Task 1: Data loading and cleaning.

a.) Load Data Function

```
import pandas as pd

def load_data(file_path):
    df = pd.read_csv(file_path)
    return df
```

b.) Clean Data Function

```
def clean_data(df):
    if df is None:
        return None

    # Handling missing values
    for column in df.columns:
        if df[column].dtype == 'object':  # Categorical data
            df[column].fillna(df[column].mode()[0], inplace=True)
        else:  # Numerical data
            df[column].fillna(df[column].mean(), inplace=True)

    # Ensure appropriate data types
    for column in df.columns:
        if 'date' in column.lower():
            df[column] = pd.to_datetime(df[column], errors='coerce')
        elif df[column].dtype == 'object':
            try:
                df[column] = df[column].astype(float)
            except ValueError:
                pass

    return df
```

```
def clean_data(df):
    # Filter and handlings missing data (na)
    df.fillna(df.mean(), inplace=True)  # There are different ways of handling missing data. here we replace with the mean.

    # This ensures and determines other types of object types and convert to allow any missing or no value (NaN)
    for col in df.columns:
        if df[col].dtype == 'object':
            df[col] = df[col].astype('category')
        elif df[col].dtype == 'int64':
            df[col] = df[col].astype('float64')

    return df
```

Task 2: Decision Making and Loops

a.) Function that calculates the average age of passengers in a given travel class.

```
def calculate_average_age(df, travel_class):
    # Filter the DataFrame based on the travel class
    df_class = df[df['TravelClass'] == travel_class]

    # Calculate the average age
    avg_age = df_class['Age'].mean()

    return avg_age
```

Take DF and TravelClass as input to return list of names of passengers who are on LoyaltyMembers.

b.) Functions that finds loyalty program members.

```
def find_loyalty_members(df):
    # Filter the DataFrame based on loyalty program membership
    df_loyalty = df[df['LoyaltyMember'] == True]

    # Extract the names of the loyalty program members
    names = df_loyalty['Name'].tolist()

    return names
```

Returns a list of names of employees with experience greater than or equal to the specified years.

```
def find_experienced_employees(df, years):
    # Filter the DataFrame based on experience
    df_experience = df[df['Experience'] >= years]

    # Extract the names of the experienced employees
    names = df_experience['Name'].tolist()

    return names
```

Task 3: Fuctions and Modules

a.) Function that calculates the average age and number of loyalty members for each travel class

```
def get_class_statistics(df):

    # Initialize an empty dictionary to store the results
    class_stats = {}

    # Iterate over each travel class
    for travel_class in df['TravelClass'].unique():
        # Filter the DataFrame based on the travel class
        df_class = df[df['TravelClass'] == travel_class]

        # Calculate the average age
        avg_age = df_class['Age'].mean()

        # Count the number of loyalty members
        loyalty_members = df_class['LoyaltyMember'].sum()

        # Add the results to the dictionary
        class_stats[travel_class] = {'Average Age': avg_age, 'Loyalty Members': loyalty_members}

    return class_stats
```

b) Write a module named passenger_analysis.py and move all the above functions to this module.

Done.

c) Import this module into your main script and call the functions as needed.

Done.

Task 4: Data Visualization with Matplotlib

a)Write a function plot_age_distribution(df) that:

Plots the distribution of ages using a histogram.

b)Save the plot as age_distribution.png.

```
import matplotlib.pyplot as plt
import pandas as pd
from datetime import datetime
```

```
def plot_age_distribution(df):
    plt.figure(figsize=(10, 6))
    plt.hist(df['Age'], bins=20, edgecolor='black')
    plt.title('Age Distribution')
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.savefig('age_distribution.png') #b saving the plot
    plt.show()
```

c)Write a function plot_average_age_by_class(df) that:

Plots the average age by travel class using a bar chart.

d) Save the plot plot as average_age_by_class.png.

```
def plot_average_age_by_class(df):
    # calculate average age by travel class
    avg_ages = df.groupby('TravelClass')['Age'].mean()

    # plot
    plt.figure(figsize=(10, 6))
    avg_ages.plot(kind='bar', edgecolor='black')
    plt.title('Average Age by Travel Class')
    plt.xlabel('Travel Class')
    plt.ylabel('Average Age')
    plt.savefig('average_age_by_class.png')#d saving the plot
    plt.show()
```

Task 5: Data Visualization with Seaborn and Plotly

```
import seaborn as sns
import plotly.express as px
```

a) Function plot age vs loyalty(df). plots a scatter plot using Seaborn.

b) saving plot

```
def plot_age_vs_loyalty(df):
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='Age', y='LoyaltyMember', data=df)
    plt.title('Age vs. Loyalty Membership')
    plt.xlabel('Age')
    plt.ylabel('Loyalty Membership')
    plt.savefig('age_vs_loyalty.png') #b save plot
    plt.show()
```

c) Function plot age distribution by class(df) using boxplot.

d) saving plot

```
def plot_age_distribution_by_class(df):
    current_year = datetime.now().year
    if 'Birthdate' in df.columns and 'TravelClass' in df.columns:
        df['Age'] = current_year - df['Birthdate'].dt.year
        fig = px.box(df, x="TravelClass", y="Age", title="Age Distribution by Travel Class"
        fig.update_traces(quartilemethod="exclusive")
        fig.update_layout(xaxis_title="Travel Class", yaxis_title="Age")
        fig.write_image("age_distribution_by_class.png")
        fig.show()
```

Bonus:

There is an employee_analysis mentioned but this is passenger_analysis.

Let's import everything in.

```
 pip install -U kaleido
```

```
Requirement already satisfied: kaleido in /usr/local/lib/python3.10/dist-packages (0.2.
```

```
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import plotly.io as pio
import plotly.express as px

import passenger_analysis
```

```
#Let's load our CSV file
df = passenger_analysis.load_data('/content/passengers.csv')

print(df)
```

```
     300          Unnamed: 1  Unnamed: 2   Unnamed: 3  Unnamed: 4 Unnamed: 5
0      1            John Doe   5/21/1987  FIRST_CLASS        True      BA249
1      2          Jane Smith  11/12/1980     BUSINESS       False      AA100
2      3            Mia Wong    3/8/1992      ECONOMY        True      BA255
3      4        Noah Johnson   7/19/1995      ECONOMY       False      AA110
4      5      Isabella Rossi   8/30/1982  FIRST_CLASS        True      BA249
..   ...                 ...         ...          ...         ...        ...
295  296        Emily Miller    9/7/1991  FIRST_CLASS       False      BA835
296  297      Brandon Wilson  12/30/1974      ECONOMY        True      AA705
297  298  Stephanie Martinez   2/23/1987     BUSINESS       False      BA840
298  299     David Hernandez   5/18/1995  FIRST_CLASS        True      AA710
299  300        Jessica Clark   8/11/1978      ECONOMY       False      BA845

[300 rows x 6 columns]
```

Headers are unnamed let's change that.

```
df.columns = df.columns.str.replace('Unnamed: 1', 'Name')
df.columns = df.columns.str.replace('Unnamed: 2', 'Birthdate')
df.columns = df.columns.str.replace('Unnamed: 3', 'TravelClass')
df.columns = df.columns.str.replace('Unnamed: 4', 'LoyaltyMember')
df.columns = df.columns.str.replace('Unnamed: 5', 'FlightNumber')
```

Here we will use a function that will determine 'Age' by using the 'Birthdate' and have a new column of 'Age'

```
# Assume df is your DataFrame and 'Birthdate' is the column with dates

# Convert the 'Birthdate' column to datetime format
df['Birthdate'] = pd.to_datetime(df['Birthdate'], errors='coerce')

# Calculate age from Birthdate
def calculate_age(birthdate):
    today = datetime.today()
    age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birth
    return age

# Create a new column in the DataFrame to store the ages
df['Age'] = df['Birthdate'].apply(calculate_age)
```

This is to create our new dataset with headers and ages.

```
# Save the updated DataFrame back to a CSV file
df = df.to_csv('passengers_ha.csv', index=False)
print(df)
```

```
None
```

As we can see there is nothing after we tried see our dataset let's reload with the other CSV file with headers and ages.

```
df = passenger_analysis.load_data('/content/passengers_ha.csv')
print(df)
```

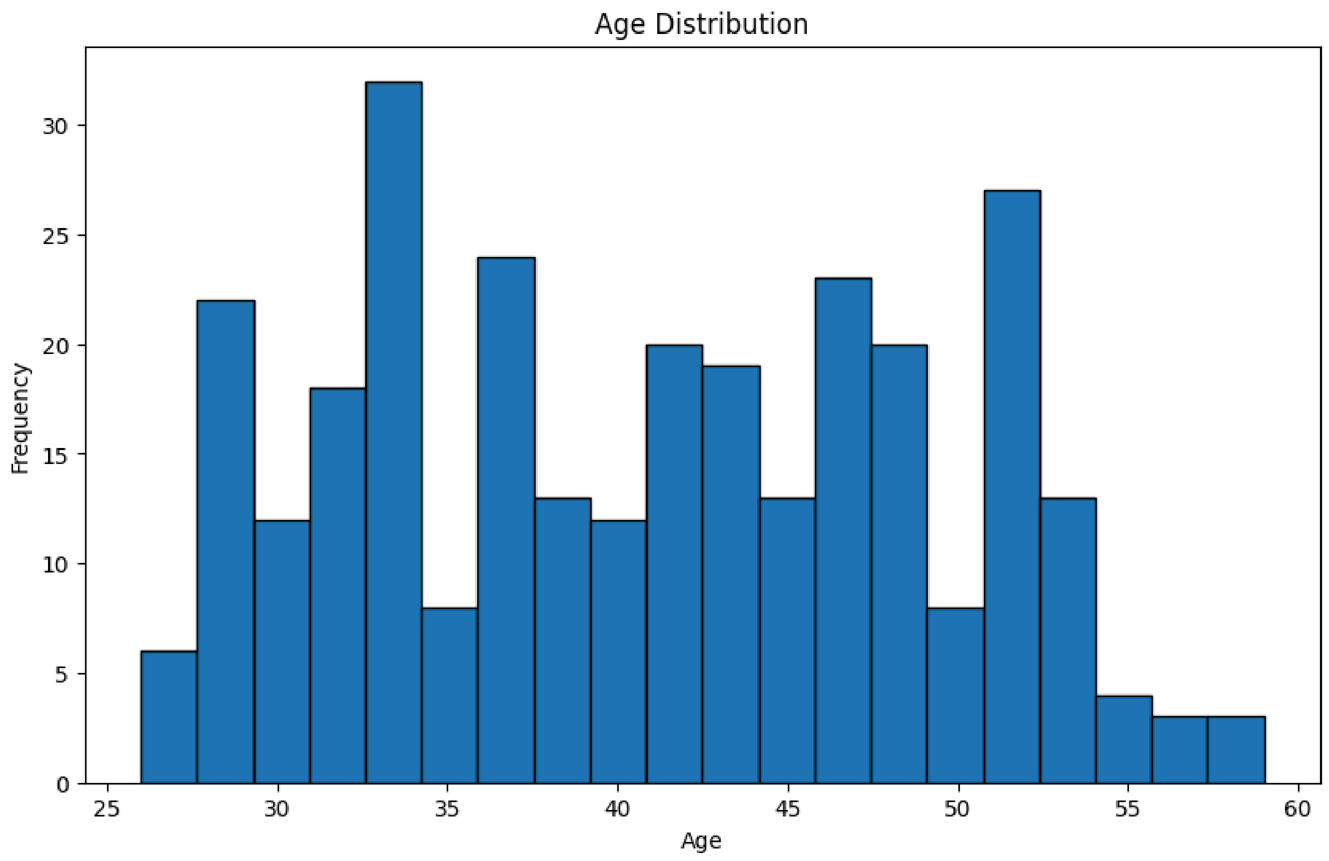
```
     300               Name   Birthdate  TravelClass  LoyaltyMember  \
0      1           John Doe  1987-05-21  FIRST_CLASS           True
1      2         Jane Smith  1980-11-12     BUSINESS          False
2      3           Mia Wong  1992-03-08      ECONOMY           True
3      4       Noah Johnson  1995-07-19      ECONOMY          False
4      5     Isabella Rossi  1982-08-30  FIRST_CLASS           True
..   ...                ...         ...          ...            ...
295  296       Emily Miller  1991-09-07  FIRST_CLASS          False
296  297     Brandon Wilson  1974-12-30      ECONOMY           True
297  298  Stephanie Martinez  1987-02-23     BUSINESS          False
298  299    David Hernandez  1995-05-18  FIRST_CLASS           True
299  300       Jessica Clark  1978-08-11      ECONOMY          False

    FlightNumber  Age
0          BA249   37
1          AA100   43
2          BA255   32
3          AA110   28
4          BA249   41
```

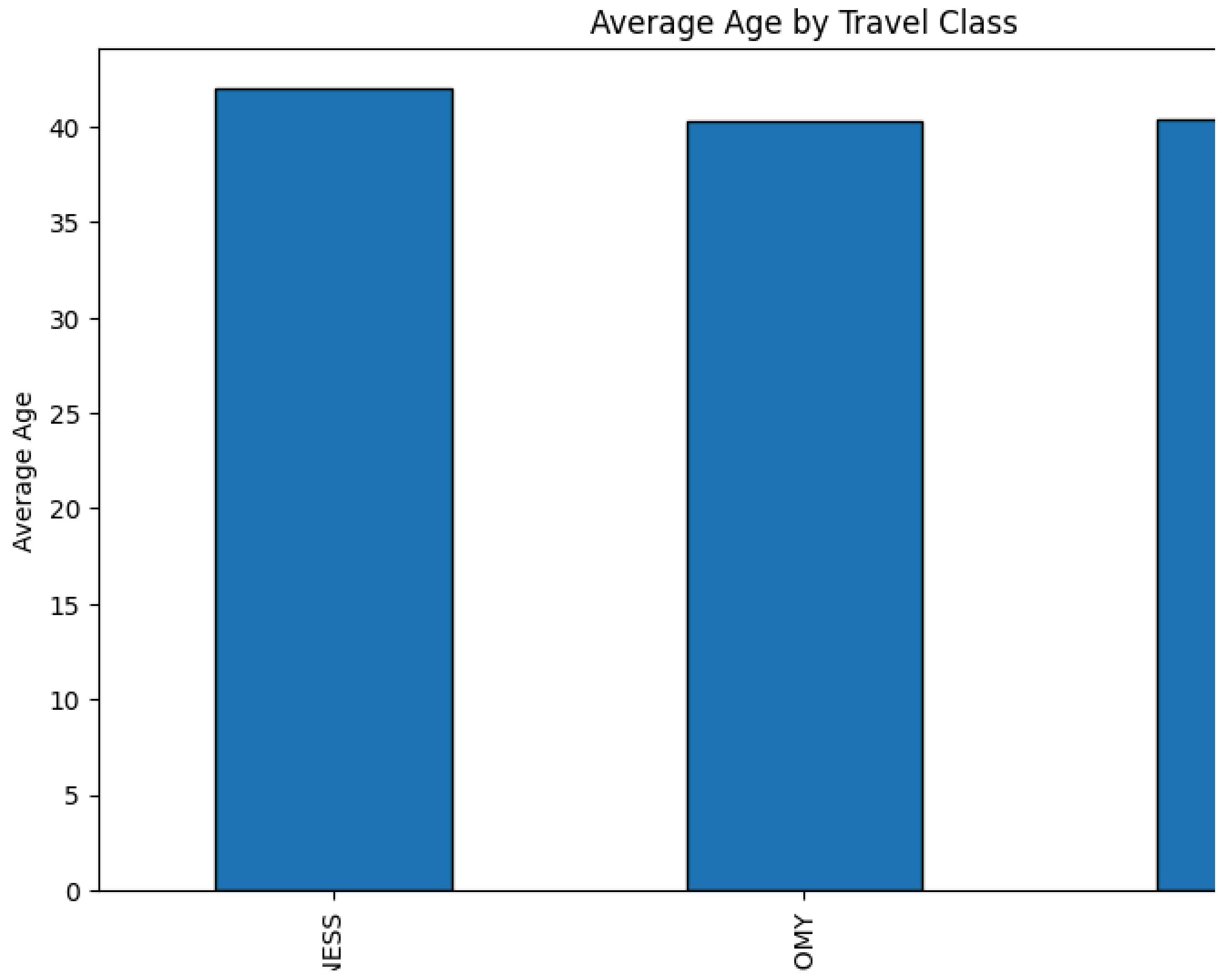
```
..           ...  ...
295        BA835   32
296        AA705   49
297        BA840   37
298        AA710   29
299        BA845   45

[300 rows x 7 columns]
```

```
print(df.dtypes)
```

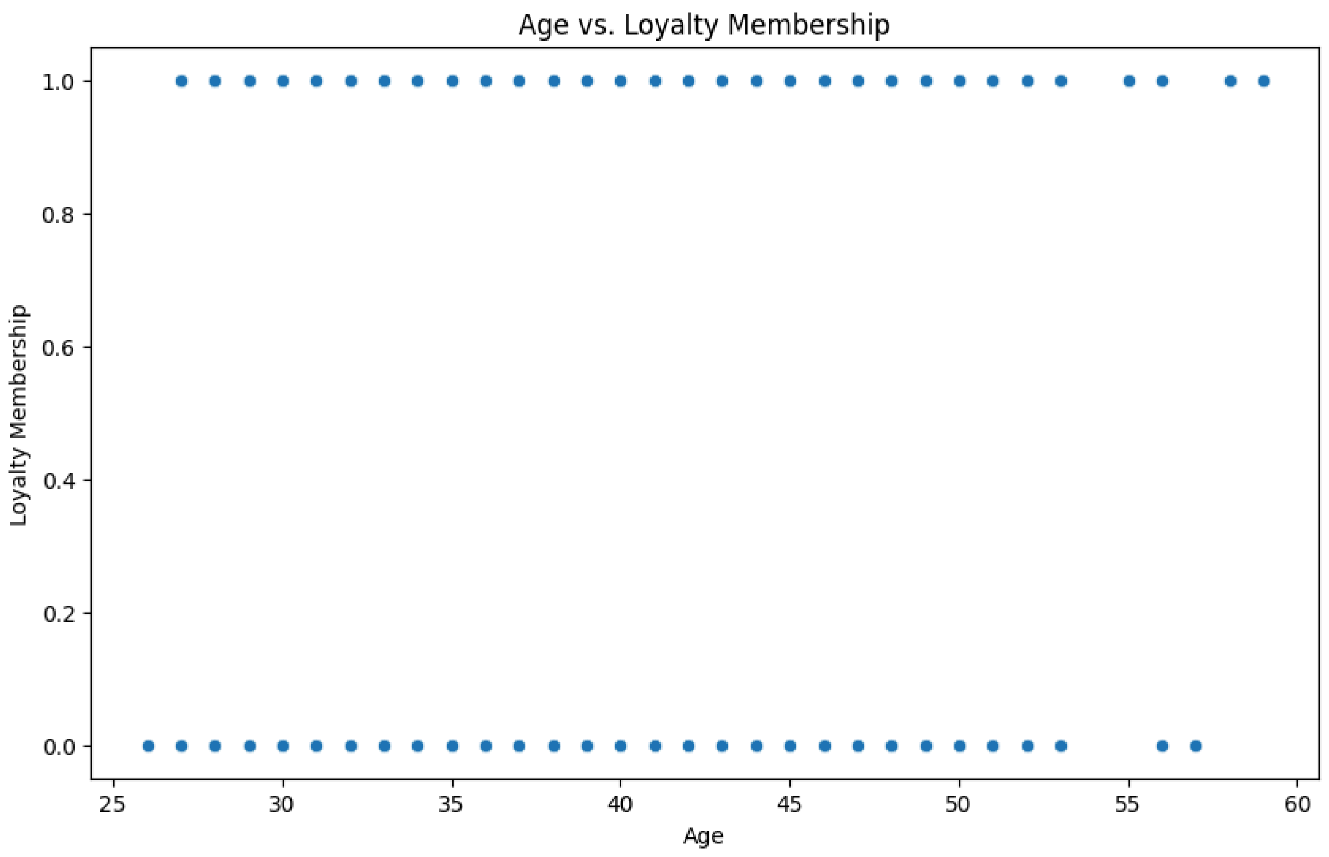
```
300               int64
Name             object
Birthdate        object
TravelClass      object
LoyaltyMember      bool
FlightNumber     object
Age               int64
dtype: object
```

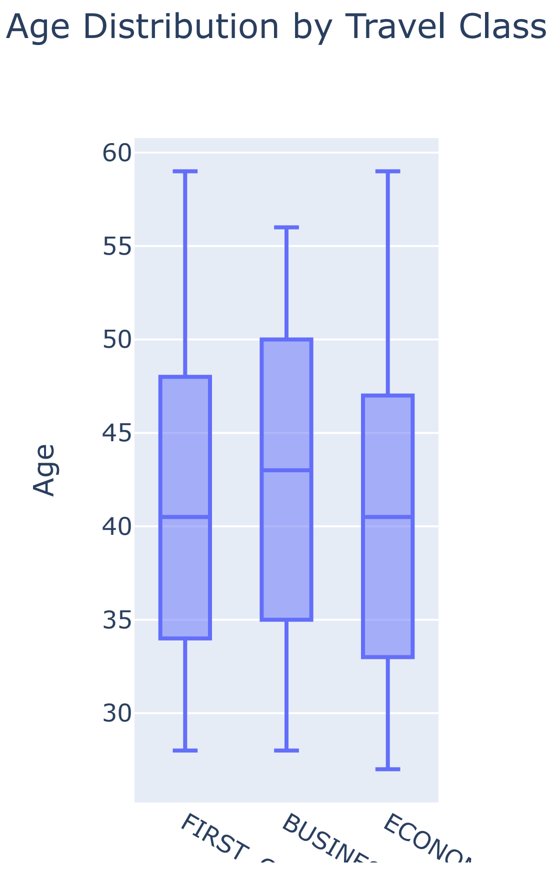
```
import passenger_analysis

def main():
    df = passenger_analysis.load_data('passengers_ha.csv')
    df = passenger_analysis.clean_data(df)

    # Plot age distribution
    plot_age_distribution(df)

    # Plot average age by class
    plot_average_age_by_class(df)

    # Plot age vs. loyalty
    plot_age_vs_loyalty(df)

    # Plot age distribution by class
    plot_age_distribution_by_class(df)

    # Class Stats
    print(passenger_analysis.get_class_statistics(df))

if __name__ == "__main__":
    main()
```


Age Distribution
Frequency
Age
0
5
10
15
20
25
30
25
30
35
40
45
50
55
60


Average Age by Travel Class
Average Age
0
5
10
15
20
25
30
35
40
IESS
OMY

BUSIN
ECON
Travel Class


Age vs. Loyalty Membership
Loyalty Membership
1.0
0.8
0.6
0.4
0.2
0.0
25
30
35
40
45
50
55
60
Age

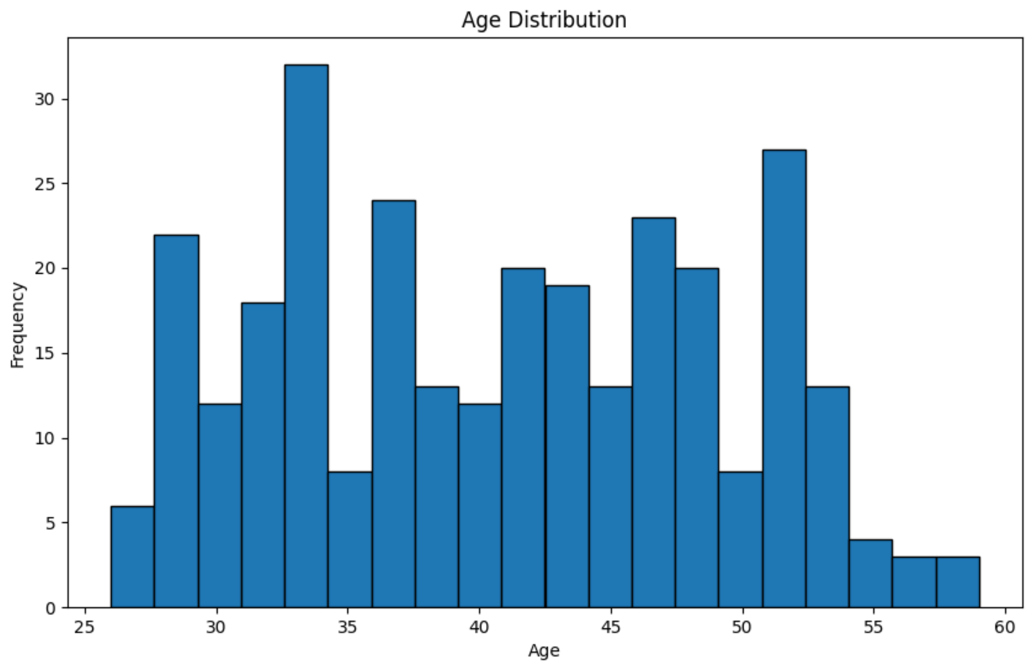

Age Distribution by Travel Class
Age
60
55
50
45
40
35
30
FIRST
BUSINE
ECONO

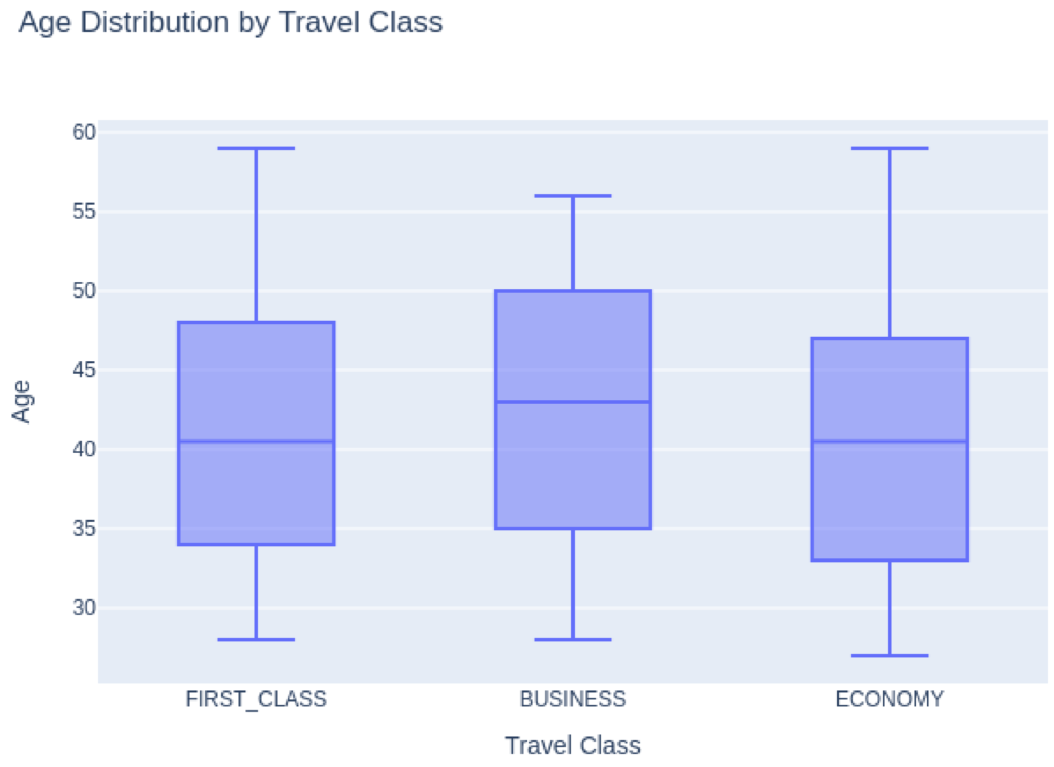
```
{'FIRST_CLASS': {'Average Age': 40.87, 'Loyalty Members': 53}, 'BUSINESS': {'Average Ag
```

This is the result of get_class_statistics

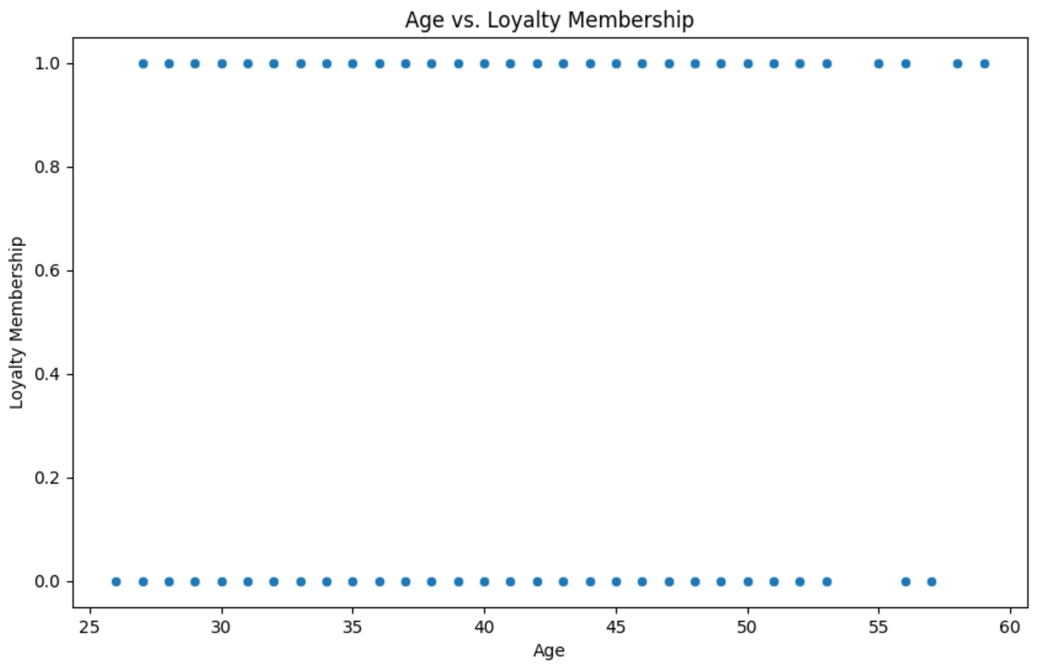
{'FIRST_CLASS': {'Average Age': 40.87, 'Loyalty Members': 53}, 'BUSINESS': {'Average Age': 42.57142857142857, 'Loyalty Members': 47}, 'ECONOMY': {'Average Age': 40.86274509803921, 'Loyalty Members': 54}}

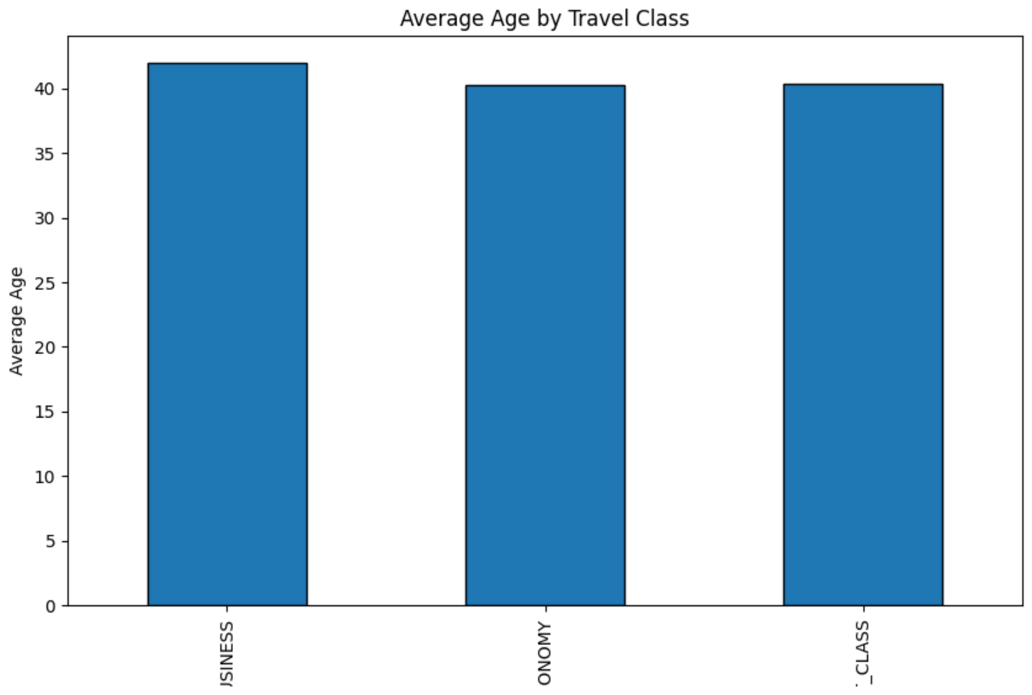
Karina Ponze
Jordyn Jones
Project #1 - Flight
CPS 3320-16





Karina Ponze
Jordyn Jones
Project #1 - Flight
CPS 3320-16


Age vs. Loyalty Membership
Loyalty Membership
Age


Average Age by Travel Class
Average Age
JSINESS
ONOMY
_CLASS