Phase 4: Development Part

1)Calculating the distribution of marginal workers based on age, industrial category, and sex

code:

#importing libraries

import pandas as pd

import numpy as np

import geopandas as gpd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

DataSet = pd.read\_csv("Tamil nadu workers.csv")

# Explore the dataset

DataSet.info()

DataSet.columns

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

numerical\_columns = [

'Worked for 3 months or more but less than 6 months - Males',

'Worked for 3 months or more but less than 6 months - Females',

'Worked for less than 3 months - Persons',

'Worked for less than 3 months - Males',

'Worked for less than 3 months - Females',

'Industrial Category - A - Cultivators - Persons',

'Industrial Category - A - Cultivators - Males',

'Industrial Category - A - Cultivators - Females',

'Industrial Category - A - Agricultural labourers - Persons',

'Industrial Category - A - Agricultural labourers - Males',

'Industrial Category - A - Agricultural labourers - Females',

'Industrial Category - A - Plantation, Livestock, Forestry, Fishing, Hunting and allied activities - Persons',

'Industrial Category - A - Plantation, Livestock, Forestry, Fishing, Hunting and allied activities - Males',

'Industrial Category - A - Plantation, Livestock, Forestry, Fishing, Hunting and allied activities - Females',

'Industrial Category - B - Persons',

'Industrial Category - B - Males',

'Industrial Category - B - Females',

'Industrial Category - C - HHI - Persons',

'Industrial Category - C - HHI - Males',

'Industrial Category - C - HHI - Females',

'Industrial Category - C - Non HHI - Persons',

'Industrial Category - C - Non HHI - Males',

'Industrial Category - C - Non HHI - Females',

'Industrial Category - D & E - Persons',

'Industrial Category - D & E - Males',

'Industrial Category - D & E - Females',

'Industrial Category - F - Persons',

'Industrial Category - F - Males',

'Industrial Category - F - Females',

'Industrial Category - G - HHI - Persons',

'Industrial Category - G - HHI - Males',

'Industrial Category - G - HHI - Females',

'Industrial Category - G - Non HHI - Persons',

'Industrial Category - G - Non HHI - Males',

'Industrial Category - G - Non HHI - Females',

'Industrial Category - H - Persons',

'Industrial Category - H - Males',

'Industrial Category - H - Females',

'Industrial Category - I - Persons',

'Industrial Category - I - Males',

'Industrial Category - I - Females',

'Industrial Category - J - HHI - Persons',

'Industrial Category - J - HHI - Males',

'Industrial Category - J - HHI - Females',

'Industrial Category - J - Non HHI - Persons',

'Industrial Category - J - Non HHI - Males',

'Industrial Category - J - Non HHI - Females',

'Industrial Category - K to M - Persons',

'Industrial Category - K to M - Males',

'Industrial Category - K to M - Females',

'Industrial Category - N to O - Persons',

'Industrial Category - N to O - Males',

'Industrial Category - N to O - Females',

'Industrial Category - P to Q - Persons',

'Industrial Category - P to Q - Males',

'Industrial Category - P to Q - Females',

'Industrial Category - R to U - HHI - Persons',

'Industrial Category - R to U - HHI - Males',

'Industrial Category - R to U - HHI - Females',

'Industrial Category - R to U - Non HHI - Persons',

'Industrial Category - R to U - Non HHI - Males',

'Industrial Category - R to U - Non HHI - Females'

]

# Extract unique values of the 'Age group' column

unique\_age\_groups = DataSet['Age group'].unique()

print(unique\_age\_groups)

Output:  
['Total' '`5-14' '15-34' '35-59' '60+' 'Age not stated']

AGE = DataSet['Age group'].replace('Total','')

AGE.unique()

Output:

array(['', '`5-14', '15-34', '35-59', '60+', 'Age not stated'],

dtype=object)

age\_groups = DataSet['Age group']

worked\_3\_to\_6\_months = DataSet['Worked for 3 months or more but less than 6 months - Persons']

Explanation:

1. Essential libraries (Pandas, NumPy, GeoPandas, Matplotlib, and Seaborn) are imported for data manipulation and visualization.
2. A dataset named "Tamil nadu workers.csv" is loaded.
3. Unique values in the 'Age group' column of the dataset are extracted and printed.
4. Data preprocessing is conducted on the 'Age group' column to replace 'Total' with an empty string, resulting in a modified 'AGE' variable.
5. A bar chart is created, showing the number of persons who worked for 3-6 months ('Worked for 3 months or more but less than 6 months - Persons') categorized by different age groups ('AGE'). The chart visualizes this relationship with appropriate labels and aesthetics

# Create a bar chart

plt.figure(figsize=(10, 6))

plt.bar(age\_groups, worked\_3\_to\_6\_months)

plt.xlabel('Age group')

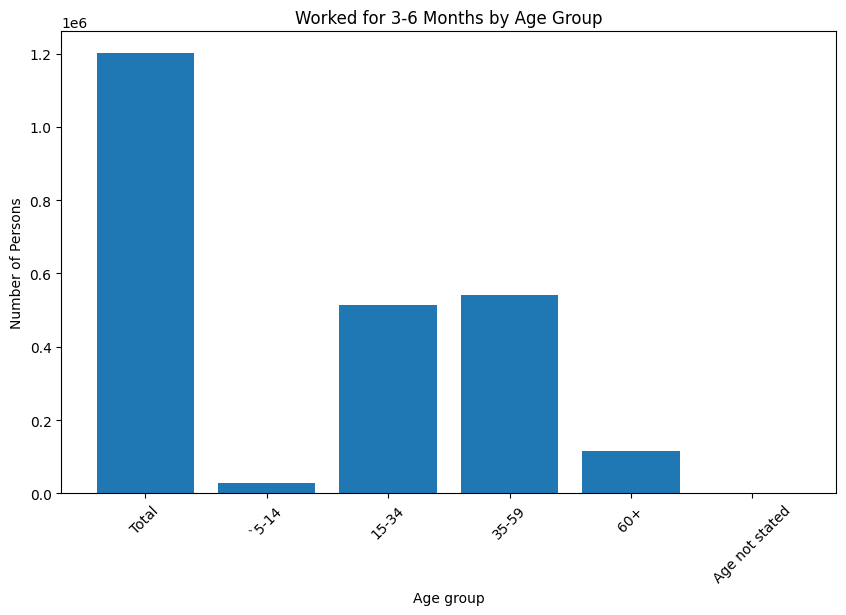
plt.ylabel('Number of Persons')

plt.title('Worked for 3-6 Months by Age Group')

plt.xticks(rotation=45)

plt.show()

Output:



numerical\_columns = numerical\_columns

# Visualize distributions using histograms and kernel density plots

for column in numerical\_columns:

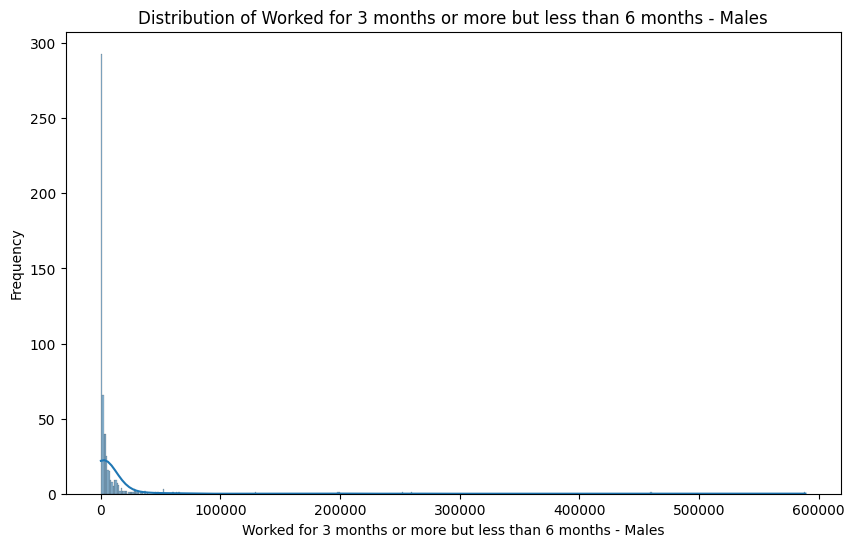
plt.figure(figsize=(10, 6))

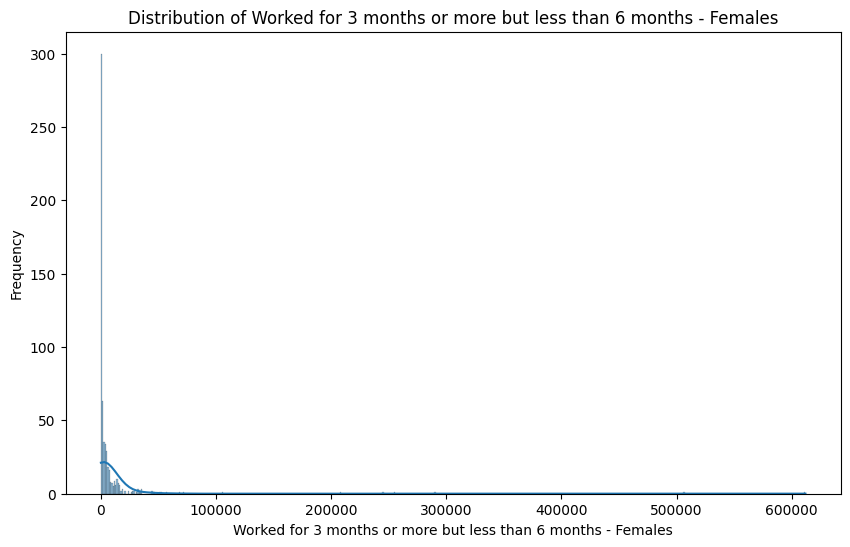
sns.histplot(data=DataSet, x=column, kde=True)

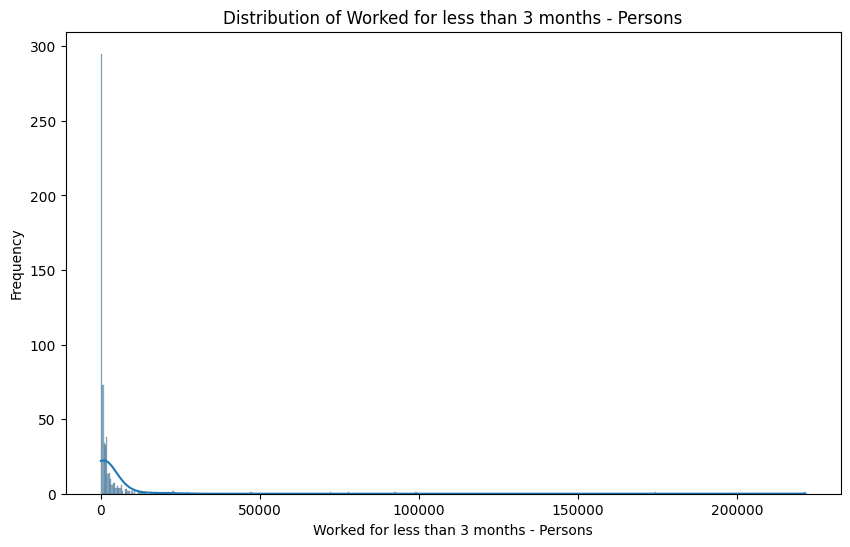
plt.title(f'Distribution of {column}')

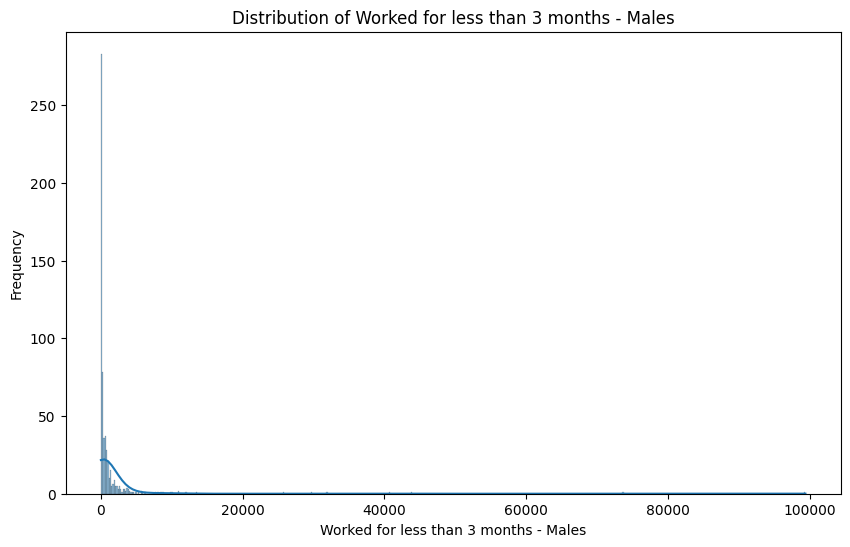
plt.xlabel(column)

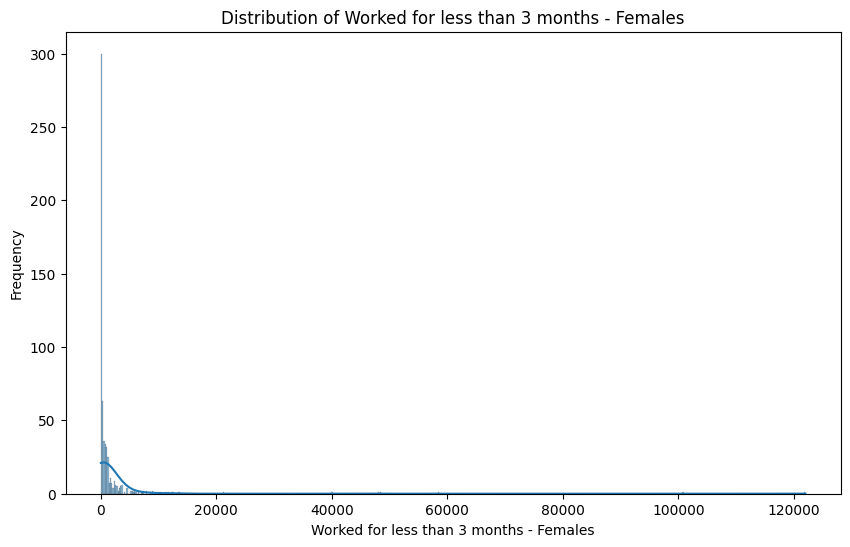
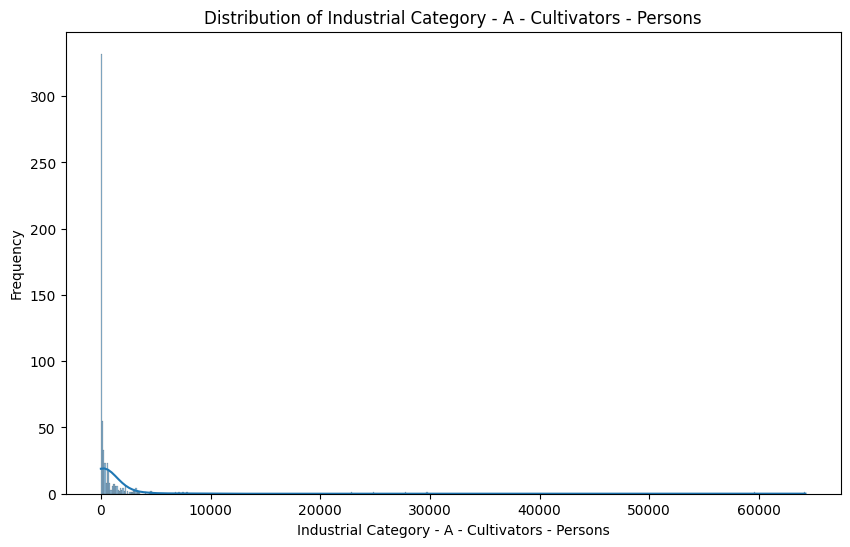
plt.ylabel('Frequency')

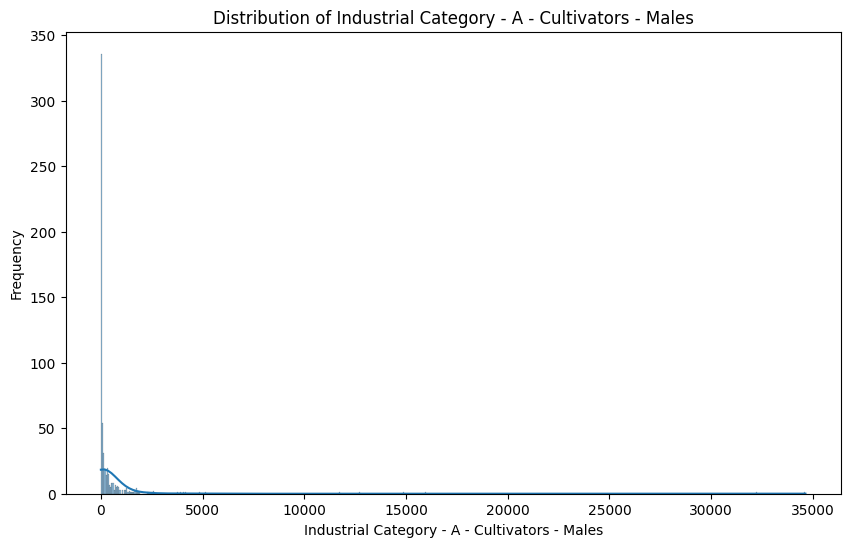
plt.show()  
  
Output:  


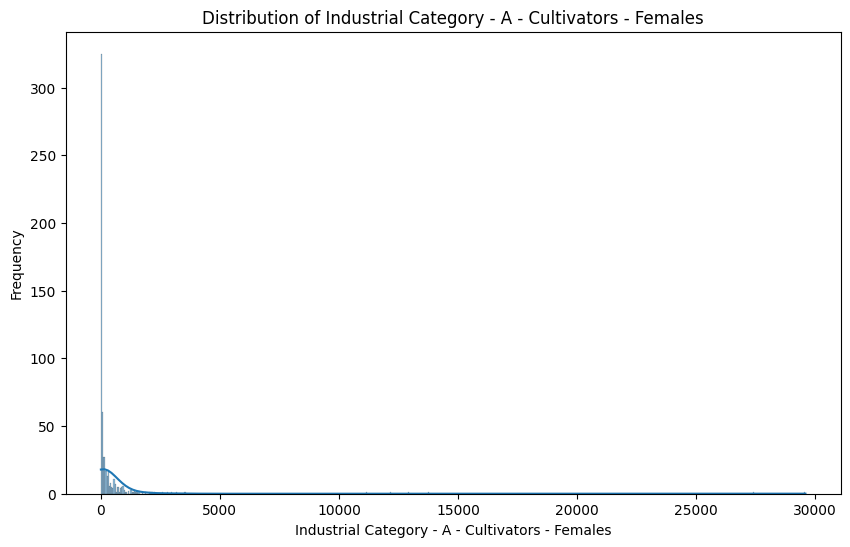


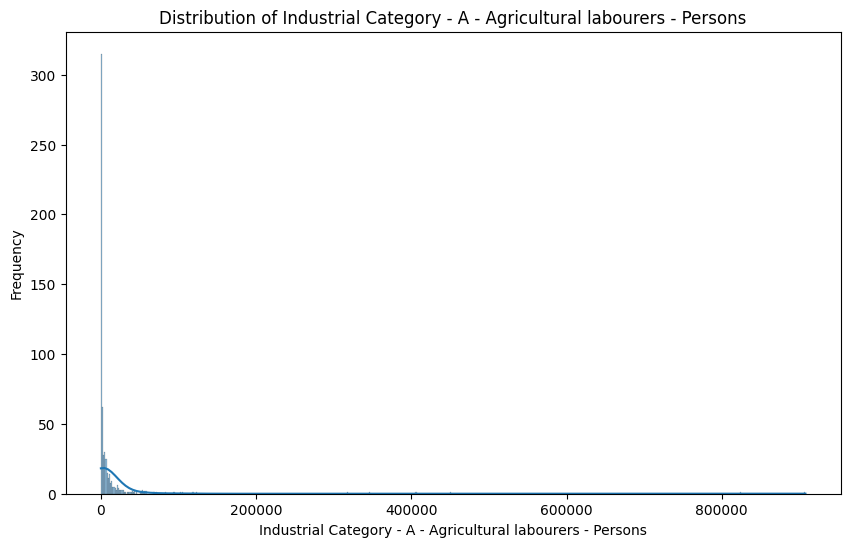


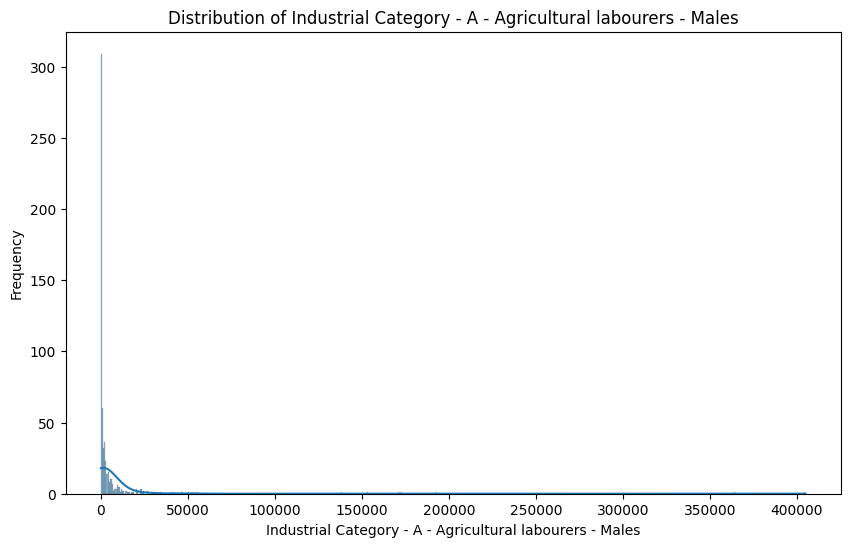


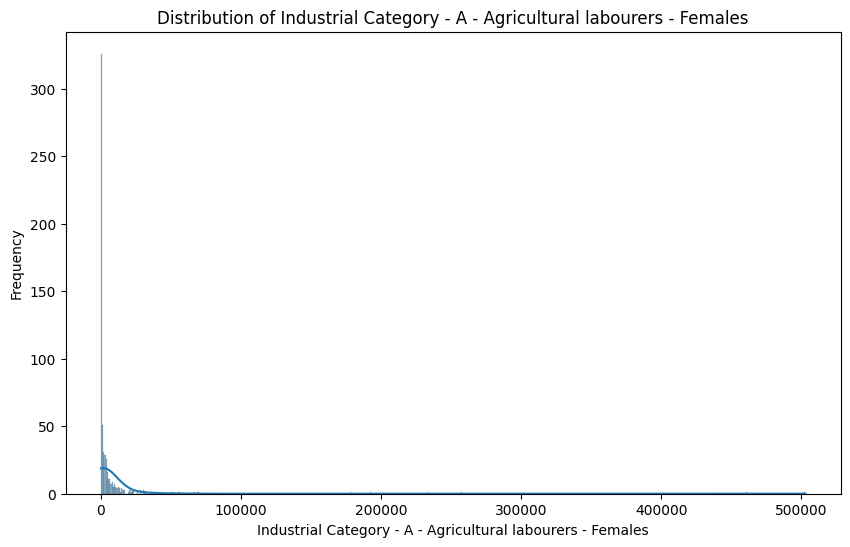
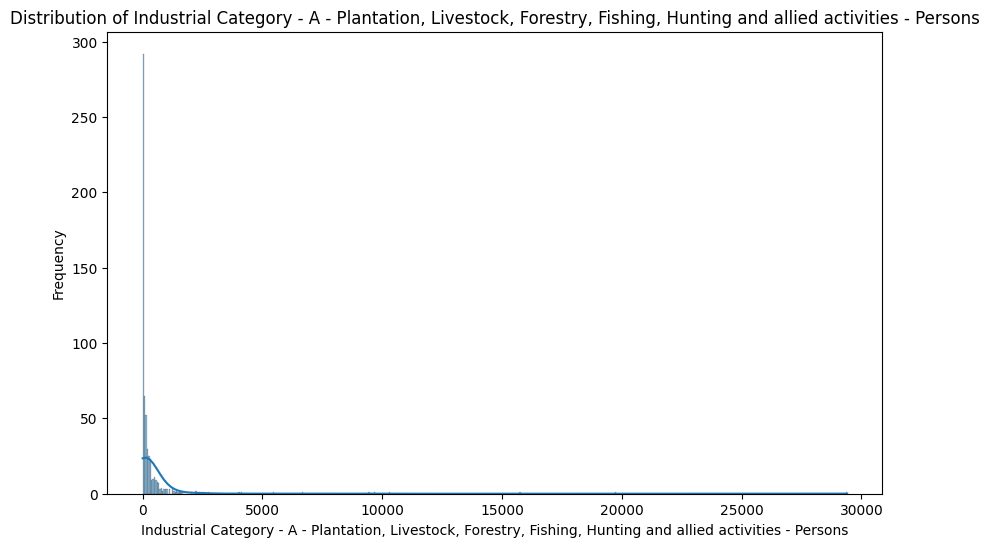
  


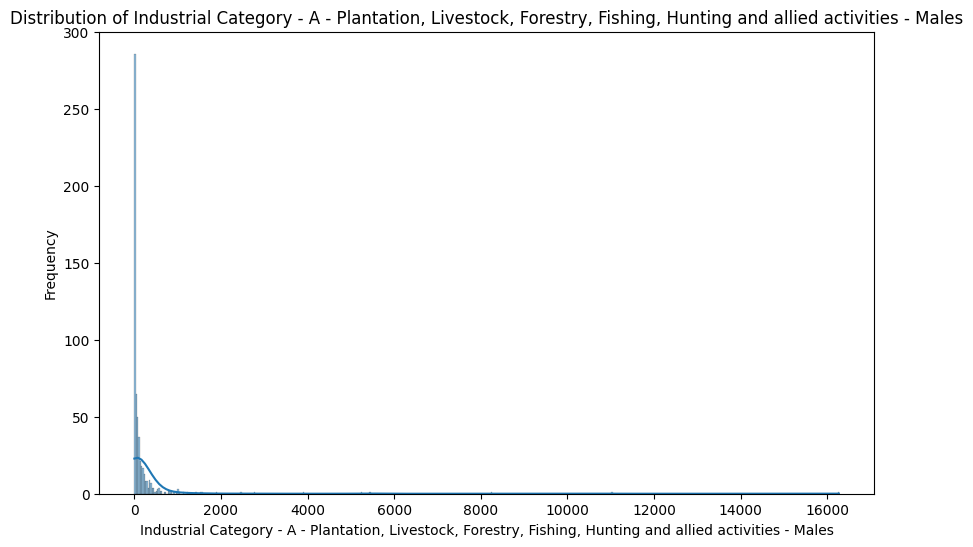


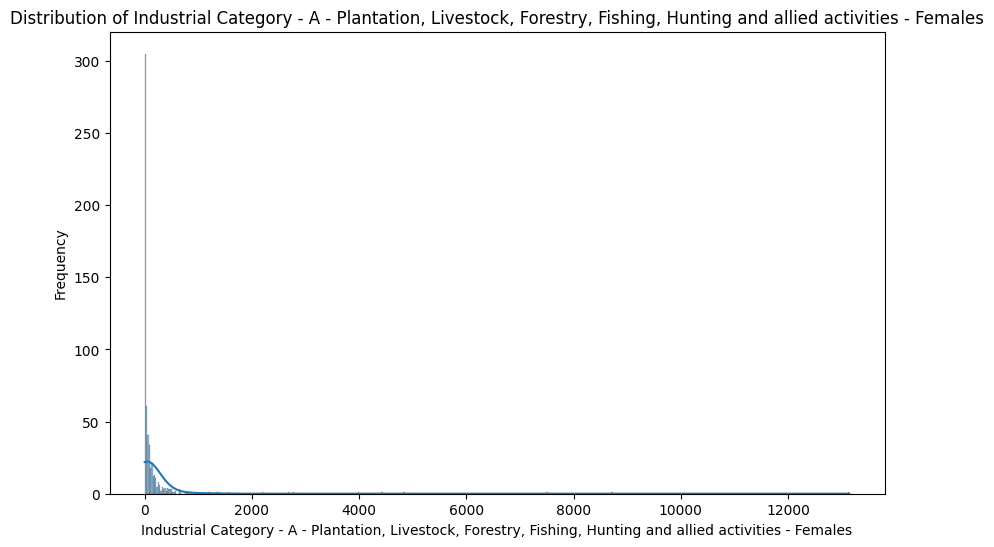


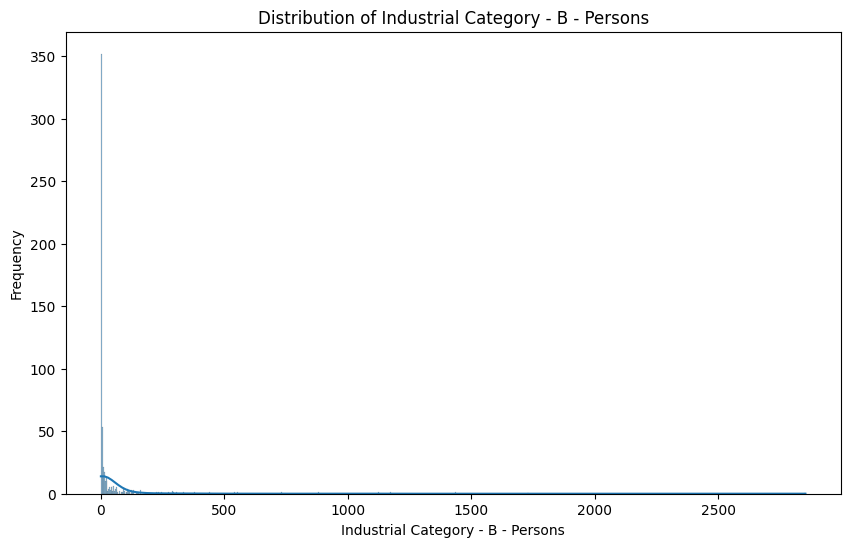


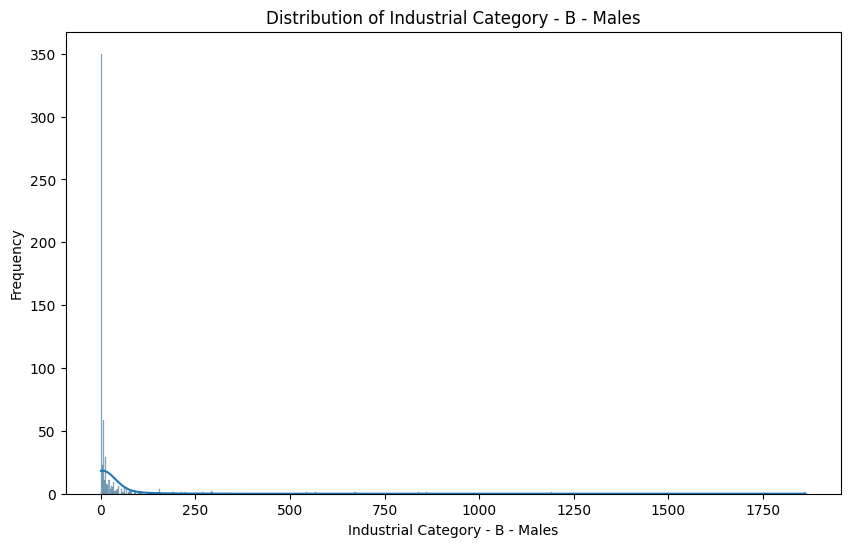


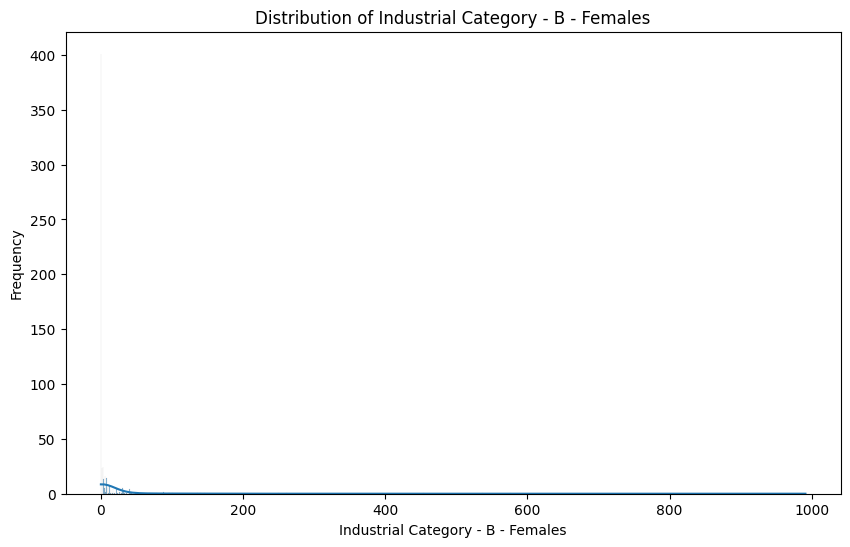
  


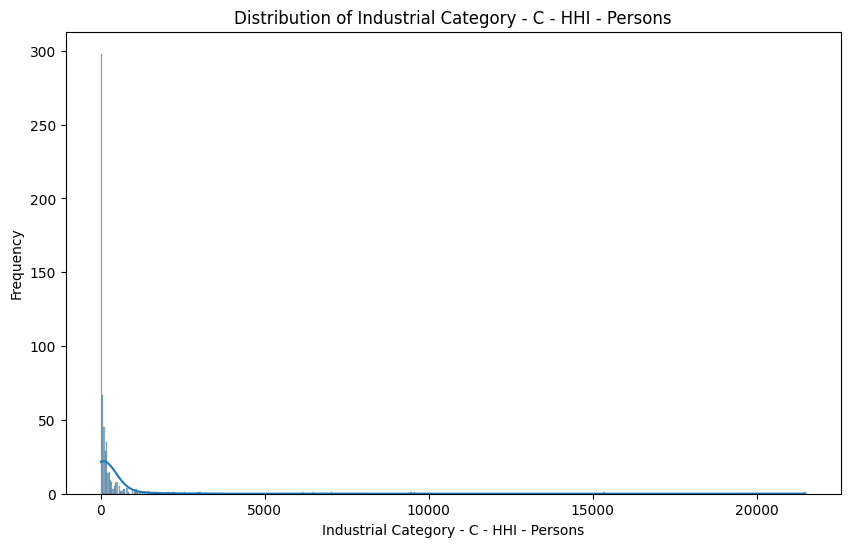


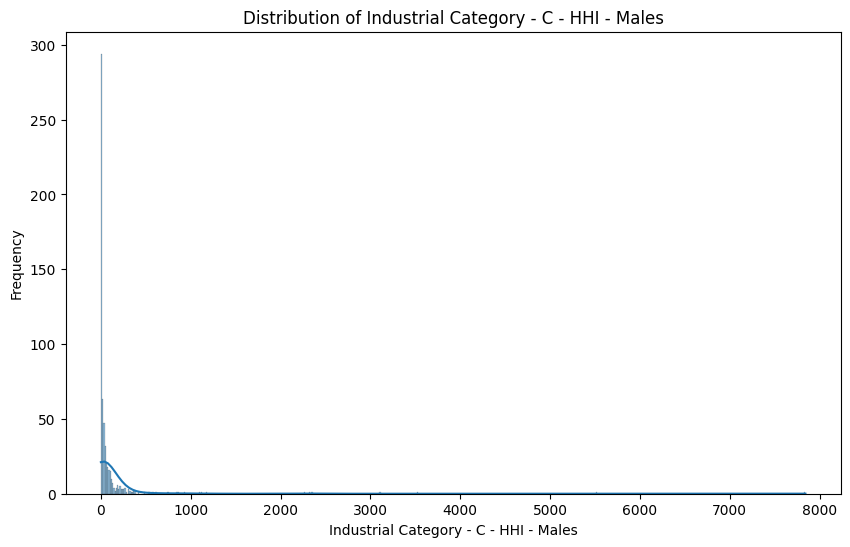


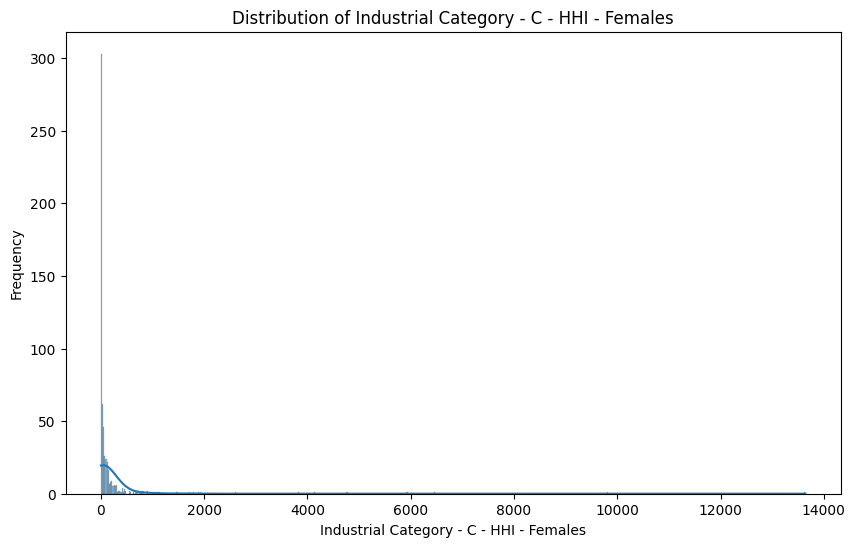


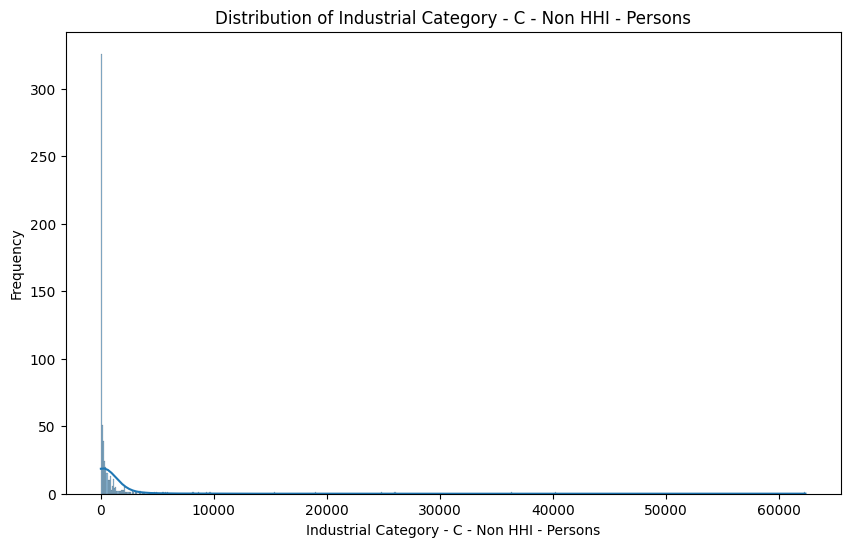


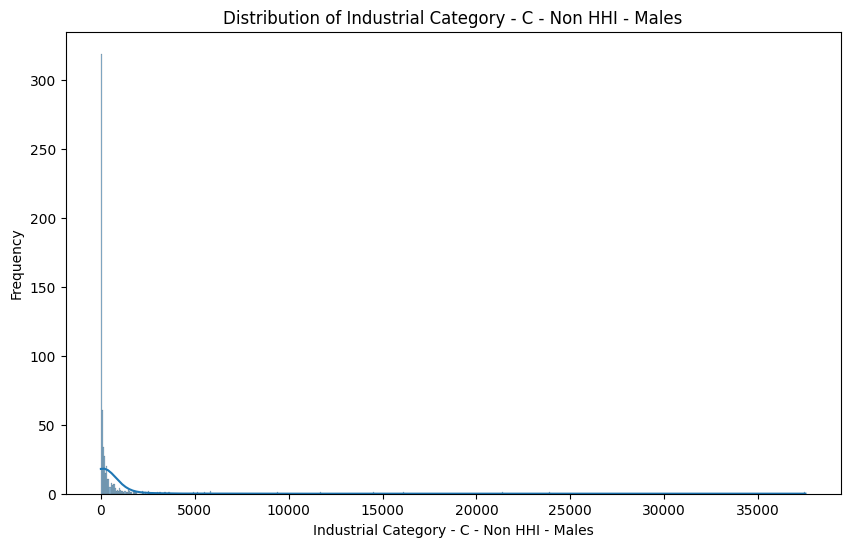


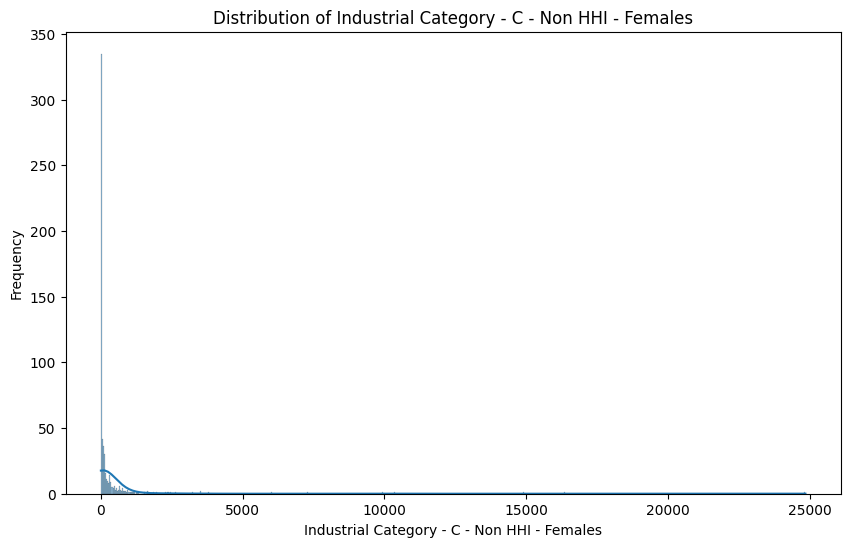


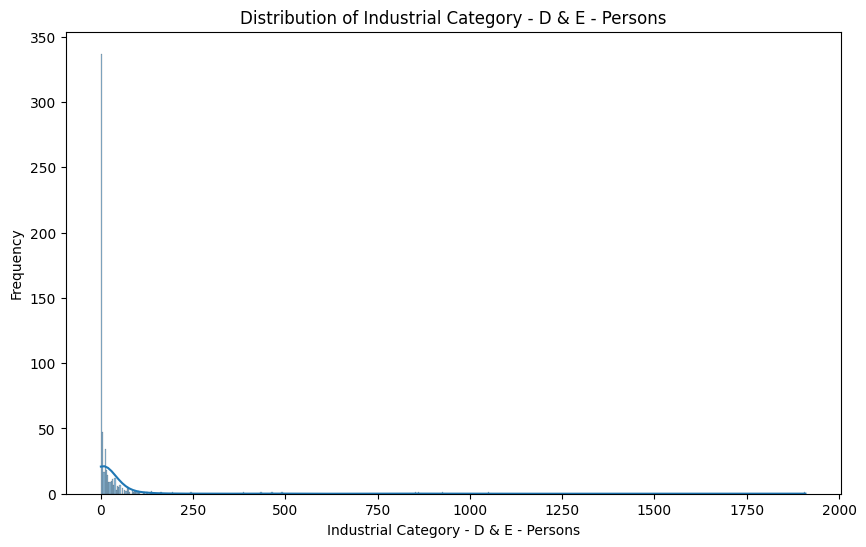
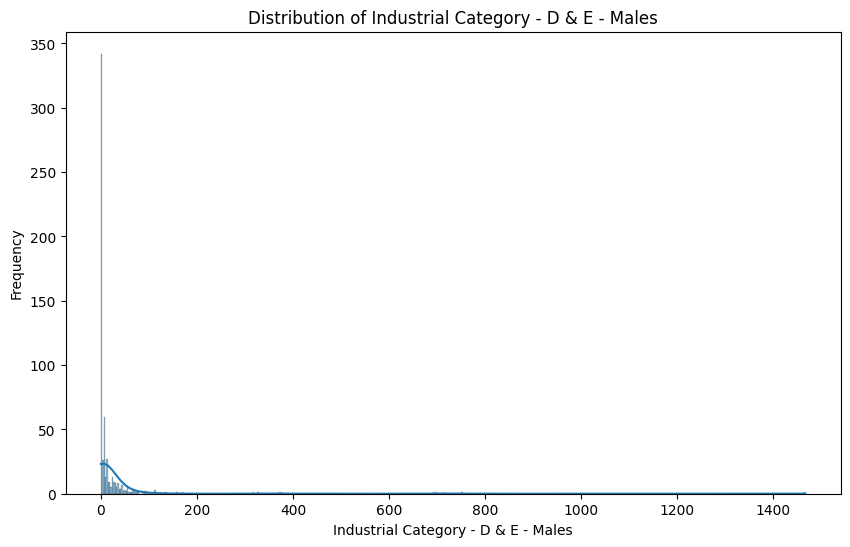


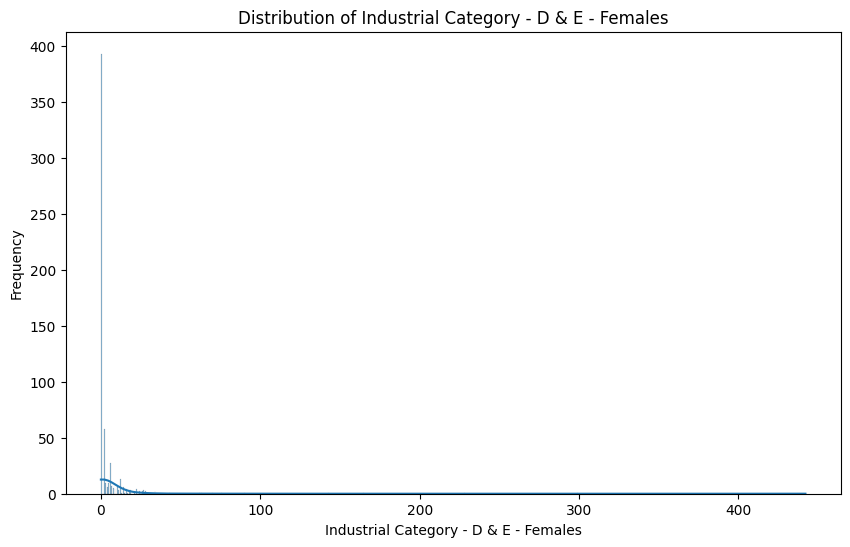


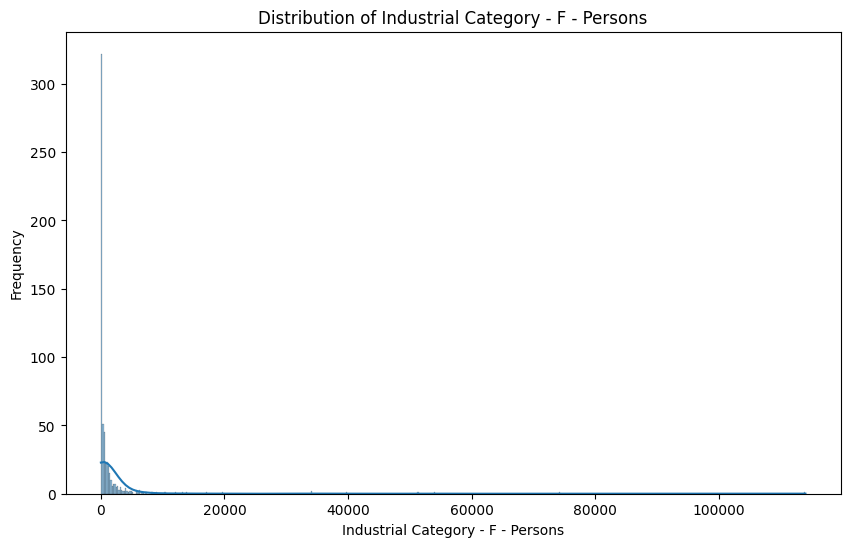


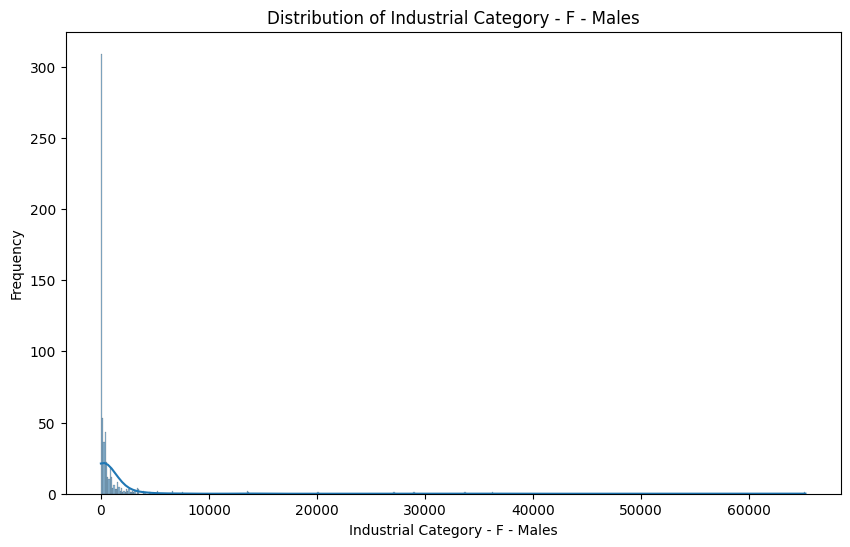


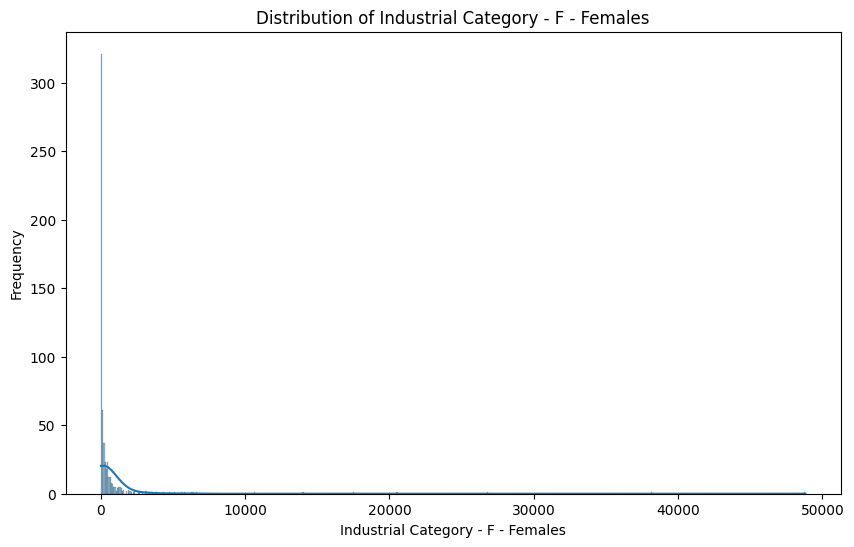


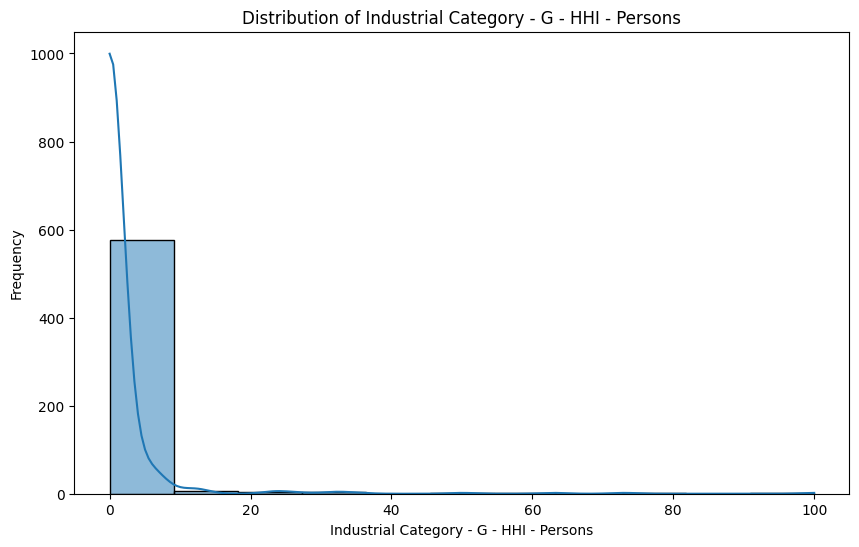
  


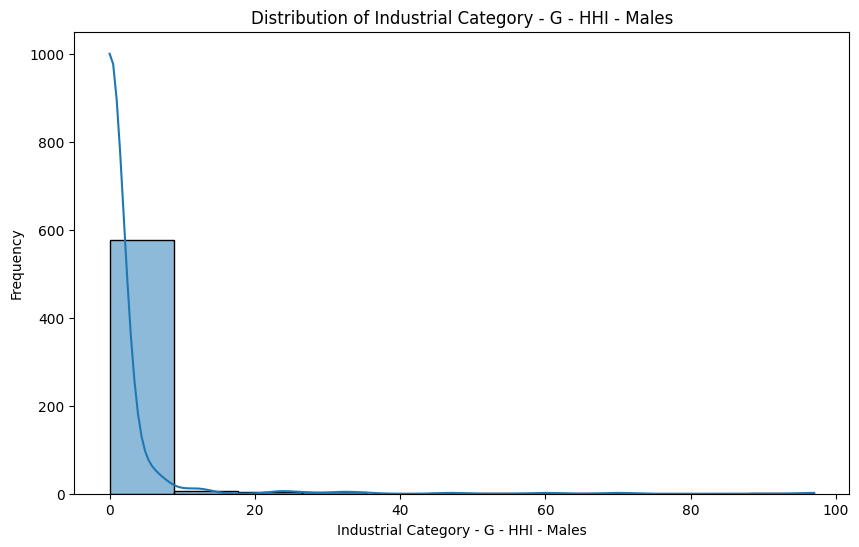


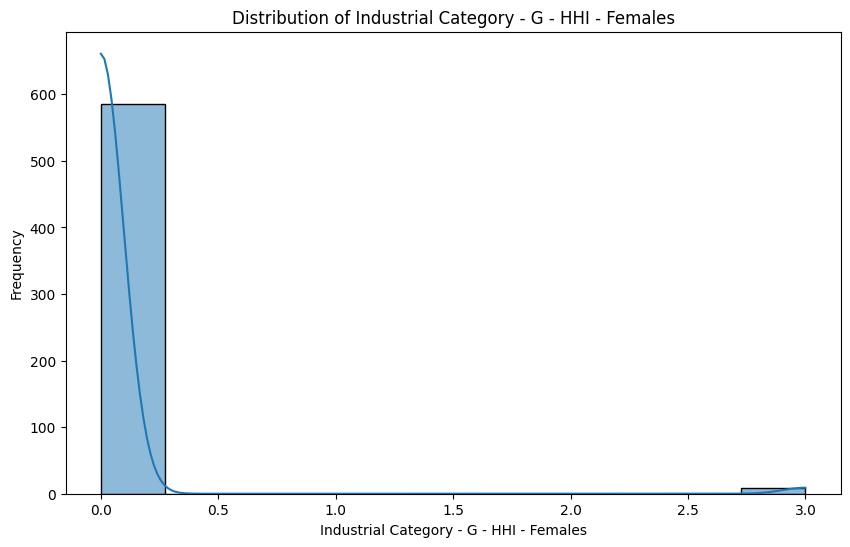


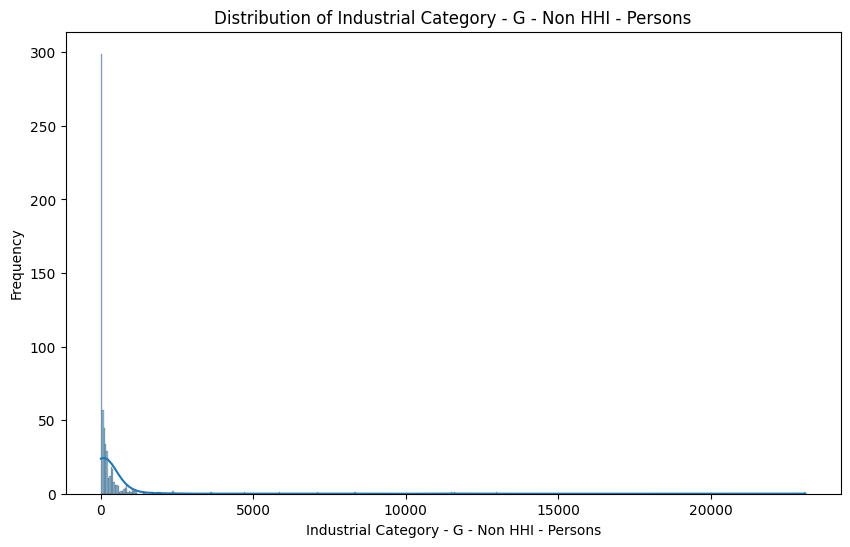


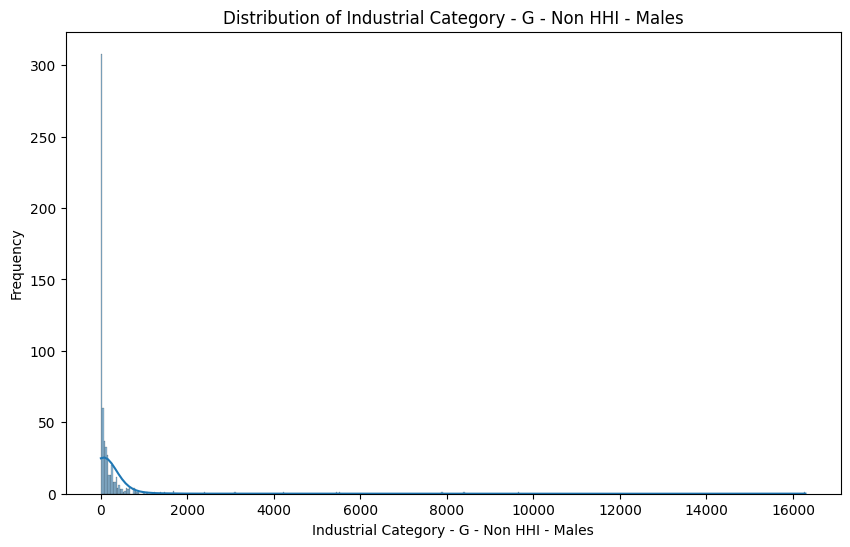


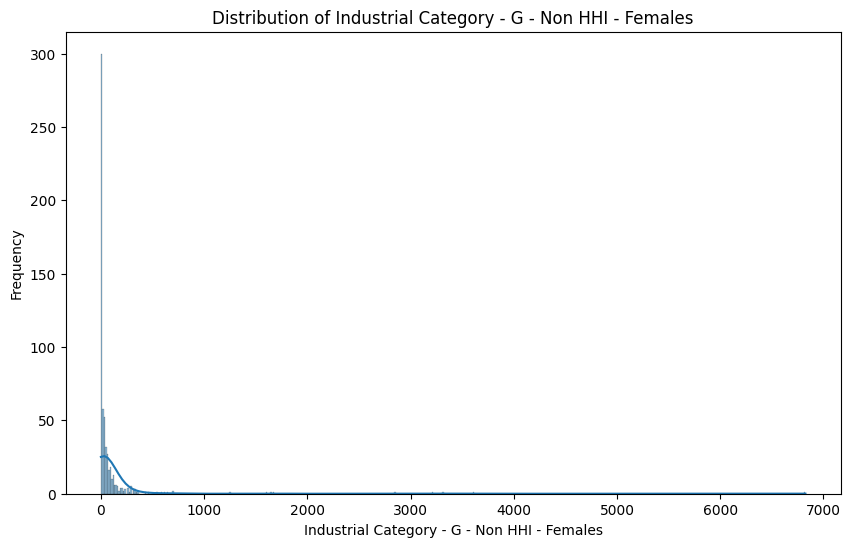


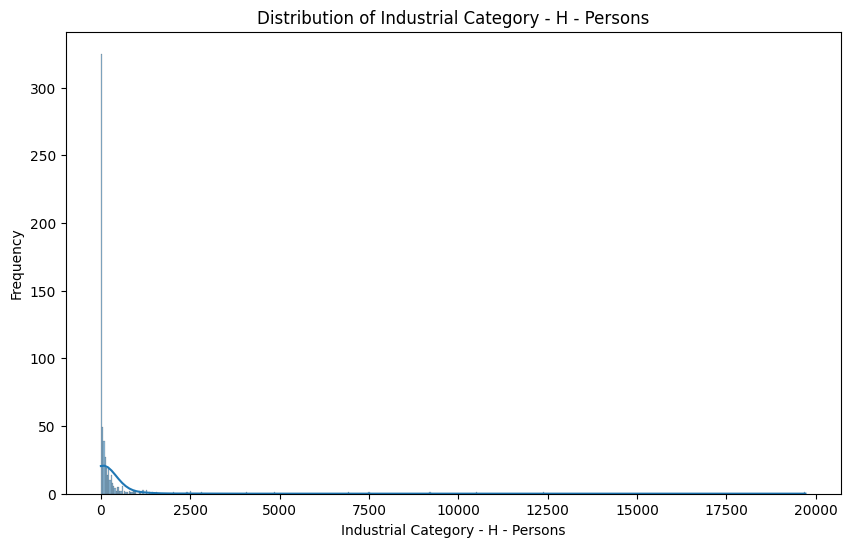


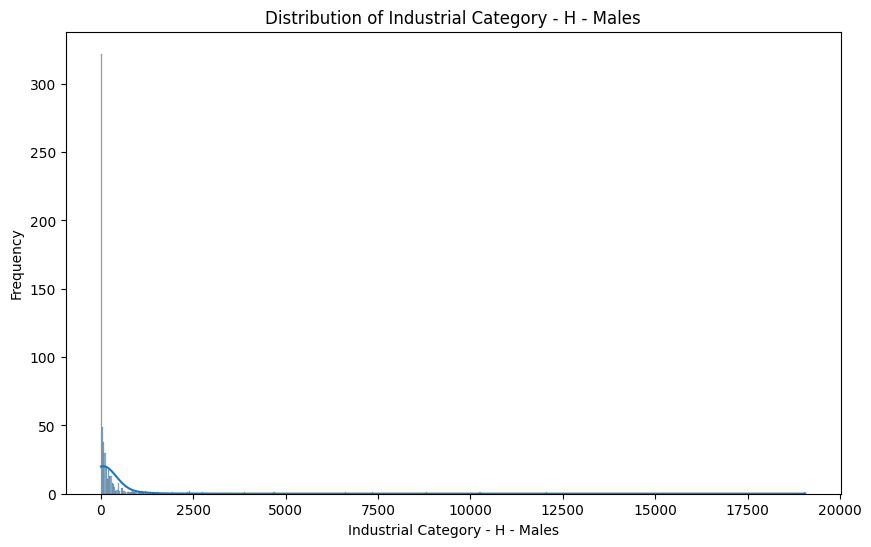


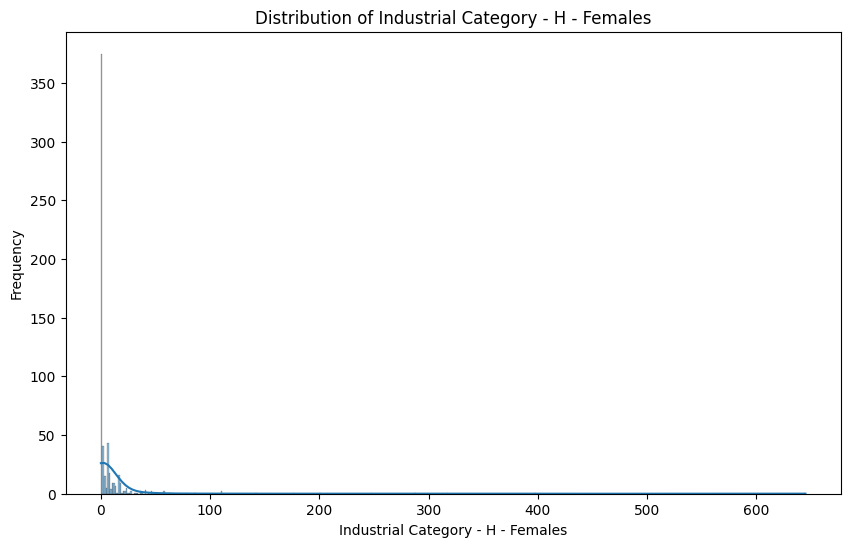


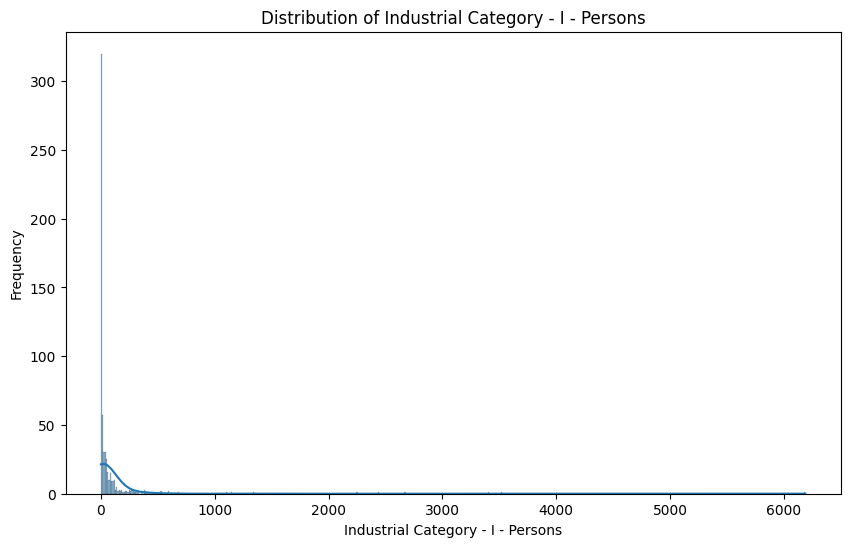


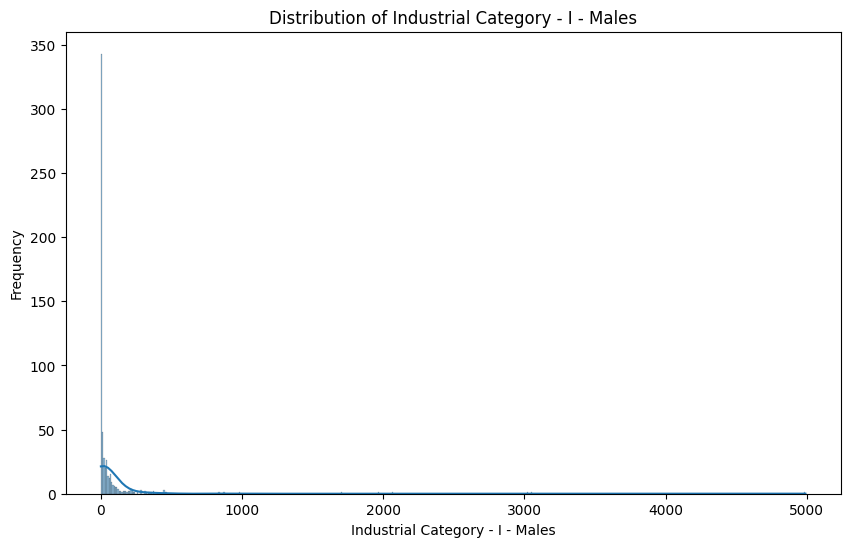


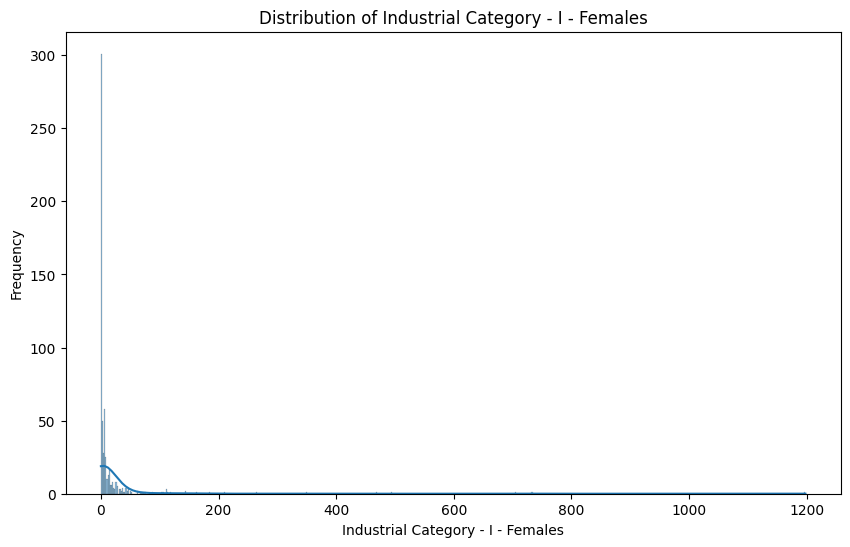


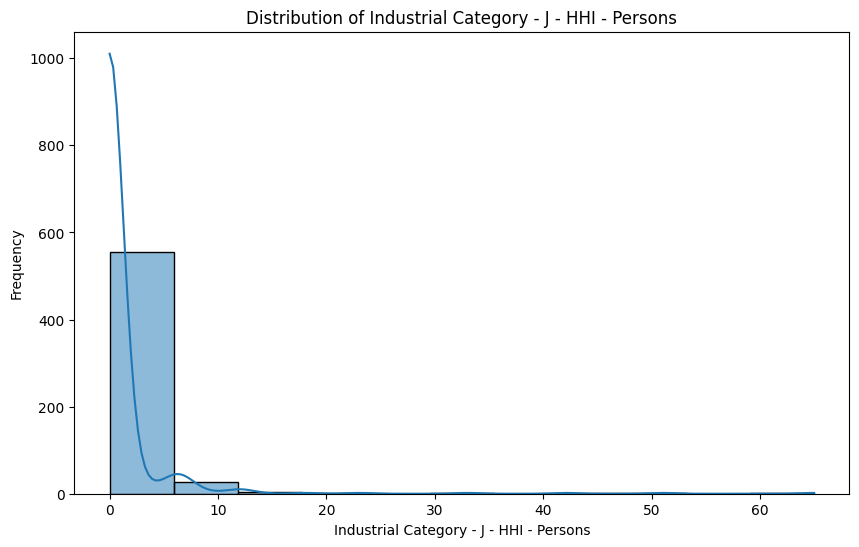


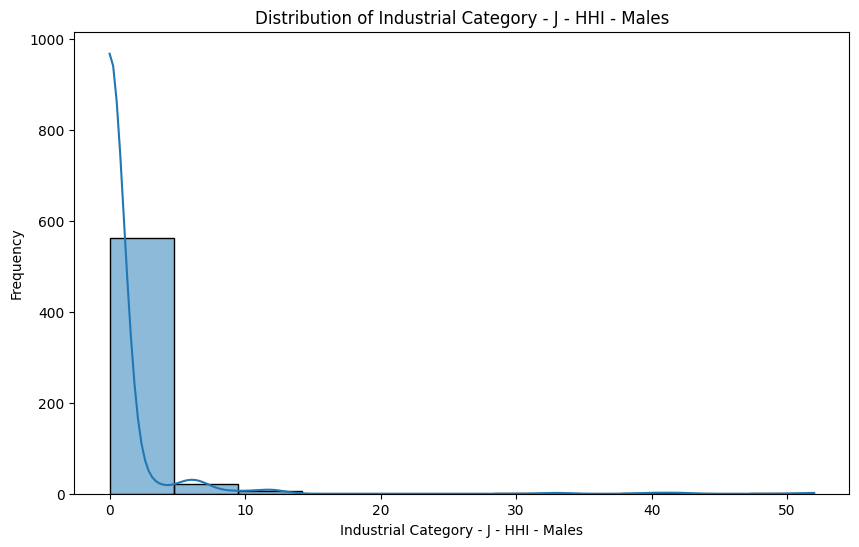


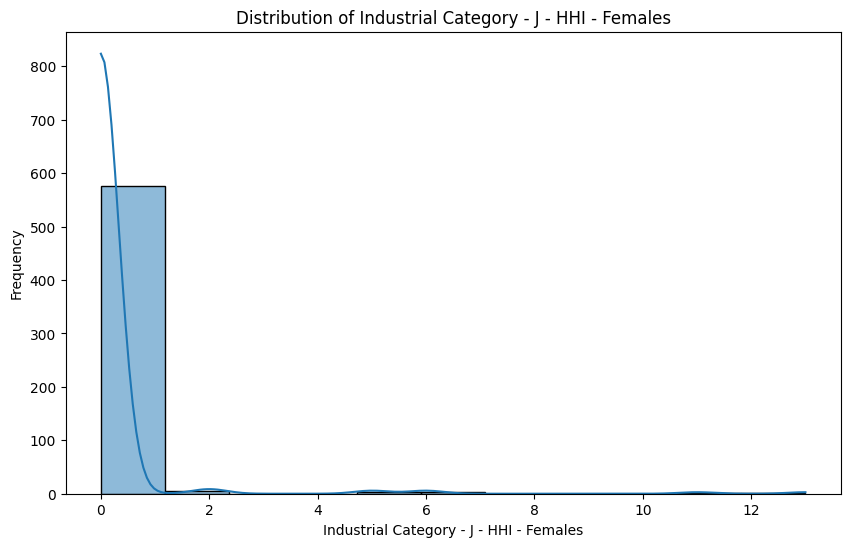


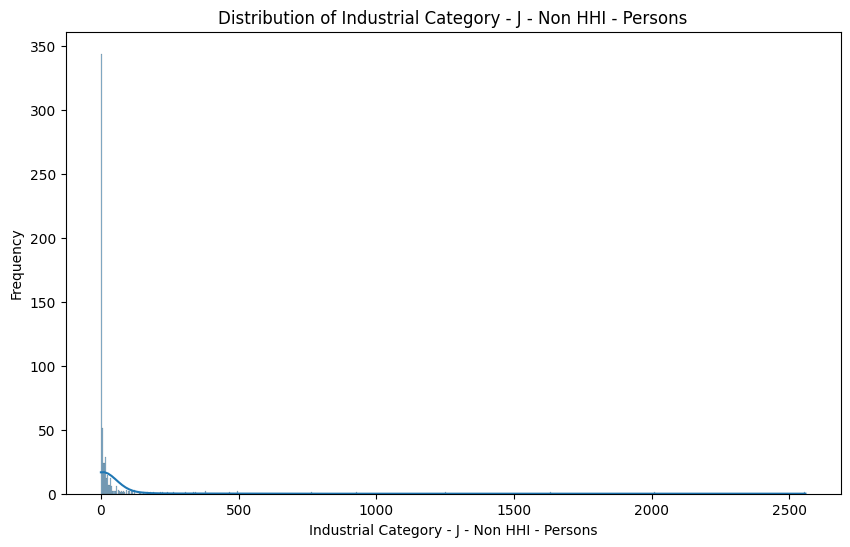


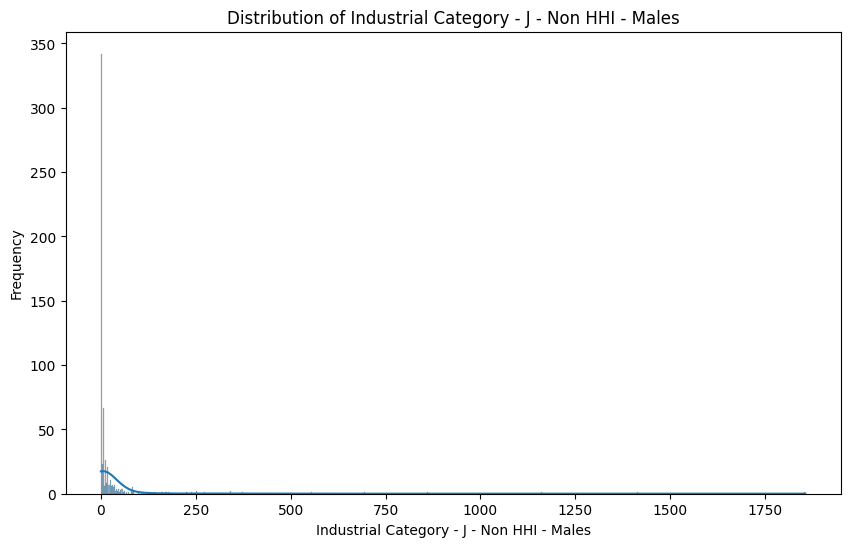


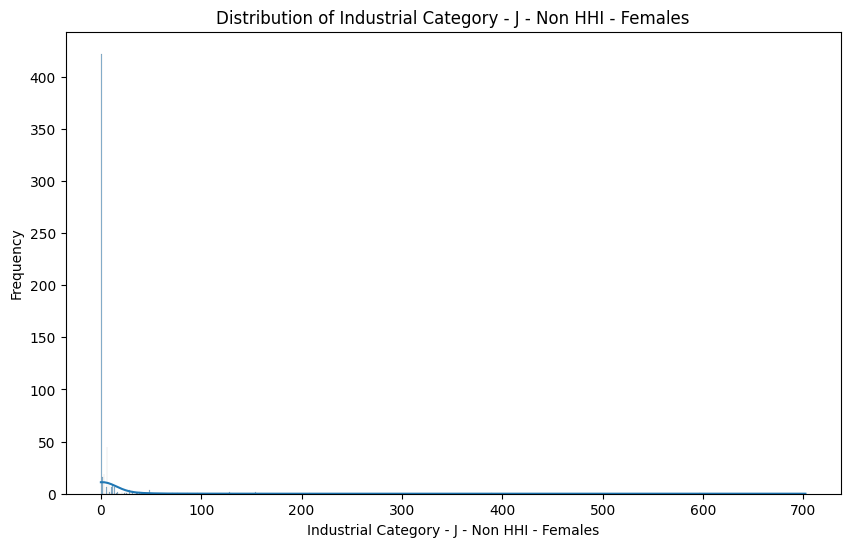


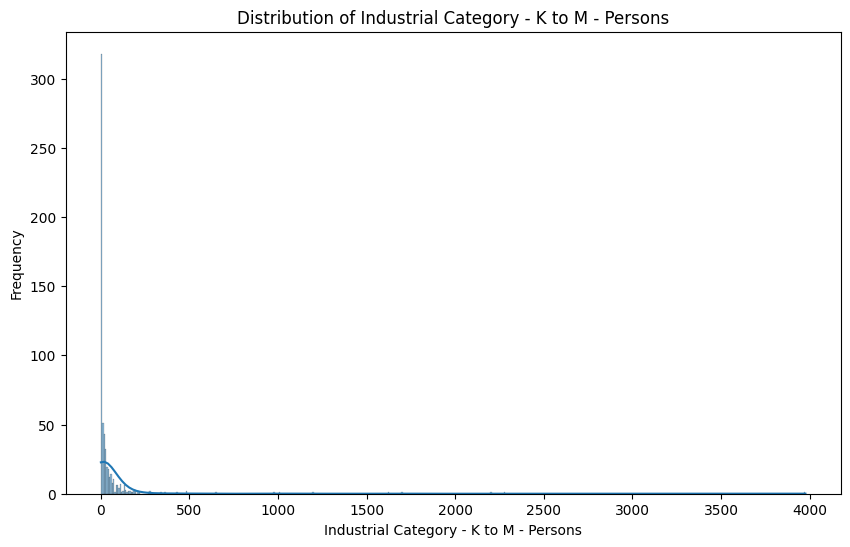


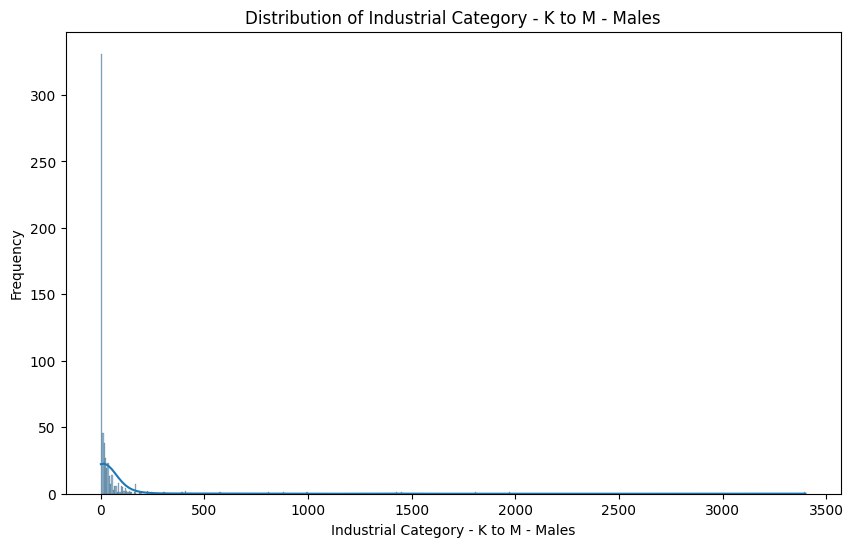


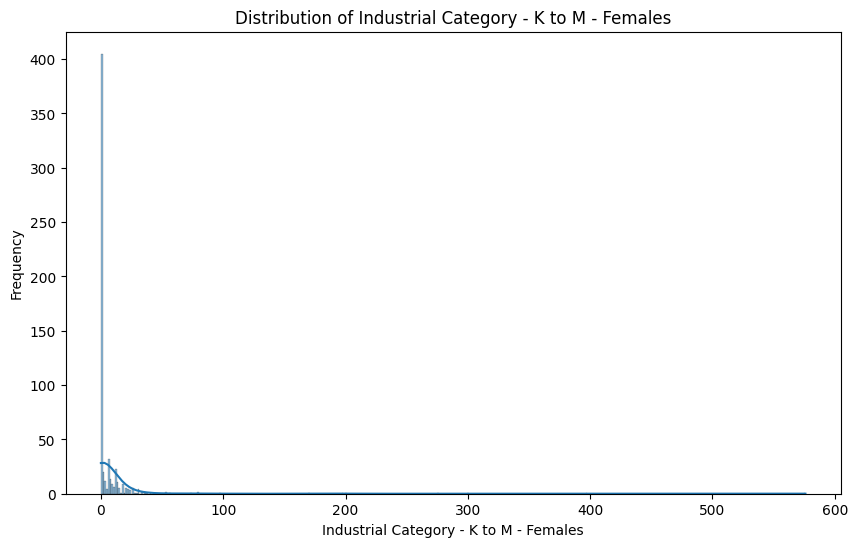


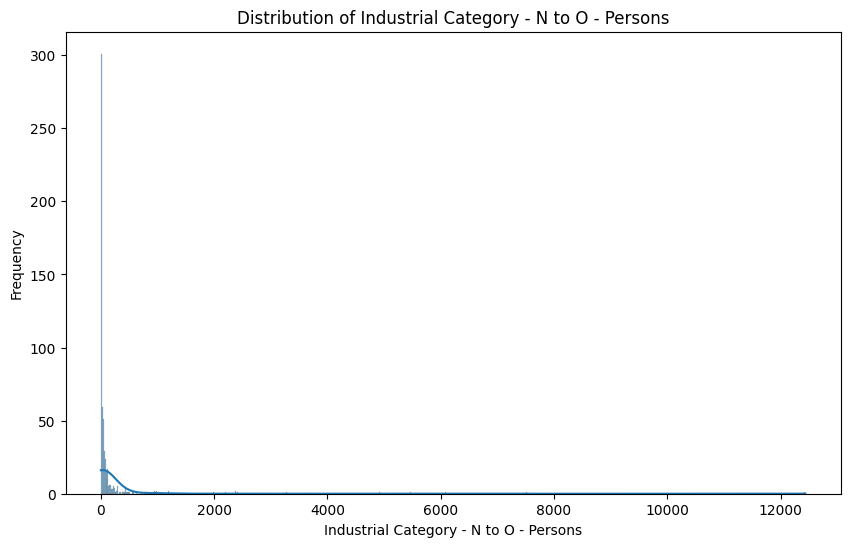


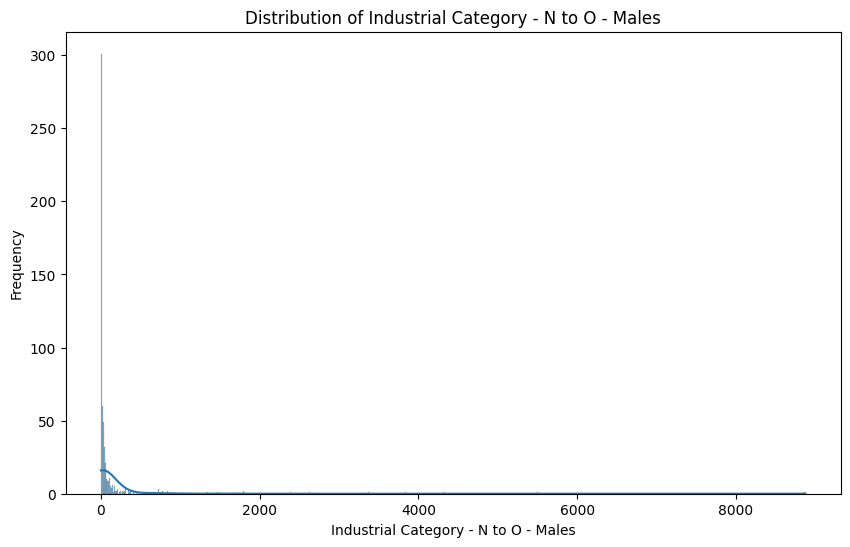


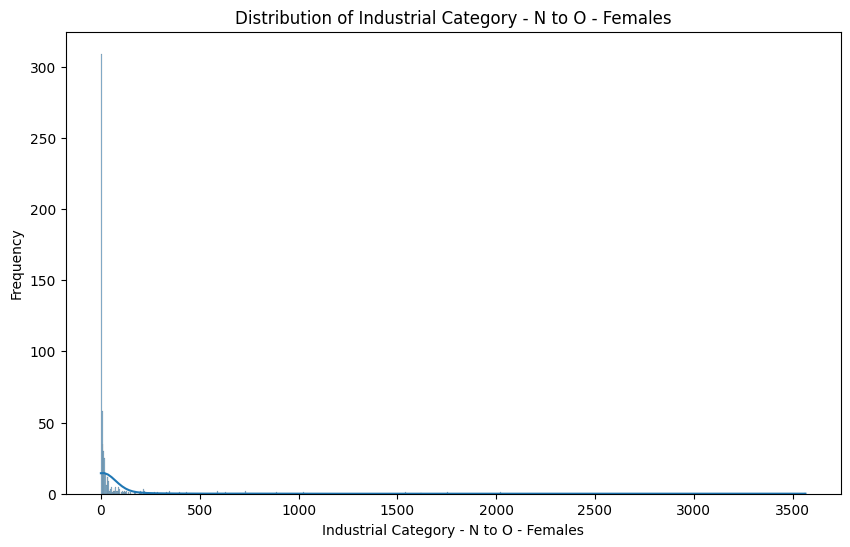


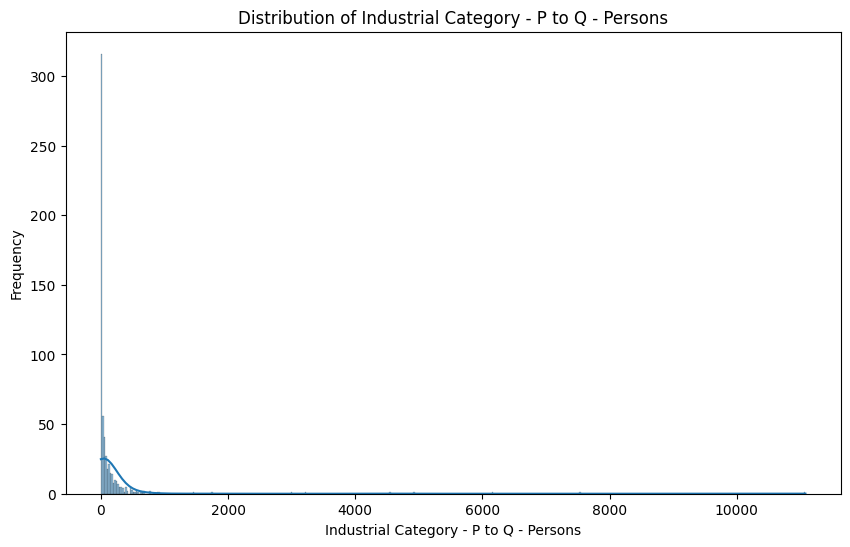


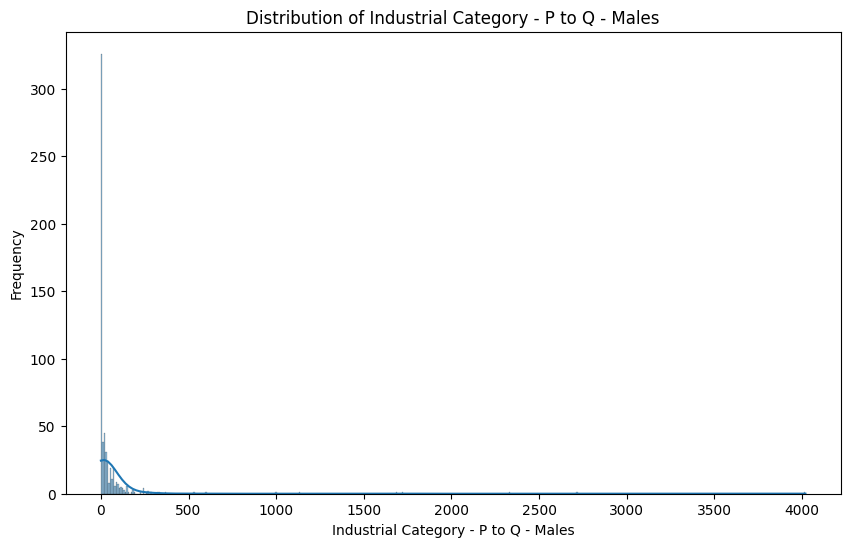


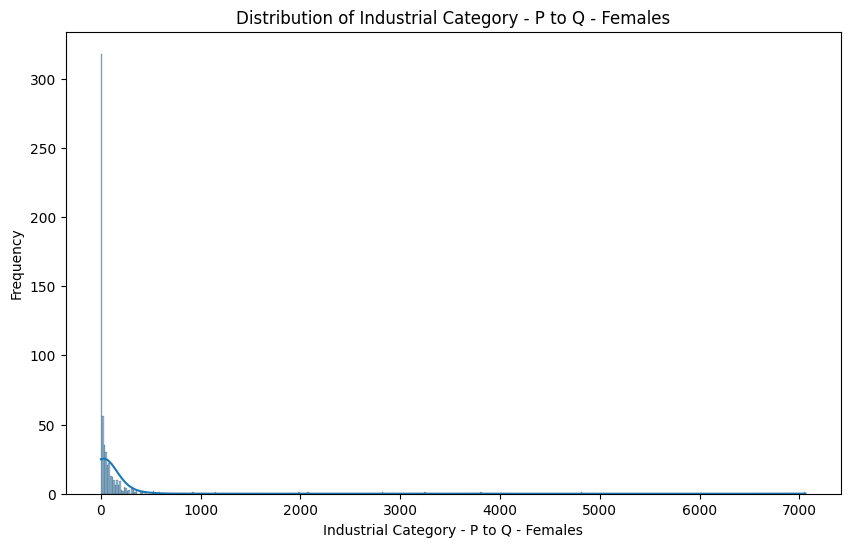


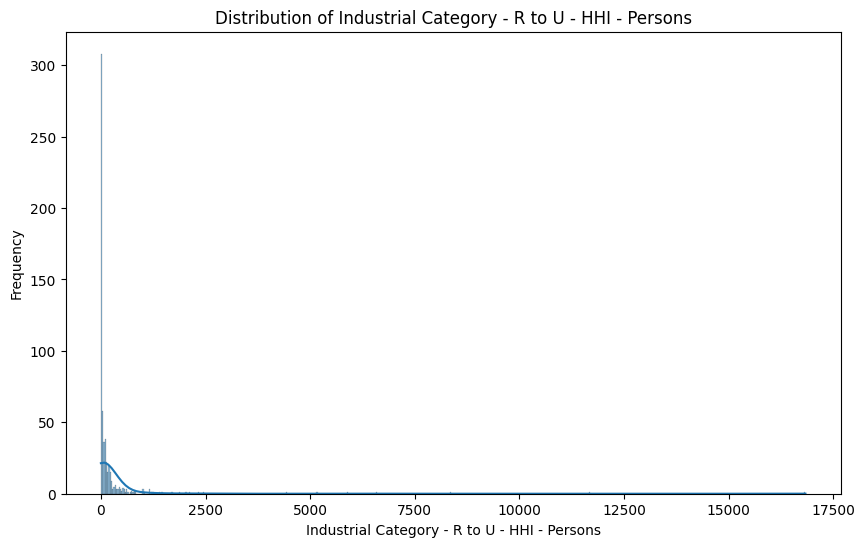


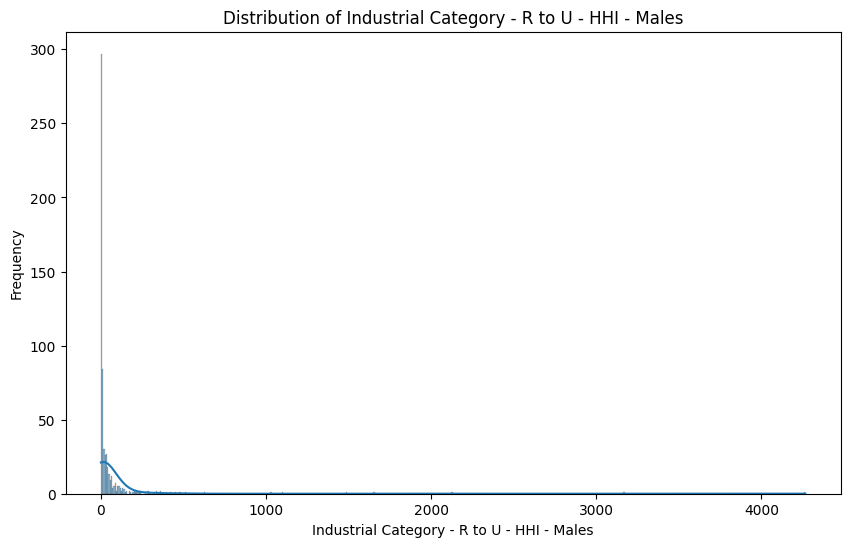


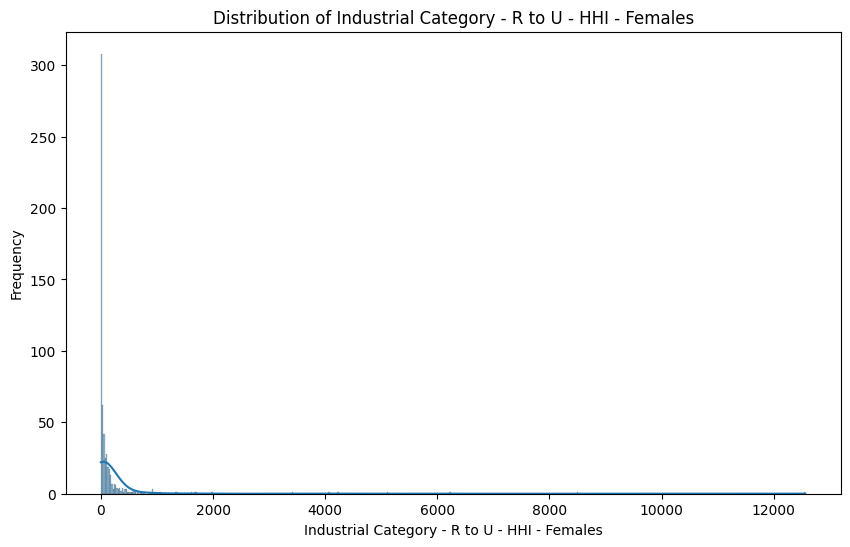


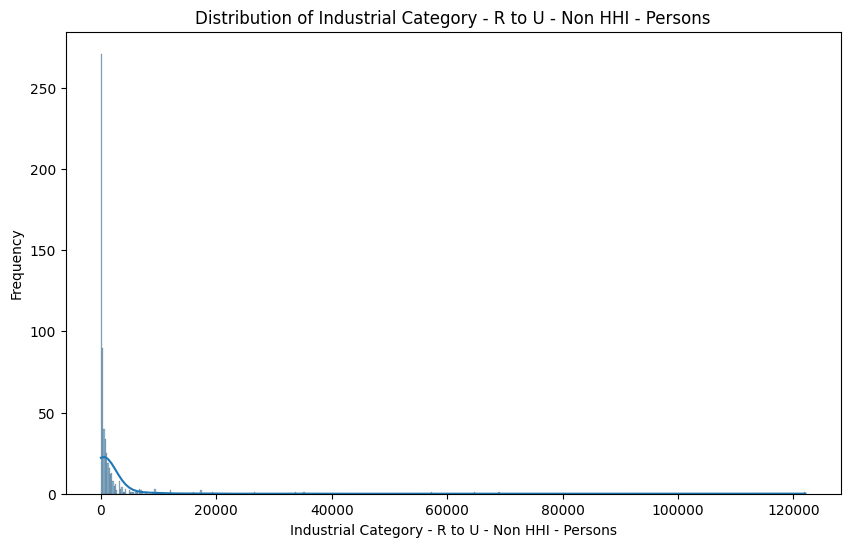


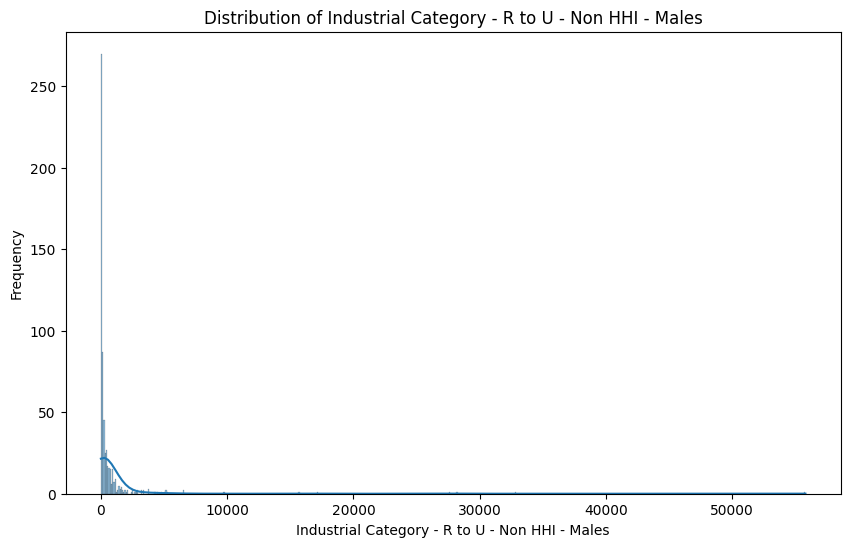


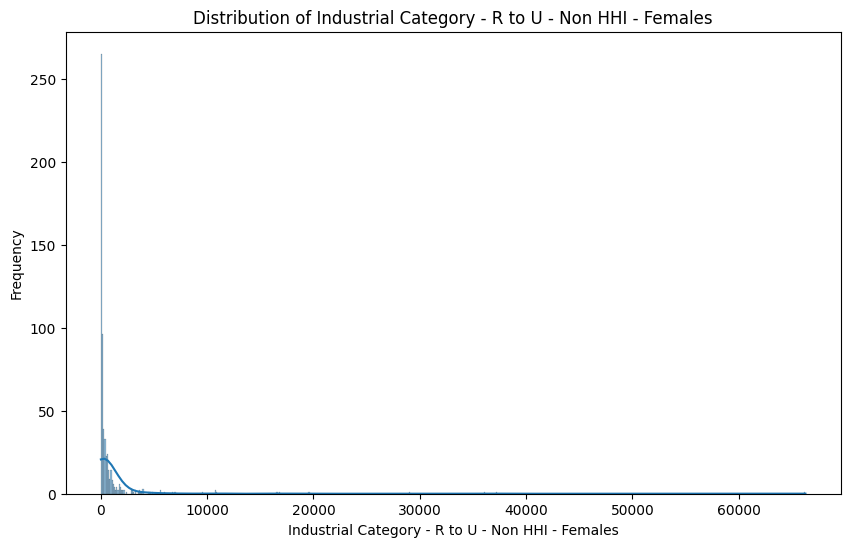












DataSet = DataSet[DataSet['Age group'].astype(str).str.strip() != '']

# Visualization

plt.figure(figsize=(12, 6))

sns.countplot(x='Area Name', hue='Age group', data=DataSet)

plt.xticks(rotation='vertical')

plt.xlabel('Area Name')

plt.ylabel('Count')

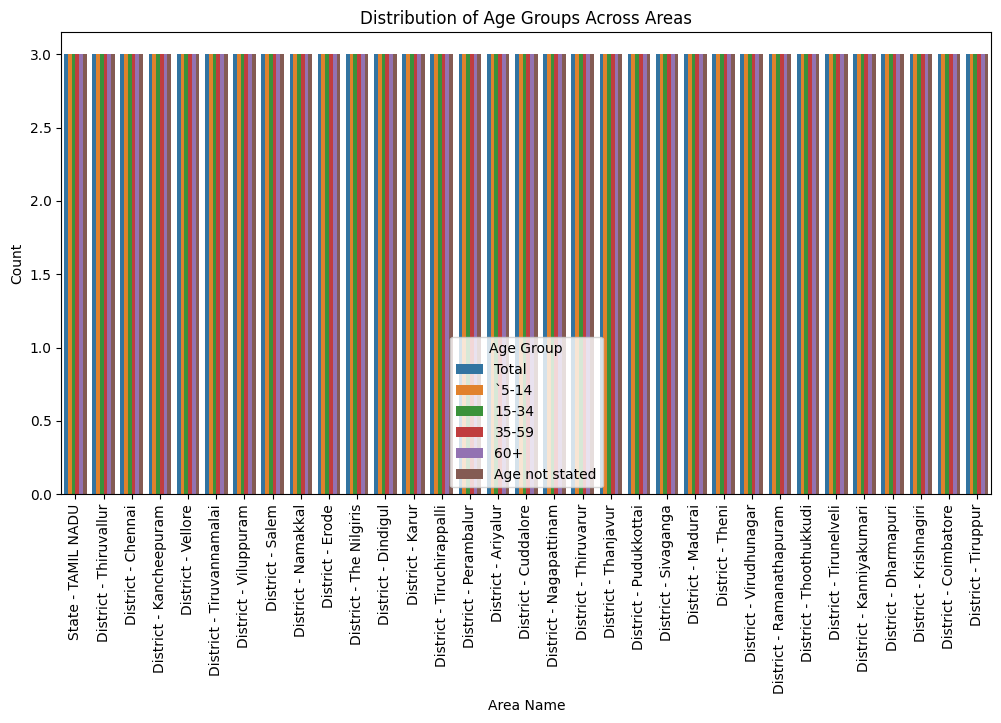
plt.title('Distribution of Age Groups Across Areas')

plt.legend(title='Age Group')

plt.show()

Explanation:

1. A bar chart is created with a figure size of 10x6, representing the number of persons who worked for 3-6 months categorized by age groups. The x-axis represents 'Age group,' the y-axis represents 'Number of Persons,' and the title is set to 'Worked for 3-6 Months by Age Group.' The x-axis labels are rotated by 45 degrees for better readability.
2. A loop iterates through the 'numerical\_columns' and visualizes the distributions of these columns using histograms and kernel density plots. For each column, a figure is created with a size of 10x6, a histogram and a kernel density plot are plotted using Seaborn, and the title, x-axis label, and y-axis label are appropriately set to describe the distribution of the specific column. This process visualizes the distributions for multiple columns in the dataset.
3. Finally, the 'DataSet' is filtered to exclude rows where the 'Age group' column is not equal to 'Total,' and these filtered results are stored back in 'DataSet.'

Output:  


# Heatmap for correlation matrix of numeric columns

correlation\_matrix = DataSet[numerical\_columns].corr()

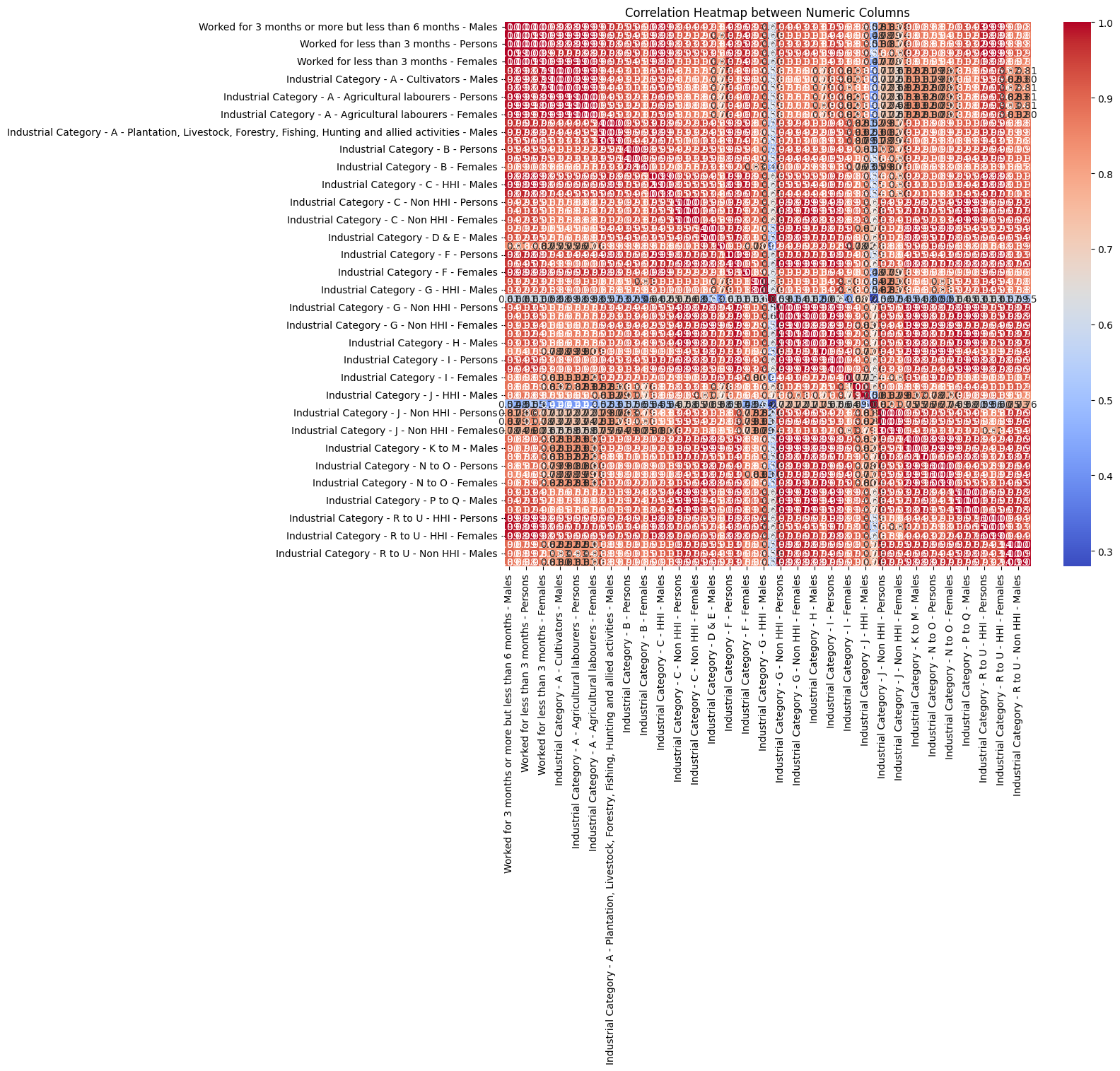
plt.figure(figsize=(12, 10))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap between Numeric Columns')

plt.show()

Output:



plt.figure(figsize=(10, 6))

sns.barplot(x='Age group', y='Worked for less than 3 months - Persons', data=DataSet, estimator=sum, ci=None, palette="viridis")

plt.title('Bar Plot between Categorical\_Column and Numeric\_Column')

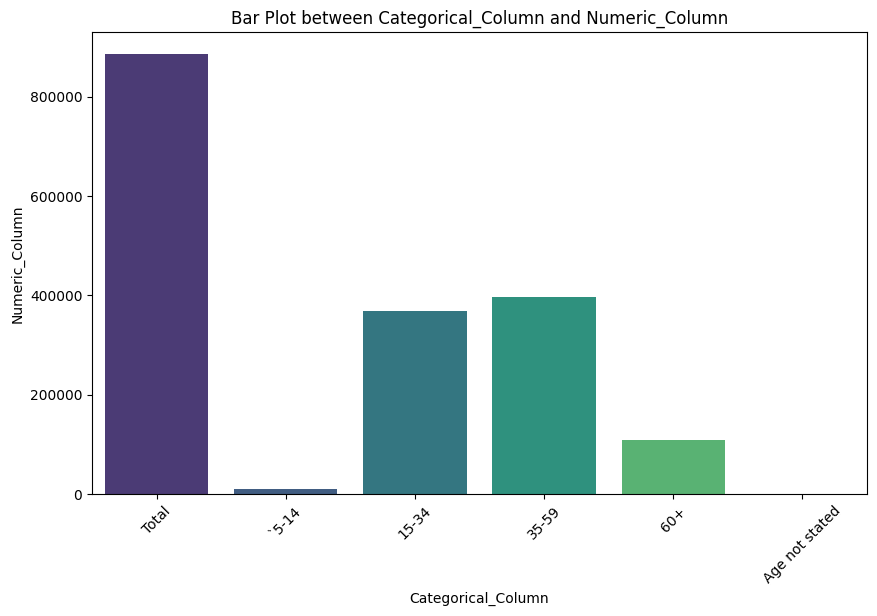
plt.xlabel('Categorical\_Column')

plt.ylabel('Numeric\_Column')

plt.xticks(rotation=45)

plt.show()

Output:



2)data aggregation and manipulation:

cities=pd.read\_csv("/content/Lat\_long\_cities\_TamilNadu.csv")

cities['Location'].value\_counts()

Output:

Adirampattinam 1

Ambasamudram 1

Tranquebar 1

Tondi 1

Tiruvallur 1

..

Kilakarai 1

Kayalpatnam 1

Kaveri R. 1

Kanchipuram 1

Vellore 1

Name: Location, Length: 112, dtype: int64

cities['Location'].unique()

DataSet['Area Name'] = DataSet['Area Name'].str.replace('District - ', '', regex=False)

DataSet['Area Name'].unique()

cities['Location'] = cities['Location'].str.strip()

DataSet['Area Name'] = DataSet['Area Name'].str.strip()

print("Unique locations in cities:", cities['Location'].unique())

print("Unique districts in DataSet:", DataSet['Area Name'].unique())

merge\_data = pd.merge(cities, DataSet, left\_on='Location', right\_on='Area Name', how='inner')

merge\_data

Merge\_data.columns

merge\_data.drop(['Unnamed: 1', 'Unnamed: 2','Unnamed: 4', 'Unnamed: 5'],axis=1)

Explanation:

1. It calculates and visualizes a heatmap representing the correlation matrix between numeric columns in the 'DataSet.' The correlation matrix is computed using the 'corr()' function, and a heatmap is created using Seaborn. The heatmap is annotated with correlation values and uses the 'coolwarm' color map. The title is set as 'Correlation Heatmap between Numeric Columns,' and the resulting heatmap is displayed.
2. A bar plot is generated with a figure size of 10x6. It displays a bar chart showing the sum of 'Worked for less than 3 months - Persons' for different 'Age group' categories. The x-axis represents 'Age group,' the y-axis represents 'Numeric\_Column,' and 'Categorical\_Column' is used as an estimator to calculate the sum. The x-axis labels are rotated by 45 degrees for readability.
3. A CSV file named 'Lat\_long\_cities\_TamilNadu.csv' is read into a DataFrame called 'cities.' It then calculates and displays the counts of unique values in the 'Location' column, showing the number of times each location appears in the dataset.
4. The unique values in the 'Location' column of the 'cities' DataFrame and the 'Area Name' column of the 'DataSet' are extracted and displayed.
5. Data cleaning and preparation steps include removing extra spaces and merging data between the 'cities' and 'DataSet' DataFrames based on the 'Location' and 'Area Name' columns, creating a new DataFrame called 'merge\_data.'
6. The 'Unnamed' columns (Unnamed: 1, Unnamed: 2, Unnamed: 4, Unnamed: 5) in the 'merge\_data' DataFrame are dropped, presumably for cleaning or preprocessing purposes.

# importing required modules

from zipfile import ZipFile

# specifying the zip file name

file\_name = "/content/tamil-nadu.zip"

# opening the zip file in READ mode

with ZipFile(file\_name, 'r') as zip:

# printing all the contents of the zip file

zip.printdir()

# extracting all the files

print('Extracting all the files now...')

zip.extractall()

print('Done!')

tamilnadu\_map = gpd.read\_file('/content/tamil-nadu.shp')

Explanation:

1. It imports the necessary module 'ZipFile' from the 'zipfile' library for handling zip files.
2. It specifies the name and path of the zip file as 'file\_name' ("/content/tamil-nadu.zip").
3. The code opens the zip file in read ('r') mode using a 'with' statement, which ensures proper handling and closure of the file.
4. Inside the 'with' block:
   * It prints the directory of all the contents within the zip file using 'zip.printdir()'. This helps you view the structure of the zip file.
   * It then extracts all the files within the zip archive to the current directory (the directory where the code is executed) using 'zip.extractall()'. This unzips all files present in the zip file to the current directory.
5. After extracting the files, the code proceeds to read a shapefile named 'tamil-nadu.shp' using 'geopandas' ('gpd') and assigns it to the 'tamilnadu\_map' variable. This shapefile likely contains geographic data for the state of Tamil Nadu.

3)Demographic Analysis:

# Convert Workers Distribution data to GeoDataFrame

geometry = gpd.points\_from\_xy(merge\_data['LON'], merge\_data['LAT'])

Workers\_Distribution = gpd.GeoDataFrame(merge\_data, geometry=geometry)

# Perform a spatial join to Marginal Workers Distribution data on Tamil Nadu's map

merged\_data = gpd.sjoin(tamilnadu\_map, Workers\_Distribution, how='inner', op='contains')

# Plotting the Workers Distribution

plt.figure(figsize=(12, 8))

tamilnadu\_map.plot(ax=plt.gca(), color='white', edgecolor='black')

ax = merged\_data.plot(column='Area Name', cmap='Greens', legend=True, markersize=50, ax=plt.gca())

plt.title('Marginal Workers in Tamil Nadu')

# Add a legend

from mpl\_toolkits.axes\_grid1 import make\_axes\_locatable

divider = make\_axes\_locatable(ax)

cax = divider.append\_axes("right", size="5%", pad=0.1)

merged\_data.plot(column='Category', cmap='coolwarm', legend=True, cax=cax, markersize=50)

cax.set\_title("Workers Area Distribution")

plt.show()

Explanation:

1. It converts data related to worker distribution into a GeoDataFrame. Geographic data in GeoPandas is represented using a 'geometry' column, which is created by combining longitude ('LON') and latitude ('LAT') columns from the 'merge\_data' DataFrame. The result is stored in a new GeoDataFrame called 'Workers\_Distribution'.
2. A spatial join operation is conducted by overlaying the 'Workers\_Distribution' data onto Tamil Nadu's geographic map ('tamilnadu\_map'). The 'how' parameter is set to 'inner' to retain only the areas where the data intersects or contains geographic features from the map.
3. It creates a plot with a figsize of 12x8. The code overlays the base map of Tamil Nadu using 'tamilnadu\_map' and colors it white with black edges. Subsequently, it plots the 'merged\_data' GeoDataFrame with different colors based on the 'Area Name' column, and it adds markers with a size of 50.
4. The code sets the title of the plot as 'Marginal Workers in Tamil Nadu'.
5. A legend is added to the plot to explain the 'Category' column. The 'make\_axes\_locatable' function is used to create a legend axis to the right of the main plot.
6. Finally, it displays the complete plot with the base map, markers, and legend to visualize the distribution of marginal workers in Tamil Nadu, with different areas represented by colors and categories explained in the legend.

Output:  
