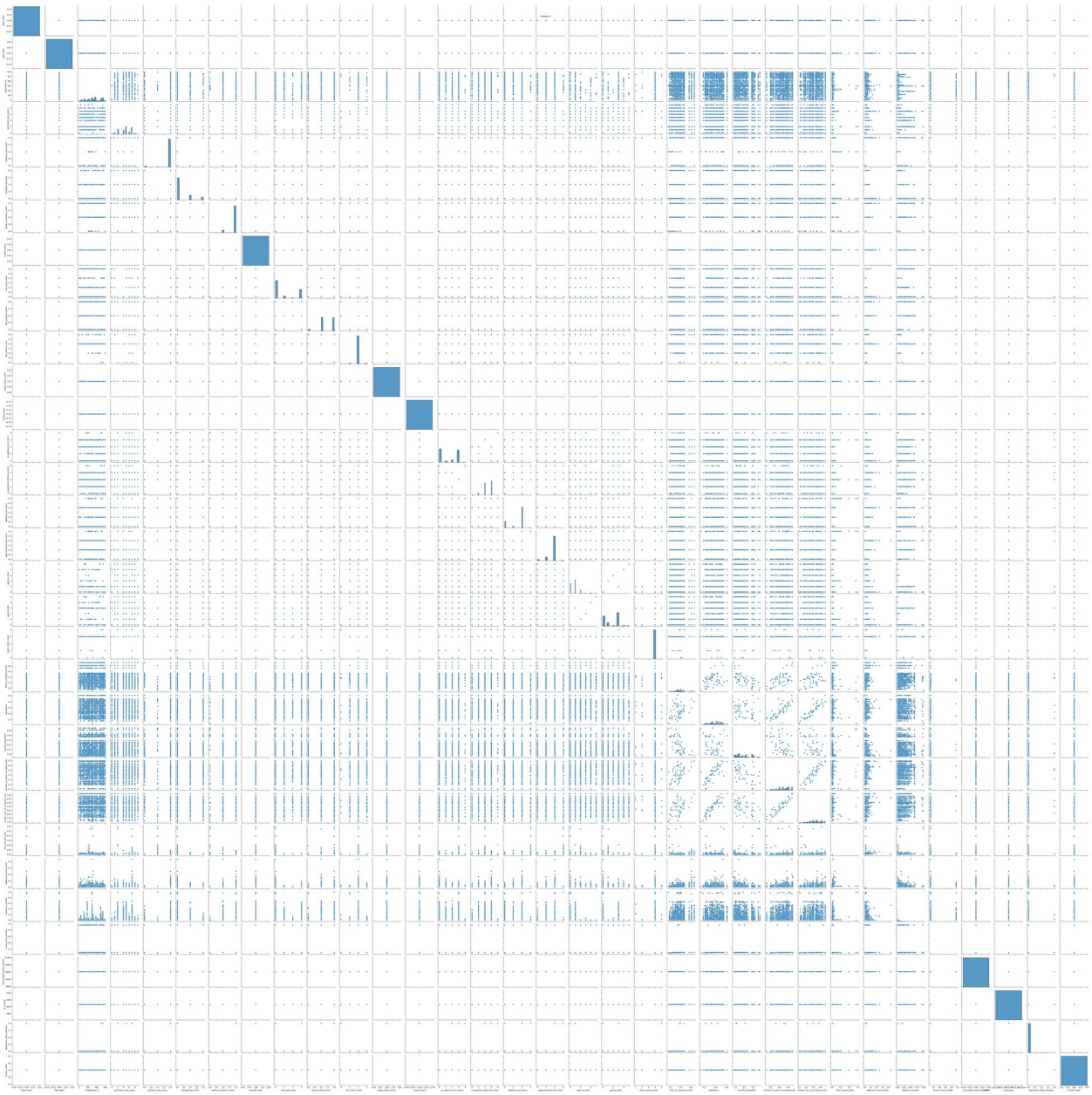
```
# Create pairplot for all clusters
for cluster_label, data in cluster_data.items():
    sns.pairplot(data)
    plt.suptitle(f'Cluster {cluster_label}')
    plt.show()
'''
```











```
time: 1h 20min 23s (started: 2024-03-16 13:50:07 +00:00)
```