Data Analysis Report

Title

ALICE Population's data analysis

Author

Phuc "William" Nguyen

Date

Jul 8, 2024

Table of Contents

1. Introduction
2. Data Collection
3. Data Cleaning and Preparation
4. Exploratory Data Analysis (EDA)
5. Data Analysis and Findings
6. Conclusion
7. Recommendations
8. References
9. Appendices

---

1. Introduction

- **Background:** Conduct research on the ALICE population in Ohio to gain a comprehensive understanding of their current situation. This knowledge will help in developing a product that is both beneficial for ALICE households and sustainable for United Way. The questions to address are:
    - What is the percentage of households at or under the ALICE threshold in Ohio? What is the trend in the past 10 years?

    - What is the percentage of households at or under the ALICE threshold of sub counties in Franklin, Delaware, and Licking? Does the rate stay consistent in all the sub counties of these counties?
    - What is the difference between ALICE thresholds and the survival and stability budget?
- **Objective:** Uncover insights on the poverty rate in Ohio, with a particular focus on Franklin, Delaware, and Licking counties. The analysis aims to explore the disparities in poverty rates within these counties, examine the trends in the ALICE (Asset Limited, Income Constrained, Employed) thresholds over the past decade, and identify the gap between the ALICE threshold and the survival and stability budgets. These insights will help inform policies and programs aimed at reducing economic disparities and supporting financially constrained households.
- **Scope:**
    - Geographic Focus:
        - Primary Focus: Franklin, Delaware, and Licking counties in Ohio.
        - State-Level Context: Comparative analysis with statewide data to provide broader context.
    - Key Metrics:
        - Poverty Rate: Analysis of the poverty rate in each county and its sub-counties.
        - ALICE Thresholds: Examination of the ALICE thresholds over the past decade, including the highest and lowest thresholds.
        - Survival and Stability Budgets: Comparison between the ALICE thresholds and the survival and stability budgets for various household types.
    - Time Frame:
        - Historical Data: Analysis of data from the past 10 years to identify trends and changes in poverty rates and ALICE thresholds.
        - Current Data: Utilization of the most recent data available (up to 2022) to assess the current economic conditions.
    - Household Types:
        - Single Adults: Analysis of economic conditions for single adult households.
        - Families with Children: Examination of the financial needs and thresholds for households with one or more children.
        - Seniors: Analysis of the economic challenges faced by senior households.
    - Comparative Analysis:
        - County-Level Comparison: Comparison of economic conditions across Franklin, Delaware, and Licking counties.
        - Sub-County Analysis: Examination of regional disparities within each county to identify areas with the highest needs.

2. Data Collection

- **Data Sources:** From the website of ALICE, specifically from the data published in the 2024 report in the “Research Center” section of Ohio
- **Data Description:** Data is in xlsx format, data types are string, int, and float.
- **Collection Method:** Collected via API of ALICE website

3. Data Cleaning and Preparation

- **Data Cleaning:** Checked for missing values, wrong data types, duplicates, data inconsistency.
- **Data Preparation:** Convert xlsx files into csv files that can be used by Python pandas library, check for data type consistency; normalization and transformation in 6 data sheets that are in the field of Survival Budget and the Stability Budget

4. Exploratory Data Analysis (EDA)

- In 2014, Athens County had the highest poverty rate at 31%, while Delaware County had the lowest at 4%.
- Between 2010 and 2022, the poverty rates in Franklin, Delaware, and Licking counties remained relatively stable, with changes not exceeding 3%.
- During this period, the ALICE threshold for households under 65 ranged from a maximum of approximately $68,000 to a minimum of $30,000.
- For households aged 65 and over, the threshold ranged from a high of around $55,000 to a low of $25,000.

5. Data Analysis and Findings

- **Analysis Methods:** Descriptive analysis as the main method, focusing on understanding the household types distribution in counties with the highest and lowest % of under ALICE Threshold, then move to analyzing the household distribution in Franklin, Delaware, and Licking counties along with the differences between the 3 counties's ALICE thresholds and the survival and stability budget provided on the ALICE website.
- **Key Findings:**
    - The standard deviation of poverty rate in a county is high with the difference in the sub counties with the lowest rate and the counties with the highest is 40%
    - Delaware and Franklin is in the top 3 counties with the highest ALICE threshold average, averaging 43000$ with the highest threshold ~68000$ and the lowest threshold is 25000$ in 10 year period
    - Licking county is in the top 10 the highest ALICE threshold in 2022, with the threshold of under 65 and at least 65 are over 50000$ which is more than the highest threshold average of the past 10 years
    - For 2 adults wanting to provide good childcare for 2 children, in the 3 counties Franklin, Delaware, and Licking they would need an income of at least 130000$, which is ~111725$ more than the 3 counties's threshold
- **Interpretation:**
    - *High Standard Deviation in Poverty Rates*: While some areas may have relatively low poverty rates, others are experiencing significantly higher levels of poverty. This disparity could point to uneven distribution of resources, opportunities, and services within the county.

- *High ALICE Thresholds in Delaware and Franklin Counties*: These counties have experienced significant economic pressures, requiring higher incomes to meet basic needs. This high threshold average reflects the elevated cost of living and economic demands placed on residents, further emphasizing the financial challenges faced by many households
- *Licking County's Rising Thresholds*: Such a sharp increase in the ALICE thresholds indicates a rapid rise in the cost of living in Licking County, which could be driven by factors such as inflation, housing costs, and other economic conditions. This trend suggests that more households are likely to struggle to make ends meet, potentially leading to increased financial stress and insecurity.
- *Income Requirements for Families with Children*: This vast discrepancy highlights the inadequacy of current income levels to support family needs, particularly in terms of childcare. It underscores the challenges families face in balancing work and childcare responsibilities, which could have broader implications for workforce participation, child development, and overall family well-being.

=> These findings underscore the economic challenges and disparities faced by households in these counties. The high standard deviation in poverty rates within counties suggests a need for targeted interventions to address inequality and support the most vulnerable populations. The high ALICE thresholds indicate that even households above the poverty line are struggling to meet basic needs, necessitating policies and programs that address the rising cost of living.

=> The findings highlight the need for:

- *Targeted Financial Assistance*: Programs that provide financial support to households struggling to meet the high cost of living, particularly in counties with high ALICE thresholds.
- *Affordable Childcare*: Policies that make childcare more affordable, reducing the financial burden on families and enabling better economic stability.
- *Economic Development*: Initiatives that promote equitable economic development within counties to reduce disparities in poverty rates and improve overall living conditions.
- *Comprehensive Support Systems*: Developing comprehensive support systems that address the varied needs of households, including housing, healthcare, education, and employment opportunities.

6. Conclusion

- **Summary:** Summarize the key points from your analysis.
  - *High Standard Deviation in Poverty Rates*:
    - There is a 40% difference in poverty rates between sub-counties with the lowest and highest rates, indicating significant disparities within counties.
  - *ALICE Thresholds in Delaware and Franklin Counties*:

        - These counties are among the top three with the highest average ALICE thresholds over the past decade, averaging $43,000, with thresholds ranging from $25,000 to $68,000.
    - *Rising Thresholds in Licking County*:
        - In 2022, Licking County had thresholds exceeding $50,000 for households under 65 and over 65, which is higher than the past decade's highest average threshold.
    - *Income Requirements for Families*:
        - Families in Franklin, Delaware, and Licking counties need at least $130,000 to provide good childcare for two children, which is significantly higher than the average ALICE threshold by approximately $111,725.

- **Implications:** Discuss the implications of your findings. How do they address the problem or question you started with?
    - *Economic Disparities*: The significant variation in poverty rates within counties suggests the need for targeted interventions to support economically disadvantaged areas.
    - *Cost of Living Pressures*: High ALICE thresholds indicate that even above-poverty-line households struggle to meet basic needs, highlighting the need for policies to address the rising cost of living.
    - *Family Financial Stress*: The substantial income required for adequate child care underscores the importance of making childcare more affordable and accessible.
    - *Support Systems*: Comprehensive support systems are necessary to address the diverse needs of households, including financial assistance, affordable housing, and accessible healthcare.

  => Thus, it answers the questions:

    - **Percentage of Households at or Under the ALICE Threshold in Ohio:**
        - *Trend Over the Past 10 Years*: The trend data over the past 10 years can be analyzed to understand how the percentage of households at or under the ALICE threshold has changed. This requires accessing historical data on ALICE thresholds and household incomes in Ohio.
    - **Percentage of Households in Sub-Counties:**
        - *Consistency Across Sub-Counties*: Analyzing the data for sub-counties in Franklin, Delaware, and Licking will show if the rate of households at or under the ALICE threshold is consistent across these areas. This analysis can reveal regional economic disparities within the counties.
    - **Difference Between ALICE Thresholds and Budgets:**
        - *Survival vs. Stability Budgets*: Comparing the ALICE thresholds with the survival and stability budgets will highlight the gap between the minimum required to meet basic needs and the income needed for long-term stability. This comparison can inform policy recommendations and support programs.

- **Limitations:**
    - *Outdated Data*: The analysis relies on data from 2022, and it is currently 2024. Economic conditions may have changed, affecting the relevance of the findings.
    - *Lack of Specific Income Data:* The analysis does not include specific income data for households in each county, limiting the ability to precisely assess economic conditions.
    - *Household Type Information*: There is no detailed information on income for different household types, which could provide more nuanced insights.
    - *Regional Variability*: The analysis does not account for regional economic variability within counties, which could affect the generalizability of the findings.
    - *Broader Economic Factors*: Factors such as inflation, employment rates, and policy changes over the past two years are not considered, which could impact the accuracy of the conclusions

  Improvements for future research:.

    - *Updated Data Collection*: Collecting and analyzing more recent data from 2023 and 2024 would provide a more accurate picture of current economic conditions.
    - *Detailed Income Analysis*: Incorporating detailed income data for different household types and regions within counties to better understand economic disparities.
    - *Longitudinal Studies*: Conducting longitudinal studies to track changes in ALICE thresholds, poverty rates, and economic conditions over time.
    - *Impact of Policies*: Evaluating the impact of recent policy changes and economic factors on household financial stability and ALICE thresholds

7. Recommendations

- Actionable insights:
    - Develop and implement targeted financial assistance programs for the most disadvantaged sub-counties. These programs should focus on areas with the highest poverty rates and the most significant economic disparities.
    - Increase access to affordable childcare services, particularly in counties with high ALICE thresholds.
    - Adjust cost of living allowances and support programs to reflect the rising ALICE thresholds in counties like Delaware, Franklin, and Licking.
    - Promote economic development initiatives that create well-paying jobs and improve economic stability in high-poverty areas.
    - Develop comprehensive support systems that address multiple aspects of household stability, including housing, healthcare, and education.

8. References

- **Citations:**
    - https://unitedforalice.org/state-overview/Ohio

9. Appendices

```
import pandas as pd
import numpy as np

# Get the files
file_path1 = './convert/sheet1.csv'
file_path2 = './convert/sheet2.csv'

df1 = pd.read_csv(file_path1)
df2 = pd.read_csv(file_path2)
print("Loaded csv files. Ready to analyse")

# Analysis

### Overview of sheet1, which is the more broad data
print(df1.head(10))

# Top 10 highest poverty rate county
df1['Poverty_Round'] = np.round(df1['Poverty Households'] / df1['Households']*100)
df1['Under 65'] = np.round(df1['ALICE Threshold - HH under 65'])
df1['Equal or greater than 65'] = np.round(df1['ALICE Threshold - HH 65 years and
over'])

# Select the required columns
df1_selected = df1[['Year', 'County', 'Poverty_Round', 'Under 65', 'Equal or greater
than 65']]

# Group by 'Year' and 'County' and ensure uniqueness
df1_grouped = df1_selected.groupby(['Year', 'County'], as_index=False).first()

# Sort by 'Year' in descending order
df1_sorted = df1_grouped.sort_values(by=['Poverty_Round'], ascending=False)

df1_sorted = df1_sorted.head(10)

print(df1_sorted)

# Athens and Scioto have the highest poverty rate

### Top 10 worst poverty rate county
```

```
df1_sorted = df1_grouped.sort_values(by=['Poverty_Round'], ascending=True)

df1_sorted = df1_sorted.head(10)

print(df1_sorted)



### Franklin, Delaware, and Licking counties over the years regardin poverty rate

# Franklin poverty rate
df1_filtered = df1[df1['County'] == 'Franklin']

# Calculate the poverty rate and round the result
df1_filtered['Poverty Rate'] = np.round((df1_filtered['Poverty Households'] /
df1_filtered['Households']) * 100)

# Group by 'year' and 'county' and aggregate the poverty rate
df1_grouped = df1_filtered.groupby(['Year', 'County']).agg({'Poverty Rate':
'first'}).reset_index()

# Sort the DataFrame by 'Poverty_Rate' in descending order
df1_sorted = df1_grouped.sort_values(by='Year', ascending=True)

# Display the result
print(df1_sorted)



# Delaware poverty rate
df1_filtered = df1[df1['County'] == 'Delaware']

# Calculate the poverty rate and round the result
df1_filtered['Poverty Rate'] = np.round((df1_filtered['Poverty Households'] /
df1_filtered['Households']) * 100)

# Group by 'year' and 'county' and aggregate the poverty rate
df1_grouped = df1_filtered.groupby(['Year', 'County']).agg({'Poverty Rate':
'first'}).reset_index()

# Sort the DataFrame by 'Poverty_Rate' in descending order
df1_sorted = df1_grouped.sort_values(by='Year', ascending=True)

# Display the result
```

```
print(df1_sorted)

#Licking poverty rate
df1_filtered = df1[df1['County'] == 'Licking']

# Calculate the poverty rate and round the result
df1_filtered['Poverty Rate'] = np.round((df1_filtered['Poverty Households'] /
df1_filtered['Households']) * 100)

# Group by 'year' and 'county' and aggregate the poverty rate
df1_grouped = df1_filtered.groupby(['Year', 'County']).agg({'Poverty Rate':
'first'}).reset_index()

# Sort the DataFrame by 'Poverty_Rate' in descending order
df1_sorted = df1_grouped.sort_values(by='Year', ascending=True)

# Display the result
print(df1_sorted)

### Poverty rate stays relatively the same for the overall county, 2-3%
increase/decrease

# Overview of sheet2
print(df2.head(10))



# Compare Franklin and Delaware and Licking
filtered_df = df2[df2['Location'].str.contains('Franklin County', case=False,
na=False) & df2['Location'].str.contains('Delaware County', case=False, na=False) &
df2['Location'].str.contains('Licking County', case=False, na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_grouped = filtered_df.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
```

```
print(df2_limited)



# Sort the DataFrame by 'ALICE_or_lower_rate' in descending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)



### Average, highest, and lowest ALICE threshold

# Group by 'County' and calculate mean, min, and max
df1_grouped = df1.groupby('County').agg({
   'ALICE Threshold - HH under 65': ['mean', 'min', 'max'],
   'ALICE Threshold - HH 65 years and over': ['mean', 'min', 'max']
}).reset_index()

# Flatten the multi-level column names
df1_grouped.columns = ['_'.join(col).strip() if col[1] else col[0] for col in
df1_grouped.columns.values]

# Rename the columns for readability
df1_grouped = df1_grouped.rename(columns={
   'ALICE Threshold - HH under 65_mean': 'Under_65_mean',
   'ALICE Threshold - HH under 65_min': 'Under_65_min',
   'ALICE Threshold - HH under 65_max': 'Under_65_max',
   'ALICE Threshold - HH 65 years and over_mean': 'Over_65_mean',
   'ALICE Threshold - HH 65 years and over_min': 'Over_65_min',
   'ALICE Threshold - HH 65 years and over_max': 'Over_65_max'
})

# Display the top 10 county with the lowest average for under 65
df1_low_average_under_65 = df1_grouped[['County', 'Under_65_mean']]
df1_low_average_under_65 = df1_low_average_under_65.sort_values(by='Under_65_mean',
ascending=True)
print("\n", df1_low_average_under_65.head(10))

# Display the top 10 county with the highest average for under 65
```

```
df1_high_average_under_65 = df1_grouped[['County', 'Under_65_mean']]
df1_high_average_under_65 = df1_high_average_under_65.sort_values(by='Under_65_mean',
ascending=False)
print("\n", df1_high_average_under_65.head(10))

# Display the top 10 county with the lowest average for at least 65
df1_low_average_over_65 = df1_grouped[['County', 'Over_65_mean']]
df1_low_average_over_65 = df1_low_average_over_65.sort_values(by='Over_65_mean',
ascending=True)
print("\n", df1_low_average_over_65.head(10))

# Display the top 10 county with the highest average for at least 65
df1_high_average_over_65 = df1_grouped[['County', 'Over_65_mean']]
df1_high_average_over_65 = df1_high_average_over_65.sort_values(by='Over_65_mean',
ascending=False)
print("\n", df1_high_average_over_65.head(10))

# Display the top 10 county with the lowest threshold for under 65
df1_min_under_65 = df1_grouped[['County', 'Under_65_min']]
df1_min_under_65 = df1_min_under_65.sort_values(by='Under_65_min', ascending=True)
print("\n", df1_min_under_65.head(10))

# Display the top 10 county with the lowest threshold for over 65
df1_min_over_65 = df1_grouped[['County', 'Over_65_min']]
df1_min_over_65 = df1_min_over_65.sort_values(by='Over_65_min', ascending=True)
print("\n", df1_min_over_65.head(10))

# Display the top 10 county with the highest threshold for under 65
df1_max_under_65 = df1_grouped[['County', 'Under_65_max']]
df1_max_under_65 = df1_max_under_65.sort_values(by='Under_65_max', ascending=False)
print("\n", df1_max_under_65.head(10))

# Display the top 10 county with the highest threshold for over 65
df1_max_over_65 = df1_grouped[['County', 'Over_65_max']]
df1_max_over_65 = df1_max_over_65.sort_values(by='Over_65_max', ascending=False)
print("\n", df1_max_over_65.head(10))


### From the top mean, min, max threshold county, check the poverty rate of
subcounties

### Under 65 - lowest mean
```

```
df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Crawford County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

### Under 65 - highest mean

df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Holmes County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
```

```
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)


### Over 65 - lowest mean

df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Mercer County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

### Over 65 - highest mean

df2['Location'] = df2['GEO.display_label']
```

```
df2_filtered = df2[df2['Location'].str.contains('Warren County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)


### Under 65 - min

df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Wyandot County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)
```

```
# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)


### Under 65 - max

df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Crawford County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
```

```
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)


### Over 65 - min

df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Guernsey County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)
```

```
# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

### Over 65 - max

df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Crawford County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)


### Crawford county has the highest threshold for all age group, and the worst poverty
rate is 52% in 2022
```

```
### Check with Franklin, Delware, and Licking counties since these are the focus

# Franklin 2022
df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Franklin County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)


# Sort the DataFrame by 'ALICE_or_lower_rate' in descending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)


# Delaware 2022
df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Delaware County, Ohio', case=False,
na=False)]
```

```
# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()

# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)



# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)



# Licking 2022
df2['Location'] = df2['GEO.display_label']

df2_filtered = df2[df2['Location'].str.contains('Licking County, Ohio', case=False,
na=False)]

# Calculate the ALICE_or_lower_rate and round the result
df2_filtered['ALICE_or_lower_rate'] = np.round(((df2_filtered['Poverty Households'] +
df2_filtered['ALICE Households']) / df2_filtered['Households']) * 100)

# Group by 'Year' and 'Location' and aggregate the ALICE_or_lower_rate
df2_grouped = df2_filtered.groupby(['Year', 'Location']).agg({'ALICE_or_lower_rate':
'first'}).reset_index()
```

```
# Sort the DataFrame by 'ALICE_or_lower_rate' in ascending order
df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=False)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

df2_sorted = df2_grouped.sort_values(by='ALICE_or_lower_rate', ascending=True)

# Limit the DataFrame to the first 10 rows
df2_limited = df2_sorted.head(10)

# Display the result
print(df2_limited)

### Threshold, survival and stability budget

# Manually extract the data

# Franklin
print("\nFranklin")
data_values = [
   ('Single Adult', '$30,084'),
   ('One Adult One Child', '$46,932'),
   ('One Adult One Childcare', '$54,480'),
   ('Two Adults', '$44,700'),
   ('Two Adults Two Children', '$73,524'),
   ('Two Adults Two Childcare', '$91,284'),
   ('Single Senior', '$34,644'),
   ('Two Seniors', '$54,468')
]


df_data = pd.DataFrame(data_values, columns=['Name', 'ANNUAL TOTAL'])
df_data['ANNUAL TOTAL'] = df_data['ANNUAL TOTAL'].str.replace('$',
'').str.replace(',', '').astype(float)

under_65 = df1.loc[df1['County'] == 'Franklin', 'ALICE Threshold - HH under
65'].values[0]
```

```
over_65 = df1.loc[df1['County'] == 'Franklin', 'ALICE Threshold - HH 65 years and
over'].values[0]

filtered_data = df_data[df_data['ANNUAL TOTAL'] <= under_65]
print(filtered_data, under_65)

filtered_data = df_data[df_data['ANNUAL TOTAL'] <= over_65]
print(filtered_data, over_65)

# Delaware
print("\nDelaware")
data_values = [
   ('Single Adult', '$31,560'),
   ('One Adult One Child', '$48,720'),
   ('One Adult One Childcare', '$56,220'),
   ('Two Adults', '$46,476'),
   ('Two Adults Two Children', '$75,984'),
   ('Two Adults Two Childcare', '$93,660'),
   ('Single Senior', '$35,688'),
   ('Two Seniors', '$55,488')
]

df_data1 = pd.DataFrame(data_values, columns=['Name', 'ANNUAL TOTAL'])
df_data1['ANNUAL TOTAL'] = df_data1['ANNUAL TOTAL'].str.replace('$',
'').str.replace(',', '').astype(float)

under_65 = df1.loc[df1['County'] == 'Delaware', 'ALICE Threshold - HH under
65'].values[0]
over_65 = df1.loc[df1['County'] == 'Delaware', 'ALICE Threshold - HH 65 years and
over'].values[0]

filtered_data = df_data1[df_data1['ANNUAL TOTAL'] <= under_65]
print(filtered_data, under_65)

filtered_data = df_data1[df_data1['ANNUAL TOTAL'] <= over_65]
print(filtered_data, over_65)

# Licking

print("\nLicking")
data_values = [
   ('Single Adult', '$28,032'),
```

```
    ('One Adult One Child', '$43,596'),
    ('One Adult One Childcare', '$49,932'),
    ('Two Adults', '$42,108'),
    ('Two Adults Two Children', '$68,400'),
    ('Two Adults Two Childcare', '$82,404'),
    ('Single Senior', '$32,532'),
    ('Two Seniors', '$51,744')
]

df_data2 = pd.DataFrame(data_values, columns=['Name', 'ANNUAL TOTAL'])
df_data2['ANNUAL TOTAL'] = df_data2['ANNUAL TOTAL'].str.replace('$',
'').str.replace(',', '').astype(float)

under_65 = df1.loc[df1['County'] == 'Licking', 'ALICE Threshold - HH under
65'].values[0]
over_65 = df1.loc[df1['County'] == 'Licking', 'ALICE Threshold - HH 65 years and
over'].values[0]

filtered_data = df_data2[df_data2['ANNUAL TOTAL'] <= under_65]
print(filtered_data, under_65)

filtered_data = df_data2[df_data2['ANNUAL TOTAL'] <= over_65]
print(filtered_data, over_65)

### Those who live alone struggles to meet with the survival budget

# Franklin
print("\nFranklin")
data = {
    'Name': ['Single Adult', 'Two Adults', 'Two Adults Two Children', 'Two Adults Two
Childcare'],
    'ANNUAL TOTAL': ['$51,432', '$74,952', '$124,632', '$148,392']
}

df_data3 = pd.DataFrame(data)
df_data3['ANNUAL TOTAL'] = df_data3['ANNUAL TOTAL'].str.replace('$',
'').str.replace(',', '').astype(float)

under_65 = df1.loc[df1['County'] == 'Franklin', 'ALICE Threshold - HH under
65'].values[0]
over_65 = df1.loc[df1['County'] == 'Franklin', 'ALICE Threshold - HH 65 years and
over'].values[0]
```

```
df_data3['difference'] = df_data3['ANNUAL TOTAL'] - under_65
print(df_data3[['Name', 'difference']])


# Delaware
print("\nDelaware")
data_values = {
   'Name': ['Single Adult', 'Two Adults', 'Two Adults Two Children', 'Two Adults Two
Childcare'],
   'ANNUAL TOTAL': ["$53,652","$77,892","$142,788","$166,896"]

}

df_data4 = pd.DataFrame(data)
df_data4['ANNUAL TOTAL'] = df_data4['ANNUAL TOTAL'].str.replace('$',
'').str.replace(',', '').astype(float)

under_65 = df1.loc[df1['County'] == 'Delaware', 'ALICE Threshold - HH under
65'].values[0]
over_65 = df1.loc[df1['County'] == 'Delaware', 'ALICE Threshold - HH 65 years and
over'].values[0]

df_data4['difference'] = df_data4['ANNUAL TOTAL'] - under_65
print(df_data4[['Name', 'difference']])


# Licking
print("\nLicking")
data_values = {
   'Name': ['Single Adult', 'Two Adults', 'Two Adults Two Children', 'Two Adults Two
Childcare'],
   'ANNUAL TOTAL': ["$48,360","$70,788","$117,036","$133,044"]

}

df_data5 = pd.DataFrame(data)
df_data5['ANNUAL TOTAL'] = df_data5['ANNUAL TOTAL'].str.replace('$',
'').str.replace(',', '').astype(float)

under_65 = df1.loc[df1['County'] == 'Licking', 'ALICE Threshold - HH under
65'].values[0]
```

```
over_65 = df1.loc[df1['County'] == 'Licking', 'ALICE Threshold - HH 65 years and
over'].values[0]

df_data5['difference'] = df_data5['ANNUAL TOTAL'] - under_65
print(df_data5[['Name', 'difference']])

# The gap to catch up to is large, ranging from 10000 to more than 100000$ for the
under 65 group
```