Exploratory Data Analysis

- StudentID: 22100163

- Name: Kim Yujin

- 1st Major: Life Science

- 2nd Major: AI Convergence

The General Social Survey (GSS), initiated in 1972, is a vital sociological research project conducted by the National Opinion Research Center (NORC) at the University of Chicago, supported by the National Science Foundation. This biannual survey gathers data on contemporary American society to monitor and explain trends and constants in attitudes, behaviors, and attributes. The survey covers a wide array of topics, including civil liberties, crime and violence, intergroup tolerance, morality, national spending priorities, psychological well-being, social mobility, and stress and traumatic events.

The GSS serves as the single best source for sociological and attitudinal trend data in the United States. It captures hundreds of trends since its inception in 1972 and adopts questions from earlier surveys, allowing trends to be followed for up to 70 years. This comprehensive dataset enables researchers to examine the structure and functioning of society, analyze the role played by relevant subgroups, and make comparisons between the United States and other nations.

Through its rigorous methodology and continuous innovations, the GSS provides valuable insights into the evolving concerns, attitudes, and experiences of U.S. residents. It has become a cornerstone in social science research, contributing to numerous academic papers, books, and student research projects. The GSS plays a pivotal role in shaping our understanding of American society and informs the decisions of scholars, students, and policymakers alike.

Using this data, we can explore the trends in respondents' age and employment status. This emphasizes the vital role played by the General Social Survey (GSS) in capturing various aspects of American society, tracking changing attitudes, behaviors, and experiences. The GSS allows us to investigate the structure and functioning of society, providing valuable insights for researchers to understand and analyze social trends.

1. Data overview

Data explanation consists of descriptives statistics on overall data, including sample size, number of variables, data type, data range, distribution, etc.

Below is the representation of 'General Social Survey' data, that is used for this EDA.

```
#Load the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import warnings
warnings.filterwarnings(action='ignore')
```

```
chunk_size = 1000
csv_file = '/Users/kim-yujin/Desktop/빅종설/eda/gss.csv'

chunk_list = []
for chunk in pd.read_csv(csv_file, chunksize=chunk_size):
    processed_chunk = chunk
    chunk_list.append(processed_chunk)

result_df = pd.concat(chunk_list, axis=0)
result_df.head(5)
```

	GSS YEAR FOR THIS RESPONDENT	RESPONDNT ID NUMBER	LABOR FORCE STATUS	LABOR FORCE STATUS_labels	NUMBER OF HOURS WORKED LAST WEEK	NUMBER OF HOURS WORKED LAST WEEK_labels	NUMBER OF HOURS USUALLY WORK A WEEK	NUMBER OF HOURS USUALLY WORK A WEEK_labels	EVER WORK AS LONG AS ONE YEAR	EVER WORK AS LONG AS ONE YEAR_labels	...
0	year	id	wrkstat	NaN	hrs1	NaN	hrs2	NaN	evwork	NaN	..
1	1972.0	1.0	1.0	WORKING FULLTIME	-1.0	IAP	-1.0	IAP	NaN	IAP	..
2	1972.0	2.0	5.0	RETIRED	-1.0	IAP	-1.0	IAP	1.0	YES	..
3	1972.0	3.0	2.0	WORKING PARTTIME	-1.0	IAP	-1.0	IAP	NaN	IAP	..
4	1972.0	4.0	1.0	WORKING FULLTIME	-1.0	IAP	-1.0	IAP	NaN	IAP	..

5 rows × 11232 columns

Above, the given csv file is read in data frame format. Since the given data is large-capacity data, we processed it efficiently using chunk. First, the chunk_size variable defines the size of each chunk. Then, use the pd.read_csv() function to read large CSV files in chunks. Each chunk traverses using the for loop, performs the necessary data processing operations, and stores the results in the chunk_list list.

Finally, use the pd.concat() function to merge the results of each chunk stored in the chunk_list into one data frame, result_df. This allows you to divide large amounts of data into several small chunks and effectively combine the results into a single data frame.

```
print(result_df.shape)

result_df.info()
```

```
(59600, 11232)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59600 entries, 0 to 59599
Columns: 11232 entries, GSS YEAR FOR THIS RESPONDENT                        to Variance primary sampling unit_la
dtypes: float64(7), object(11225)
memory usage: 5.0+ GB
```

We can see from 'result_df.shape' that this GSS data contains 59600 rows and 11232 columns.

Through the code `result_df.info()`, we can determine that the data is stored in pandas as a DataFrame. From this, we can gather the following information:

1. The DataFrame's index is a `RangeIndex` ranging from 0 to 59599.
2. The DataFrame consists of a total of 11232 columns.
3. The columns in this DataFrame have two distinct data types:
    - float64: 7 columns are of the float64 data type, representing numbers that may include decimals.
    - object: 11225 columns are of the object data type, typically representing textual or string data, which may include text or string values.

```
print(result_df.isnull().sum().sum())
```

```
281735581
```

When calculating the total sum of missing values (or NaN) in `result_df`, it reveals that there are 281,735,581 missing values in the dataset. This information can be valuable for evaluating data completeness and deciding on strategies to handle these missing values.

```
result_df.describe(include='all')
```

	GSS YEAR FOR THIS RESPONDENT	RESPONDNT ID NUMBER	LABOR FORCE STATUS	LABOR FORCE STATUS_labels	NUMBER OF HOURS WORKED LAST WEEK	NUMBER OF HOURS WORKED LAST WEEK_labels	NUMBER OF HOURS USUALLY WORK A WEEK	NUMBER OF HOURS USUALLY WORK A WEEK_labels	EVER WORK AS LONG AS ONE YEAR	EVER WOR AS LONG A ON YEAR_label
count	59600.0	59600.0	59600.0	59583	59593.0	24946	59598.0	58385	20833.0	5953
unique	32.0	4574.0	17.0	8	94.0	2	64.0	2	7.0	
top	2006.0	1131.0	1.0	WORKING FULLTIME	-1.0	IAP	-1.0	IAP	1.0	IA
freq	4510.0	30.0	29404.0	29437	24840.0	24903	58319.0	58382	17453.0	3876
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

11 rows × 11232 columns

The output generated by `result_df.describe(include='all')` provides a comprehensive summary of the dataset, offering a detailed overview of various aspects. It includes statistics such as count, unique values, top value, and frequency for categorical columns, and descriptive statistics like mean, standard deviation, minimum, quartiles, and maximum for numerical columns. This summary allows us to grasp the distribution, central tendencies, and dispersion of the data.

2. Univariate analysis

Univariate analysis is one of the fundamental analytical methods used in statistics and data analysis. It involves examining and analyzing a single variable to understand its characteristics, distribution, central tendencies, and variability. Univariate analysis is primarily employed to gain insights into the basic properties of a variable, allowing researchers to comprehend the data represented by that variable. By exploring the fundamental traits of the variable, univariate analysis provides valuable insights into the nature of the data, laying the groundwork for more complex multivariate analyses. It enhances the in-depth understanding and interpretation of the data's characteristics.

In this analysis, respondents were analyzed and expressed visually using the "GSS YEAR FOR THIS RESPONDENT" and "AGE OF RESPONDENT" variables.

2.1 Response classification by year

```
result_df.dropna(axis=1,how='all',inplace=True)
```

Remove columns with value values all non.

```
result_df.shape
```

```
    (59600, 11225)
```

In result_df, delete the column with all values NaN and apply this action to the existing result_df. This allows you to remove columns that contain all missing values from the data frame.

```
year_counts = result_df.loc[1:, 'GSS YEAR FOR THIS RESPONDENT                       '].value_counts()
year_counts.index = [int(float(str(index))) for index in year_counts.index]
```

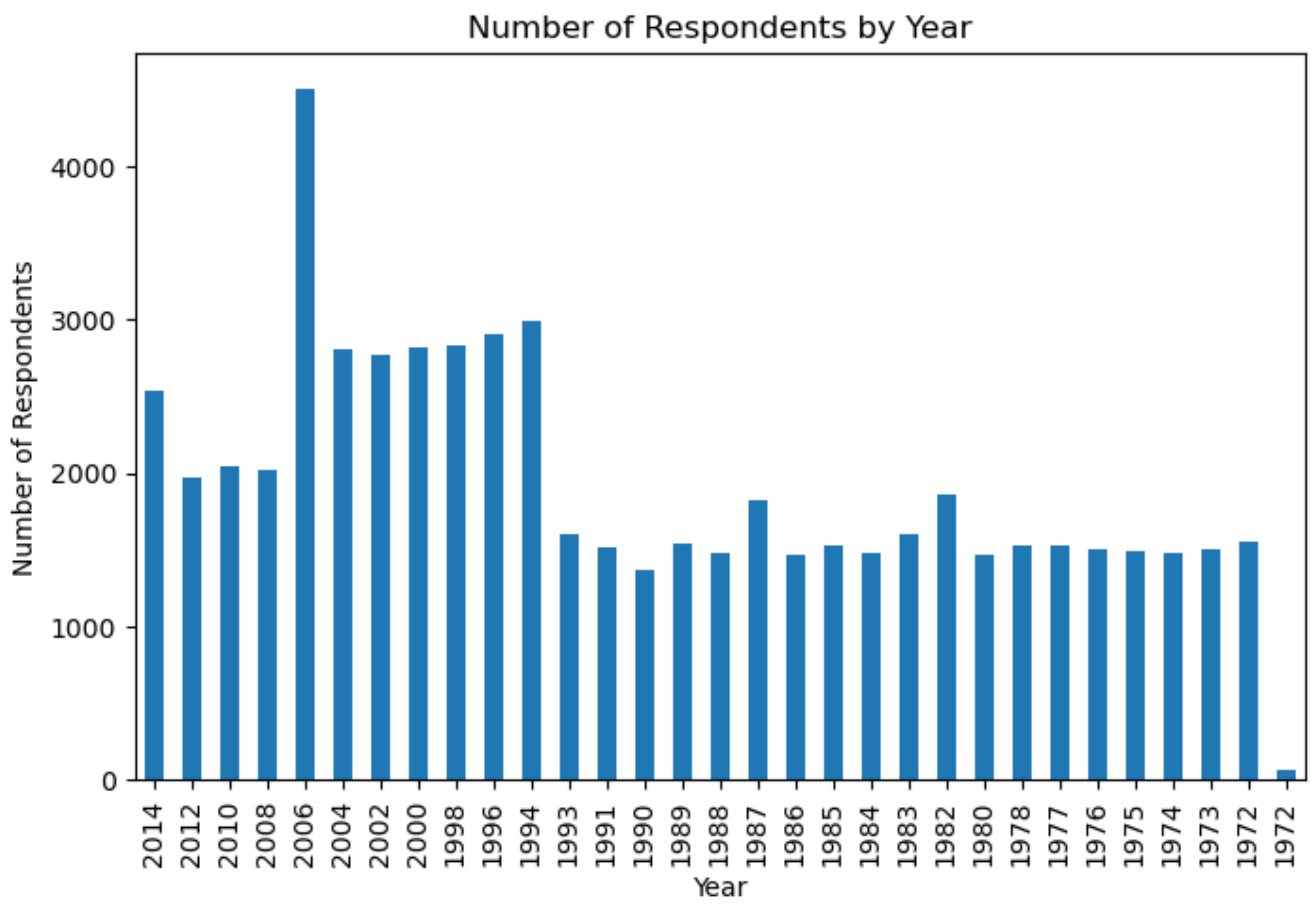
year_counts is data that has the number of responses for each year using the column in result_df. Of the value values, only the 1972.0 value is now in the string state, so the string is converted to an integer ("1972.0" string is converted to an integer 1972)

```
print(year_counts.index)
```

```
    Index([2006, 1994, 1996, 1998, 2000, 2004, 2002, 2014, 2010, 2008, 2012, 1982,
           1987, 1993, 1983, 1972, 1989, 1985, 1978, 1977, 1991, 1973, 1976, 1975,
           1974, 1988, 1984, 1986, 1980, 1990, 1972],
          dtype='int64')
```

year_counts.Use index to determine the unique index of year_counts.

```
year_counts = year_counts.sort_index(ascending=False)
year_counts
year_counts.plot(kind='bar', figsize=(8, 5))
plt.xlabel('Year')
plt.ylabel('Number of Respondents')
plt.title('Number of Respondents by Year')
plt.show()
```



This visualization allows you to see changes in the number of respondents over the year and evaluate trends, outliers, patterns, and the quality of data collection. If you look at the graph above, you can see how many respondents respond to each year. The bar graph above shows that the number of responses in 2006 was the largest.

2.2 Age distribution of respondents

"Labor Force Status" indicates the employment status of respondents. This column represents which labor force group the respondent belongs to. Typically, labor force status can include various values such as:

WORKING FULLTIME RETIRED WORKING PARTTIME KEEPING HOUSE etc..

```
result_df=result_df.iloc[1:]
age_df=result_df[['AGE OF RESPONDENT']]
```

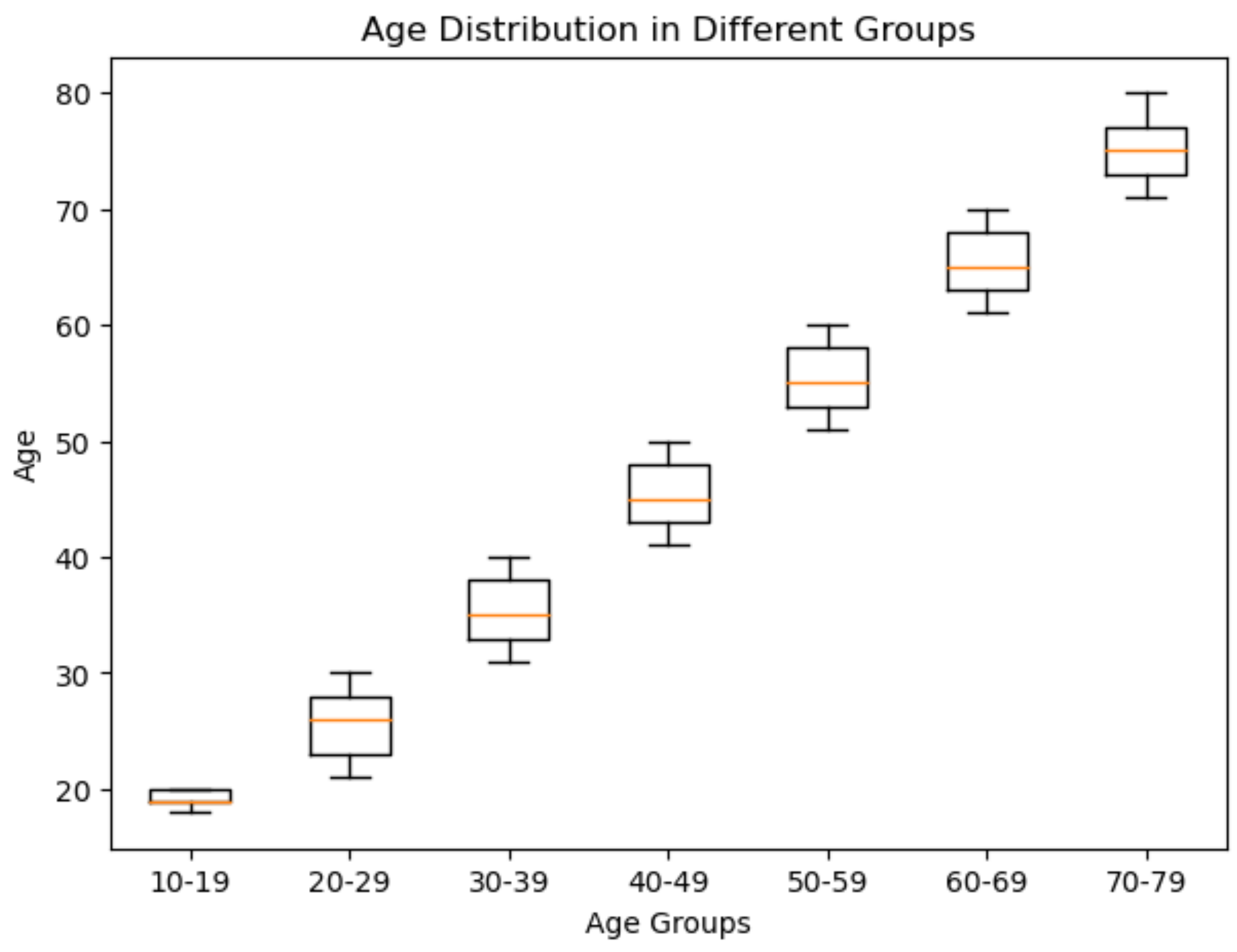
A new DataFrame `age_df` is created by selecting only the 'AGE OF RESPONDENT' column from the modified `result_df` DataFrame. This operation results in a new DataFrame `age_df` containing only the 'AGE OF RESPONDENT' column data, allowing further analysis specifically focused on the respondents' ages.

```
age_df['AGE OF RESPONDENT'] = pd.to_numeric(age_df['AGE OF RESPONDENT'], errors='coerce')

bins = [10, 20, 30, 40, 50, 60, 70, 80]
labels = ['10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']
age_categories = pd.cut(age_df['AGE OF RESPONDENT'], bins=bins, labels=labels)
age_counts = age_categories.value_counts()
```

The code above categorizes age data by age group and calculates the number of respondents in each age group. This allows you to determine the distribution of respondents by age group. For example, you can visually see how many respondents are in different age groups: 10-19 years old, 20-29 years old, ... and 70-79 years old. This can help you understand the overall nature of the respondents' age.

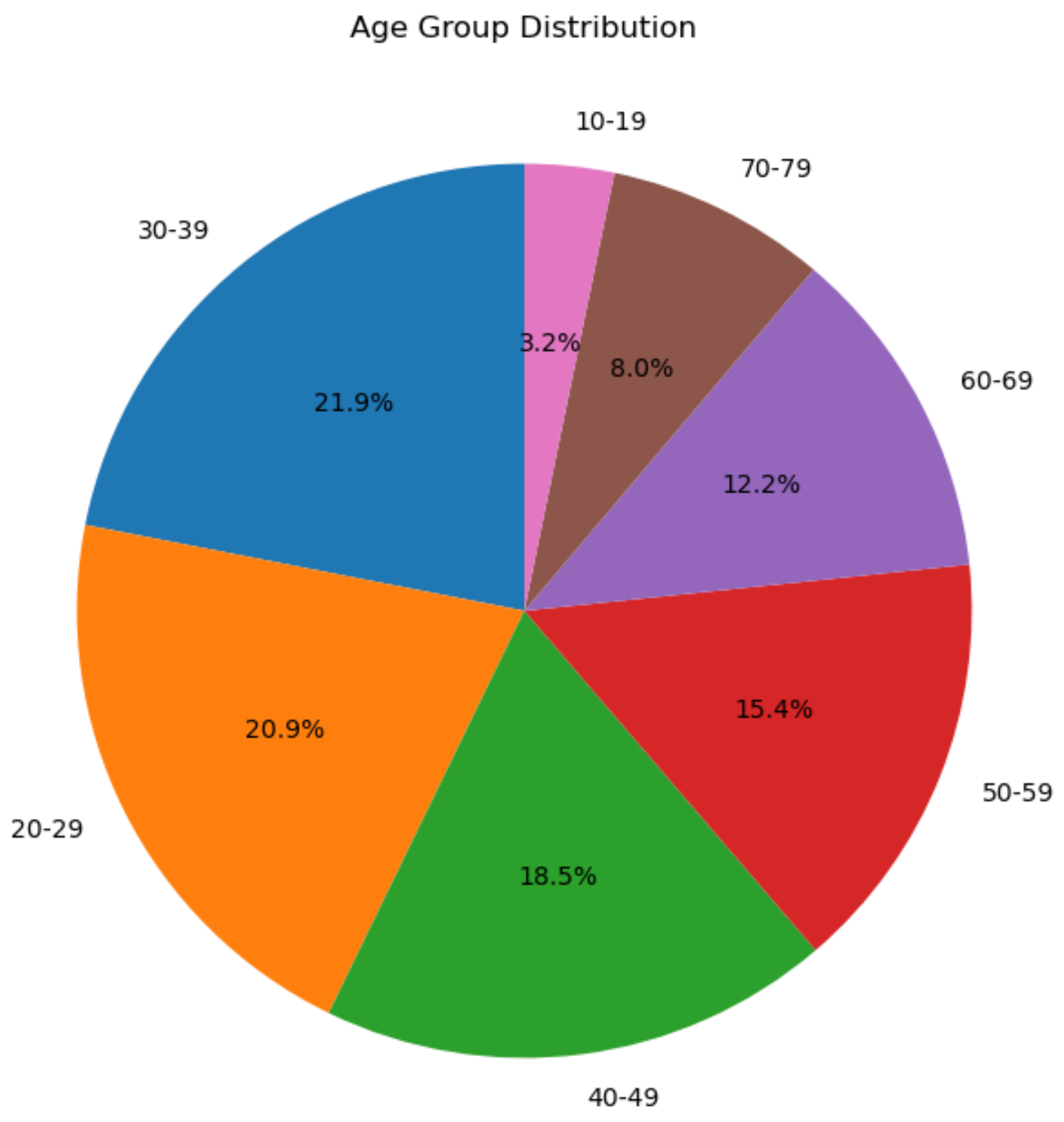
```
plt.figure(figsize=(7, 5))
age_df['AGE OF RESPONDENT'] = pd.to_numeric(age_df['AGE OF RESPONDENT'], errors='coerce')
age_df_cleaned = age_df.dropna()
plt.boxplot([age_df_cleaned[age_categories == label]['AGE OF RESPONDENT'] for label in labels],
            labels=labels)
plt.xlabel('Age Groups')
plt.ylabel('Age')
plt.title('Age Distribution in Different Groups')
plt.show()
```



A visualization of the age distribution of respondents according to each age group. A boxplot shows the age distribution of respondents by age group. This allows you to visually see the difference in age distribution by age group by comparing the median, quartile range, and outliers of each age group.You can compare the distribution of ages by age group to see which age groups are more spread or clustered around.

The box plot for each age group contains a horizontal line representing the median. The box represents the interquartile range, the middle 50% of the data. This allows you to compare the spread of the age distribution with the median age in each age group. In addition, you can compare the boxplot for each age group horizontally to see at which age group the age is more spread and has more unusual values.

```
plt.figure(figsize=(8, 8))
plt.pie(age_counts, labels=age_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Age Group Distribution')
plt.show()
```



The above figure is a pie chart visualization of the distribution of respondents by age group. The pie chart provides the following information:

The pie chart represents the number of respondents in each age group as a relative percentage. The size of each pie slice represents the percentage of respondents in that age group.

The pie chart also shows that the age group with the largest pie slice is the age group with the largest percentage of respondents. Similarly, the age group with the smallest slice of pie is the age group with a relatively small percentage. This data shows that the percentage of people in their 30s is 21.9%, the most.

Finally, the size of the pie chart fragment for each age group provides a visual indication of the proportion of all respondents in that age group.

This lets you know which age groups most commonly appear within the dataset, and which have a smaller percentage.

3. Multivariate analysis

Multivariable analysis refers to statistical descriptions and methodologies that understand and interpret relationships between variables. This method is used to understand how two or more variables interact with each other and how they change together.

This analysis is used in data science, statistics, machine learning, social sciences, economics, and many other fields. By identifying the relationships between variables, you can find patterns in the data, develop predictive models, or support decision making. Correlation analysis and k-means clustering were utilized in the eda task.

3.1 Correlation Analysis of Age and Labor Force Status in Respondents

```
result_df=result_df.iloc[1:]
age_labor_df=result_df[['AGE OF RESPONDENT','LABOR FORCE STATUS','LABOR FORCE STATUS_labels']]
```

age_labor_df is a df that stores three columns required for correlation analysis separately.('AGE OF RESPONDENT','LABOR FORCE STATUS','LABOR FORCE STATUS_labels')

```
result_df['AGE OF RESPONDENT'] = pd.to_numeric(result_df['AGE OF RESPONDENT'], errors='coerce')
result_df['LABOR FORCE STATUS'] = pd.to_numeric(result_df['LABOR FORCE STATUS'], errors='coerce')
```

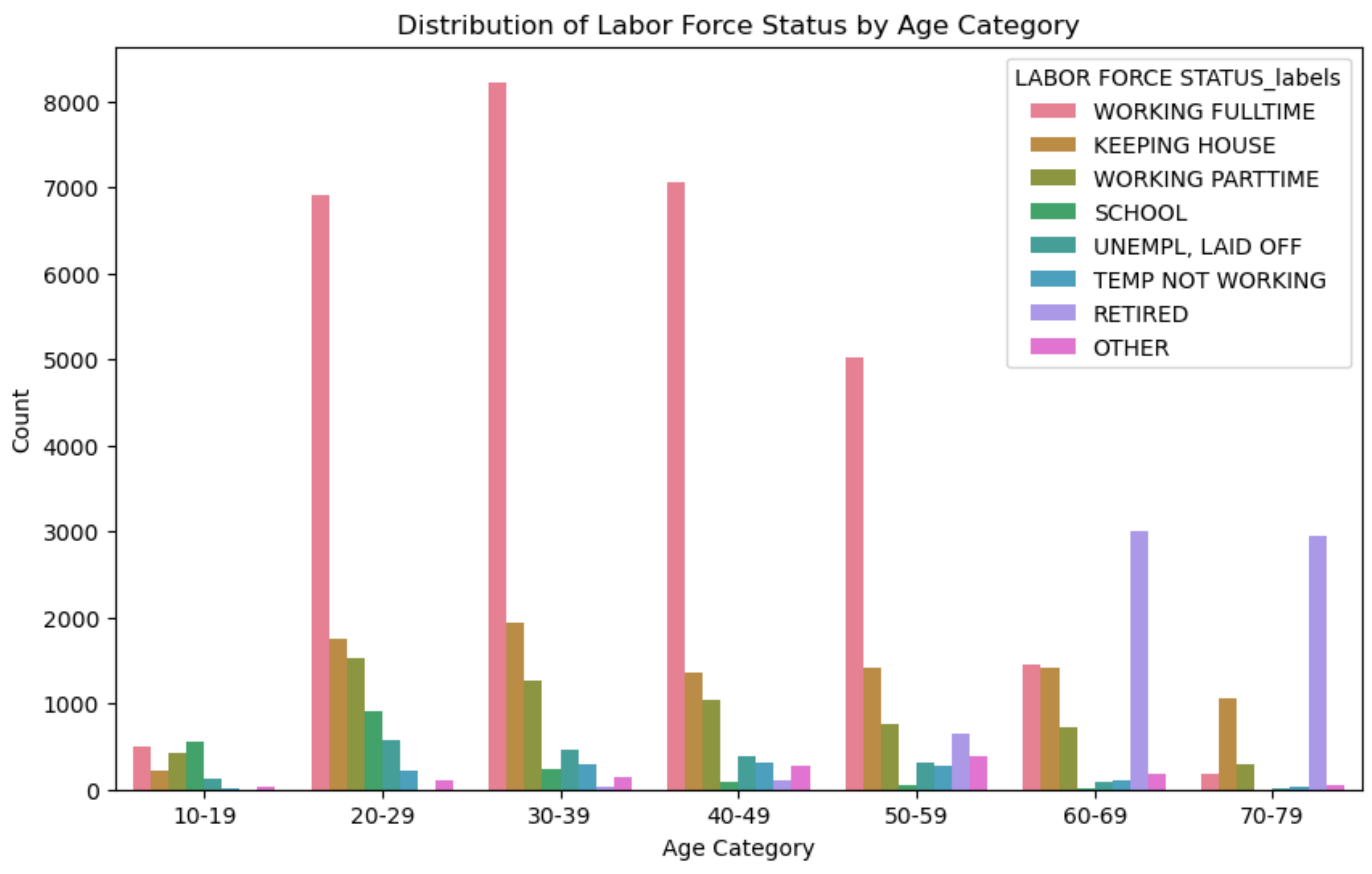
```
import seaborn as sns
import matplotlib.pyplot as plt

bins = [10, 20, 30, 40, 50, 60, 70, 80]
labels = ['10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']

age_categories = pd.cut(age_labor_df['AGE OF RESPONDENT'], bins=bins, labels=labels)

age_labor_df['Age Category'] = age_categories

plt.figure(figsize=(10, 6))
sns.countplot(data=age_labor_df, x='Age Category', hue='LABOR FORCE STATUS_labels', palette='husl')
plt.xlabel('Age Category')
plt.ylabel('Count')
plt.title('Distribution of Labor Force Status by Age Category')
plt.show()
```

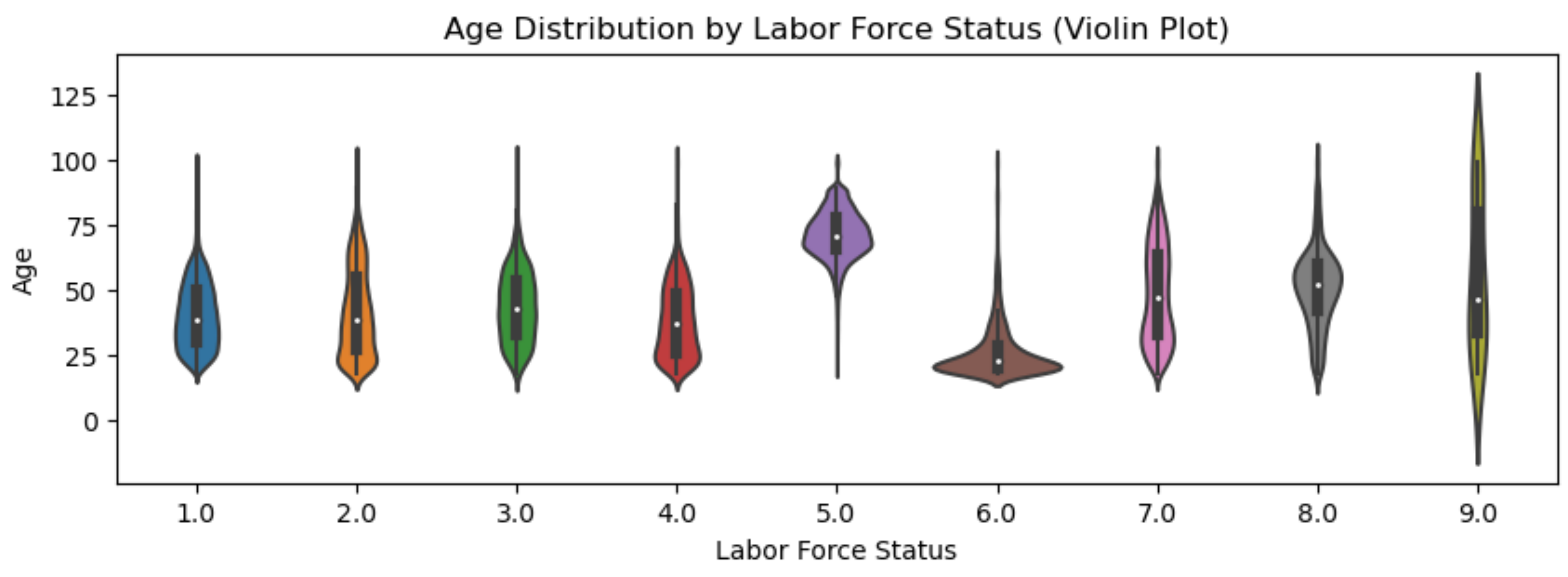


The code divides the respondents' ages into different age groups and visually shows the distribution of the value 'LABOR FORCE STATUS_labels' within each age group. A bar graph of the frequency of each 'LABOR FORCE STATUS_labels' value (code representing the state of work) for different age categories.

This graph makes it easy to compare the distribution of labor conditions by age, and which labor conditions usually appear in each age group. This allows you to visually see which types of working conditions are more common in a particular age group and the effect of age on working conditions.

```
# pip install seaborn
```

```
plt.figure(figsize=(10, 3))
sns.violinplot(x='LABOR FORCE STATUS', y='AGE OF RESPONDENT', data=result_df)
plt.title('Age Distribution by Labor Force Status (Violin Plot)')
plt.xlabel('Labor Force Status')
plt.ylabel('Age')
plt.show()
```



Above is a violin plot showing the distribution of 'AGE OF RESPONDENT' according to 'LABOR FORCE STATUS'. The violin plot represents the distribution of data, and it is a graphic that visually shows the distribution and central tendency of the data by representing Box Plot and Kernel Density Estimation together.

This violin plot shows the distribution of data for each category of 'LABOR FORCE STATUS'. The horizontal axis represents the working status, and the vertical axis represents the age of the respondent. A visual representation of the range and density of the age distribution for each working status category.

Respondents in the 'WORKING FULL TIME' category are mainly distributed in their late 20s to early 60s. Most 'WORKING FULL TIME' respondents are concentrated in their late 20s to early 50s.

Respondents in the 'RETIRED' category are mainly in their late 40s to 80s. Respondents in this category have greater variability as they age.

Respondents in the 'KEEPING HOUSE' category are mainly in their late 20s to 60s. 'KEEPING HOUSE' respondents are mostly concentrated in their late 30s to early 50s.

This interpretation provides a visual understanding of the difference in age distribution in each working status category. The violin plot is used as a useful tool for identifying the overall distribution and singularity of the data.

3.2 K-Means Clustering Results Based on Age and Labor Force Status

K-Means clustering is a popular algorithm known for its simplicity and ease of understanding. It can handle large-scale datasets efficiently. However, it has limitations such as sensitivity to initial centroids, which can lead to different results, and it assumes clusters to be spherical for optimal performance. K-Means is particularly suitable for large datasets due to its fast convergence and relatively low memory usage.

```
print(age_labor_df['LABOR FORCE STATUS_labels'].unique())

status_mapping = {'RETIRED':5, 'WORKING PARTTIME':2, 'WORKING FULLTIME':1, 'KEEPING HOUSE':7, 'SCHOOL':6,'UNEMPL, LAI

age_labor_df['LABOR FORCE STATUS_labels'] = age_labor_df['LABOR FORCE STATUS_labels'].map(status_mapping)
```

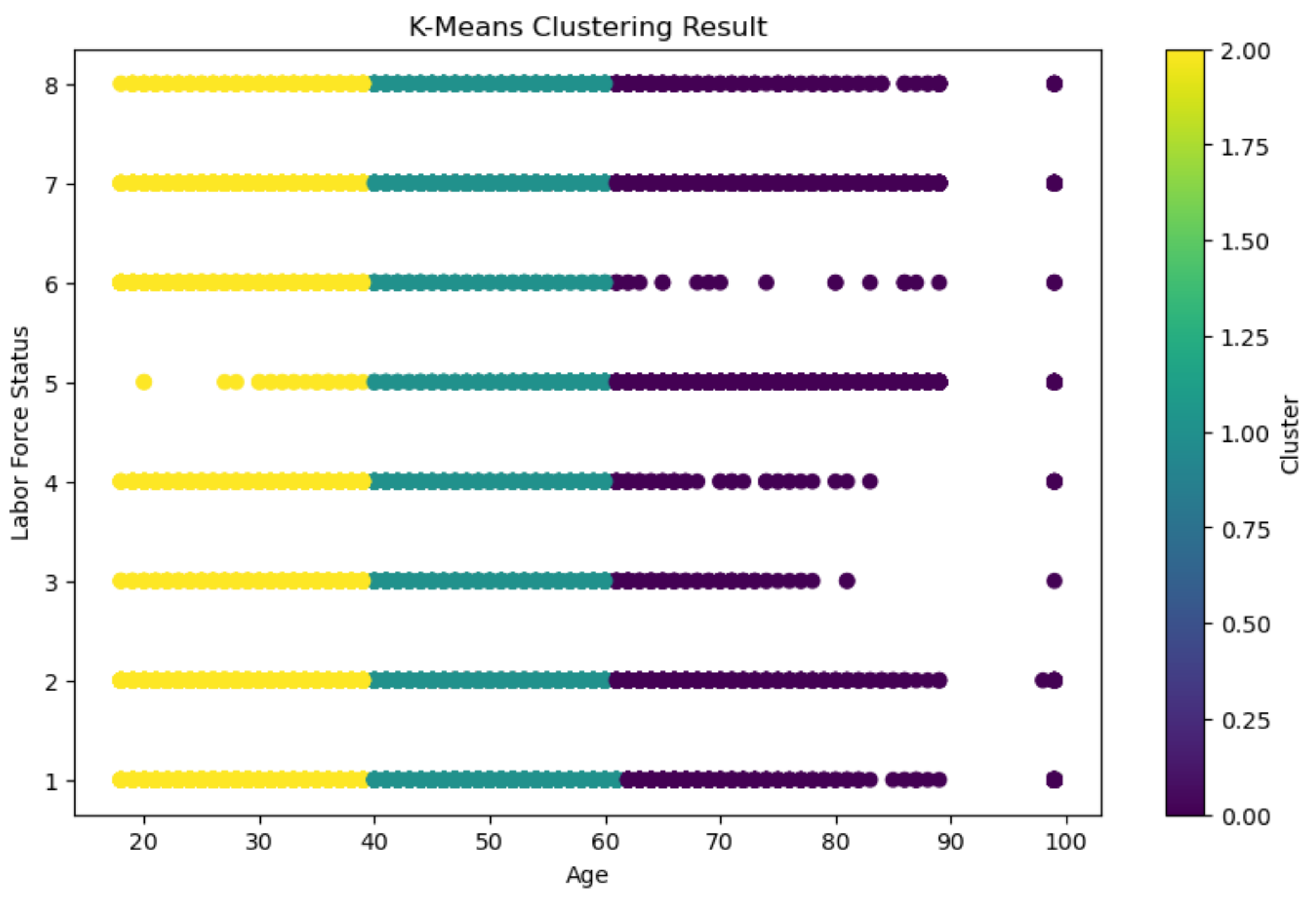
```
    ['WORKING FULLTIME' 'KEEPING HOUSE' 'WORKING PARTTIME' 'SCHOOL'
     'UNEMPL, LAID OFF' 'TEMP NOT WORKING' 'RETIRED' 'OTHER' nan]
```

Map LABOR FORCE STATUS_label and LABOR FORCE STATUS values respectively.

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

features = age_labor_df[['AGE OF RESPONDENT', 'LABOR FORCE STATUS']]

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(features)
age_labor_df['Cluster'] = kmeans.labels_
plt.figure(figsize=(10, 6))
plt.scatter(age_labor_df['AGE OF RESPONDENT'], age_labor_df['LABOR FORCE STATUS_labels'], c=age_labor_df['Cluster'],
plt.xlabel('Age')
plt.ylabel('Labor Force Status')
plt.title('K-Means Clustering Result')
plt.colorbar(label='Cluster')
plt.show()
```



The given code visualizes the results of K-Means clustering based on two variables, 'AGE OF RESPONDENT' and 'LABOR FORCE STATUS_labels.' In this case, it has been grouped into three clusters. Each cluster is differentiated and represented by a different color.

- **Cluster 0 (Yellow):** This group may primarily consist of younger individuals. Within this cluster, there is a variety of employment statuses, with a significant number of people likely in 'WORKING FULLTIME' and 'WORKING PARTTIME' positions.
- **Cluster 1 (Purple):** This group corresponds to individuals in the average age range, encompassing people with diverse employment statuses.
- **Cluster 2 (Turquoise):** This cluster could represent older individuals. It is expected that this group predominantly comprises people in the 'RETIRED' status.

4. Suggestion

Through the comprehensive analysis conducted, we have gained valuable insights into the complex interplay between age groups and their labor force status. These insights can serve as a foundation for shaping government policies and market strategies that cater to the distinct needs of different age demographics.

Firstly, targeting the younger generation (Cluster 0), the analysis reveals a prevalent trend of active employment. To harness their potential fully, policies should focus on bolstering educational initiatives and career guidance programs. By equipping them with essential skills and entrepreneurial knowledge, we can stimulate innovation and empower them to establish their ventures, fostering economic growth and self-reliance.

For the middle-aged population (Cluster 1), characterized by diverse labor force statuses, tailored solutions are imperative. Offering technical training and self-development programs will enhance their adaptability in the face of industrial transformations. By providing them with the tools to navigate changing job landscapes, we can foster resilience and open doors to a myriad of employment opportunities.

Lastly, the analysis underscores the significance of addressing the needs of the elderly (Cluster 2). With retirement as their predominant status, this group necessitates comprehensive support to enhance their quality of life during their golden years. Expanding social activities, cultural events, and artistic programs can enrich their daily experiences, ensuring they remain socially engaged and mentally stimulated. Moreover, physical education activities can promote their well-being, ensuring a healthier and more fulfilling retirement.

Incorporating these findings, governments and non-profit organizations can implement targeted, multi-faceted support services. These initiatives should not only focus on immediate needs but also envision the long-term well-being of each generation. By fostering a sense of community, providing continuous learning opportunities, and promoting active participation, societies can create an inclusive environment where individuals of all ages thrive. These efforts align with the global demographic shift, offering solutions that are not only proactive but also sustainable, ensuring a harmonious coexistence of diverse age groups within our societies.