5CS037 - Concepts and Technologies of AI. Exploratory Data Analysis - Part -II. Advance operations with Pandas and Review of Matplotlib.

Prepared By: Siman Giri {Module Leader - 5CS037}

November 26, 2025

1 Instructions

{Disclaimer:Exploratory Data Analysis is designed to be two part exercise and this is Part 2, which mostly focuses on advance operation with pandas for sorting and subsetting of data and applying group-by method on data for exploration and analysis of data.This worksheet also provides a reference on Matplotlib to revise your understanding from Level 4 Computational Mathematics Course.}

This worksheet contains programming exercises on Data cleaning and Data Transformation with pandas based on the material discussed from the slides.This is not a graded exercise and submission are optional but highly recommended as it will be the base of your first assignment.
Please answer the questions below using python in the Jupyter Notebook and follow the guidelines

below: • This worksheet must be completed individually.

• All the solutions must be written in Jupyter Notebook.

• Our Recommendation - Google Colaboratory.

• Dataset used for this session can be downloaded from shared drive.

Figure 1: Pandas.

2 Advance Operations with Pandas.

This Section contains all the sample code from the slides and are here for your reference, you are highly recommended to run all the code with some of the input changed in order to understand the meaning of the operations and also to be able to solve all the exercises from further sections.

- Cautions!!!:
  - This Guide may not contain sample output, as we expect you to re-write the code and observe the output.
  - If found: any error or bugs, please report to your instructor and Module leader. *{Will hugely appreciate your effort.}*

2.1 Sorting and Subsetting:

1. Sorting:

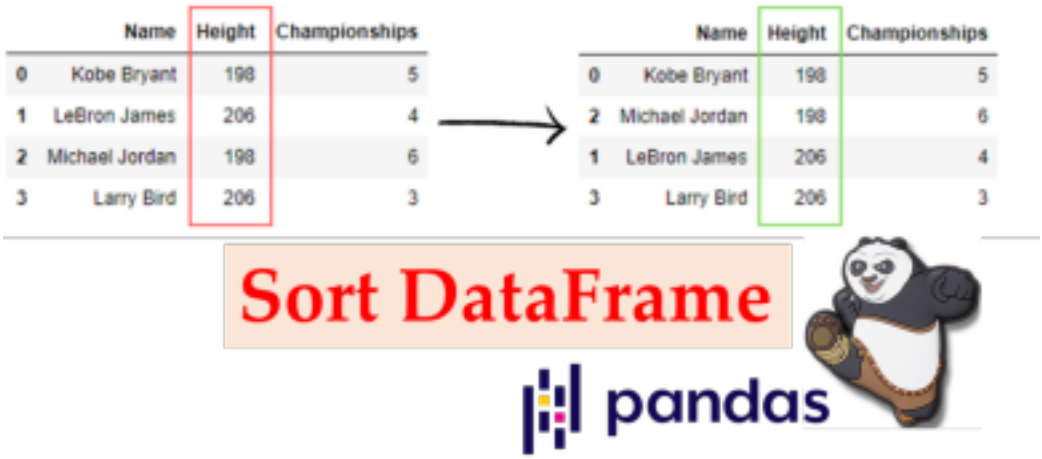

Figure 2: Sorting of Dataframe

Sample Code from Slide - 6 - Sorting Example.

```
import pandas as pd
# Creating a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [24, 19, 22, 25],
        'Score': [88, 92, 85, 95]}
df = pd.DataFrame(data)
print(df.head())
# Example 1: Sort by 'Age' using sort_values()
sorted_by_age = df.sort_values(by='Age')
print(sorted_by_age.head())
# Example 2: Sort by index using sort_index()
sorted_by_index = df.sort_index()
print(sorted_by_index.head())
```


2. Subsetting - Indices:

Sample Code from Slide - 7 - 8 - Subsetting by indices.

```
import pandas as pd
# Creating a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [24, 19, 22, 25],
        'Score': [88, 92, 85, 95]}
df = pd.DataFrame(data)
# 1.Using iloc[]:Accessing rows and columns by index
subset_iloc = df.iloc[1:3, 0:2]
print(subset_iloc)
# 2.Using loc[]:Accessing rows by condition and specific columns
subset_loc = df.loc[df['Age'] > 20, ['Name', 'Score']]
print(subset_loc)
# 3.Using []: Selecting specific columns
subset_brackets = df[['Name', 'Age']]
print(subset_brackets)
```

3. Subsetting by Values - Columns:

Sample Code from Slide - 9 - Subsetting by Columns.

```
import pandas as pd
# Creating a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [24, 19, 22, 25],
        'Score': [88, 92, 85, 95]}
df = pd.DataFrame(data)
# Subsetting a single column
name_column = df['Name']
print(name_column)
# Subsetting multiple columns
name_and_age = df[['Name', 'Age']]
print(name_and_age)
```

4. Subsetting By Rows - *{*aka Filtering*}*:

Sample Code from Slide - 10 - Subsetting by Row.

```
import pandas as pd
# df: Dataframe from slide 09
# Filter rows with a single condition (Age > 20)
filtered_single = df[df['Age'] > 20]
print(filtered_single)
# Filter rows with multiple conditions (Age > 20 and Score > 85)
filtered_multiple = df[(df['Age'] > 20) & (df['Score'] > 85)]
print(filtered_multiple)
```


5. Filtering on Categorical Values:

Sample Code from Slide - 11 - 12 - Filtering on Categorical Values.

```
#Transforming in-built data structures-DataFrame
#Style-1
import pandas as pd
```

```
pd.DataFrame({'Bob': ['I liked it.','It was awful'], 'Sue': ['Pretty good.', 'Bland.']}) #Style-2
pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'], 'Sue': ['Pretty good.', 'Bland.']}, index=['Product A',
                'Product B'])
```

2.2 The Group-By Method:

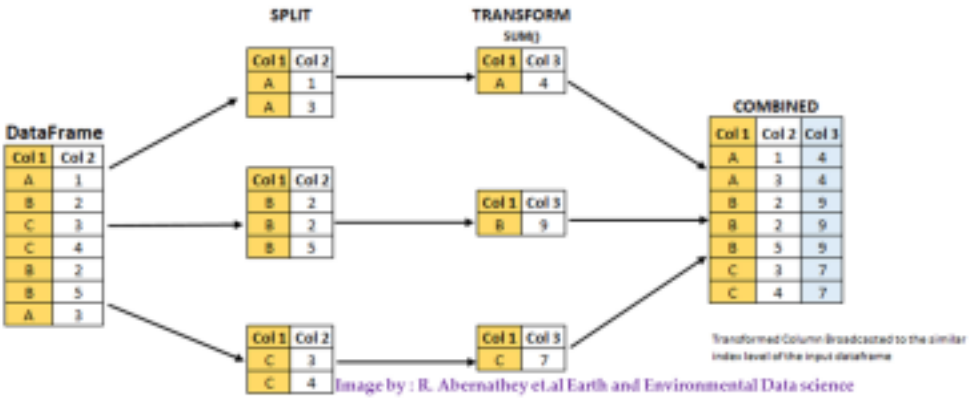

Figure 3: Group By: Split - Apply - Combined.

1. Split - Apply - Combined.

Sample Code from Slide - 16 - Split - Apply{Aggregation} - Combined.

```
import pandas as pd
# Sample DataFrame
data = {'Category': ['A', 'B', 'A', 'B', 'A'],
        'Value': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)
# Aggregation: Calculate mean for each group
grouped = df.groupby('Category')['Value'].mean()
print(grouped)
```

Sample Code from Slide - 17 - Split - Apply{Transformation} - Combined.

```
import pandas as pd
# Sample DataFrame
data = {'Category': ['A', 'B', 'A', 'B', 'A'],
        'Value': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)
# Transformation: Normalize values within each group
df['Normalized'] = df.groupby('Category')['Value'].transform(lambda x: x / x.sum()) print(df)
```


Sample Code from Slide - 18 - Split - Apply{Filtration} - Combined.

```
import pandas as pd
# Sample DataFrame
data = {'Category': ['A', 'B', 'A', 'B', 'A'],
        'Value': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)
# Filtration: Keep groups where sum of values > 60
filtered = df.groupby('Category').filter(lambda x: x['Value'].sum() > 60)
print(filtered)
```

2.3 Summary - Some Advance Operations with Pandas:

Method/Function	Syntax
sort_values()	df.sort_values(by='column_name', ascending=True)
sort_index()	df.sort_index()
head()	df.head(n)
tail()	df.tail(n)
iloc[]	df.iloc[rows, columns]
loc[]	df.loc[condition]
[] (brackets)	df['column_name']
df[df['column_name'] > value]	
groupby()	df.groupby('column_name')
sum()	df.groupby('column_name')['value_column'].sum()
mean()	df.groupby('column_name')['value_column'].mean()
count()	df.groupby('column_name')['value_column'].count()
transform()	df.groupby('column_name')['value_column'].transform(lambda x: x - x.mean())
apply()	df.groupby('column_name').apply(custom_function)
filter()	df.groupby('column_name').filter(lambda x: x['value_column'].sum() > 100)
agg()	df.groupby('column_name').agg('value_column': ['sum', 'mean'])
size()	df.groupby('column_name').size()
isin()	df[df['column_name'].isin([value1, value2])]
pd.cut()	df['new_column'] = pd.cut(df['column_name'], bins, labels)

Table 1: Summary of Common Data Manipulation Methods and Functions in Pandas

Figure 4: Summary Table.

5CS037 Worksheet - 3: Advance Operations with Pandas and Review of Matplotlib. Siman Giri

2.4 Data Visualization with Pandas:

1. Line Plot - Barchart - Histogram - Scatter - Boxplot:

Sample Code from Slide - 27 to 31 - Various Plots.

```
import pandas as pd
import matplotlib.pyplot as plt
# Sample Data
data = {'Month': ['Jan', 'Feb', 'Mar', 'Apr'],
        'Sales': [200, 220, 250, 280]}
df = pd.DataFrame(data)
# Line Plot
df.plot(x='Month', y='Sales', kind='line', marker='o', title='Monthly Sales')
plt.show()
#Bar chart
# Sample Data
data = {'Category': ['A', 'B', 'C'],
        'Values': [10, 20, 15]}
df = pd.DataFrame(data)
df.plot(x='Category', y='Values', kind='bar', title='Category Comparison', color='skyblue') plt.show()
# Sample Data - Histogram
data = {'Scores': [50, 60, 70, 75, 80, 85, 90, 95, 100]}
df = pd.DataFrame(data)
# Histogram
df['Scores'].plot(kind='hist', bins=5, title='Score Distribution', color='orange') plt.show()
# Sample Data - scatter plot
data = {'Height': [150, 160, 170, 180],
```

```
        'Weight': [50, 60, 70, 80]}
df = pd.DataFrame(data)
# Scatter Plot
df.plot(x='Height', y='Weight', kind='scatter', title='Height vs Weight')
plt.show()
# Sample Data
data = {'Scores': [50, 60, 70, 75, 80, 85, 90, 95, 100, 105]}
df = pd.DataFrame(data)
# Box Plot
df.boxplot(column='Scores')
plt.title('Score Distribution')
plt.show()
```

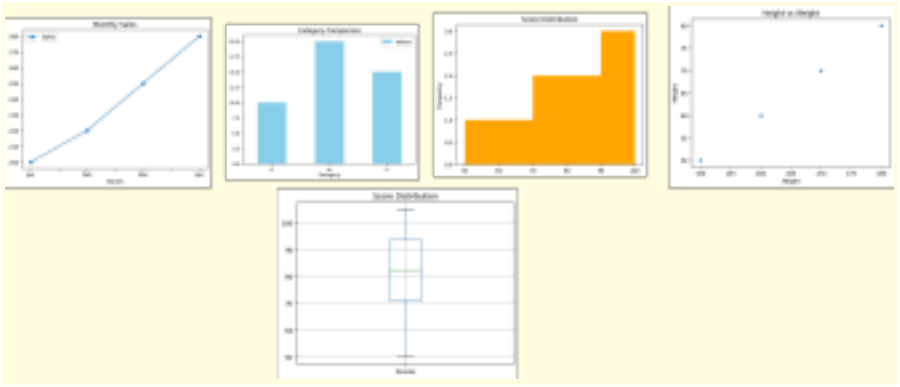

Figure 5: Sample Outputs - Line Chart - Barchart - Histogram - Scatter - Boxplot
5CS037 Worksheet - 3: Advance Operations with Pandas and Review of Matplotlib. Siman Giri

2.5 Data Visualization with Matplotlib:

1. Plotting Your First Figure

- Style:1

Sample Code from Slide - 35 - Plotting Your First Figure.

```
import matplotlib.pyplot as plt
days = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"]
steps_walked = [8934,14902,3409,25672,12300,2023,6890]
plt.plot(steps_walked)
```

- Style:2

Sample Code from Slide - 35 - Plotting Your First Figure.

```
import matplotlib.pyplot as plt
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
steps_walked = [8934, 14902, 3409, 25672, 12300, 2023, 6890]
plt.plot(days, steps_walked)
plt.show()
```

Observe the difference between above two outputs.

2. Anatomy of Matplotlib Figure

When working with data visualization in Python, you'll want to have control over all aspects of your figure. In this section, you'll learn about the main components that make up a figure in Matplotlib. Everything in

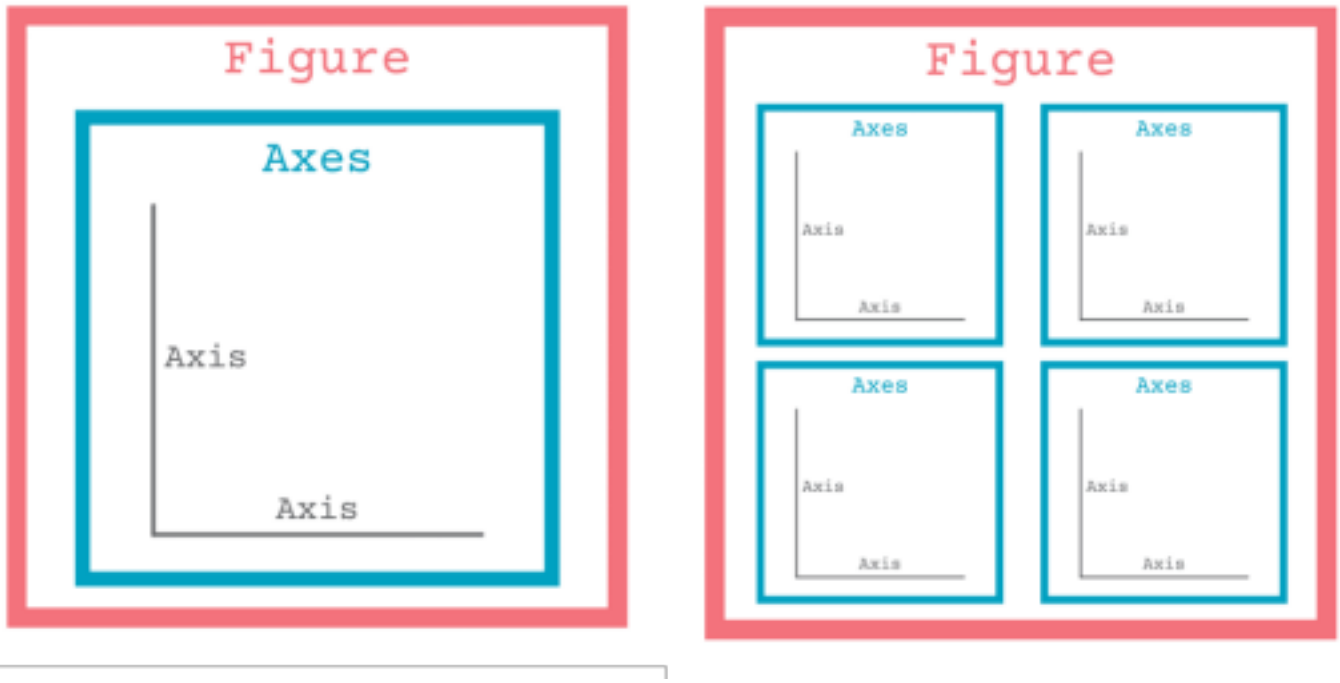

Figure 6: Components of Matplotlib Figure

Python is an object, and therefore, so is a Matplotlib figure. In fact, a Matplotlib figure is made up of several objects of different data types. There are three main parts to a Matplotlib figure:

- Figure: This is the whole region of space that's created when you create any figure. The Figure object is the overall object that contains everything else.
- Axes: An Axes object is the object that contains the x-axis and y-axis for a 2D plot. Each Axes object corresponds to a plot or a graph. You can have more than one Axes object in a Figure, as you'll see later on in this Chapter.
- Axis:An Axis object contains one of the axes, the x-axis or the y-axis for a 2D plot.

Customizing the plots

3. Add a custom marker

Sample Code from Slide - 38 - Add a Custom Marker.

```
import matplotlib.pyplot as plt
days = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"]
steps_walked = [8934,14902,3409,25672,12300,2023,6890]
plt.plot(days, steps_walked, "o")
plt.show()
```

Cautions:Please consult matplotlib documentation for updated version and type of marker available.

4. Adding titles,labels and legends.

Sample Code from Slide - 39 - Adding titles - -

```
import matplotlib.pyplot as plt
days = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"]
steps_walked = [8934,14902,3409,25672,12300,2023,6890]
steps_last_week = [9788,8710,5308,17630,21309,4002,5223]
plt.plot(days, steps_walked, "o-g")
plt.plot(days, steps_last_week, "v--m")
plt.title("Step count | This week and last week")
plt.xlabel("Days of the week")
plt.ylabel("Steps walked")
plt.grid(True)
plt.legend(["This week", "Last week"])
plt.show()
```

Observe the output.

5. Creating a Subplots

Sample Code from Slide - 40 - Creating Subplot.

```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)
t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
plt.subplot(212)
```



```
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

Observe and find what are the arguments for plt.subplot().

3 To - Do - Task

Please Complete all the problem listed below.

3.1 Warm Up Exercises:

1. Sorting and Subsetting:

Complete all following Task:

- Dataset for the Task: "titanic.csv"

Following task is common for all the problem:

1. Load the provided dataset and import in pandas DataFrame.

```
import pandas as pd

# 1. Load the dataset
df = pd.read_csv("Titanic.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

2. Check info of the DataFrame and identify following:

Problem 1 - Sorting:

1. Create a DataFrame called fare that contains only the Fare column of the Titanic dataset. Print the head of the result.

```
fare = df[['Fare']]
print(fare.head())
```

```
      Fare
0   7.2500
1  71.2833
2   7.9250
3  53.1000
4   8.0500
```

2. Create a DataFrame called class age that contains only the Pclass and Age columns of the Titanic dataset, in that order. Print the head of the result.

```
class_age = df[['Pclass', 'Age']]
print(class_age.head())
```

```
   Pclass   Age
0       3  22.0
1       1  38.0
2       3  26.0
3       1  35.0
4       3  35.0
```

3. Create a DataFrame called survived gender that contains the Survived and Sex columns of the Titanic dataset, in that order.Print the head of the result.

```
survived_gender = df[['Survived', 'Sex']]
print(survived_gender.head())
```

```
   Survived     Sex
0         0    male
1         1  female
2         1  female
3         1  female
4         0    male
```

Problem - 2 - Subsetting:

Complete all the following Task:

Subsetting Rows:

1. Filter the Titanic dataset for cases where the passenger's fare is greater than 100, assigning it to fare gt 100.View the printed result.
2. Filter the Titanic dataset for cases where the passenger's class (Pclass) is 1, assigning it to first class.View the printed result.
3. Filter the Titanic dataset for cases where the passenger's age is less than 18 and the passenger is female (Sex is "female"), assigning it to female under 18. View the printed result.

```python
import pandas as pd

# Load dataset
df = pd.read_csv("Titanic.csv")

# 1. Fare > 100
fare_gt_100 = df[df['Fare'] > 100]
print("Passengers with Fare > 100:")
print(fare_gt_100)

# 2. Pclass = 1 (First Class passengers)
first_class = df[df['Pclass'] == 1]
print("\nFirst Class Passengers:")
print(first_class)

# 3. Female passengers under 18
female_under_18 = df[(df['Age'] < 18) & (df['Sex'] == 'female')]
print("\nFemale Passengers Under 18:")
print(female_under_18)
```

```
Passengers with Fare > 100:
     PassengerId  Survived  Pclass  \
27            28         0       1
31            32         1       1
88            89         1       1
118          119         0       1
195          196         1       1
215          216         1       1
258          259         1       1
268          269         1       1
269          270         1       1
297          298         0       1
299          300         1       1
305          306         1       1
306          307         1       1
307          308         1       1
311          312         1       1
318          319         1       1
319          320         1       1
325          326         1       1
```

Subsetting Rows by Categorical variables:

1. Filter the Titanic dataset for passengers whose Embarked port is either "C" (Cherbourg) or "S" (Southampton), assigning the result to embarked c or s. View the printed result.

2. Filter the Titanic dataset for passengers whose Pclass is in the list [1, 2] (indicating first or second class), assigning the result to first second class.View the printed result.

```
                                                  Name     Sex   Age  SibSp  \
...  0                            Braund, Mr. Owen Harris    male  22.0      1
     1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
     2                             Heikkinen, Miss. Laina  female  26.0      0
     3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
     4                           Allen, Mr. William Henry    male  35.0      0
     ..                                               ...     ...   ...    ...
     884                          Sutehall, Mr. Henry Jr    male  25.0      0
     886                           Montvila, Rev. Juozas    male  27.0      0
     887                    Graham, Miss. Margaret Edith  female  19.0      0
     888          Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
     889                           Behr, Mr. Karl Howell    male  26.0      0

          Parch            Ticket     Fare Cabin Embarked
     0        0         A/5 21171   7.2500   NaN        S
     1        0          PC 17599  71.2833   C85        C
     2        0  STON/O2. 3101282   7.9250   NaN        S
     3        0            113803  53.1000  C123        S
     4        0            373450   8.0500   NaN        S
     ..     ...               ...      ...   ...      ...
     884      0   SOTON/OQ 392076   7.0500   NaN        S
     886      0            211536  13.0000   NaN        S
     887      0            112053  30.0000   B42        S
     888      2        W./C. 6607  23.4500   NaN        S
     889      0            111369  30.0000  C148        C

     [812 rows x 12 columns]

     Passengers in First or Second Class:
          PassengerId  Survived  Pclass  \
     1              2         1       1
     3              4         1       1
     6              7         0       1
     9             10         1       2
     11            12         1       1
     ..           ...       ...     ...
     880          881         1       2
     883          884         0       2
     886          887         0       2
     887          888         1       1
```

3.2 Exploratory Data Analysis Practice Exercise - 1.

Warning: Handle missing values in the Age column by filling them with the median age of the dataset before performing the division.)

Answer the following questions from Dataset:

Which passenger had the highest fare paid relative to their age?

To answer the question perform following operations:

1. Add a column to the Titanic dataset, fare per year, containing the fare divided by the age of the

passenger(i.e., Fare/Age).

```
import pandas as pd

# Load dataset
df = pd.read_csv("Titanic.csv")

# Fill missing Age values with median age
median_age = df['Age'].median()
df['Age'] = df['Age'].fillna(median_age)

# Add the new column Fare / Age
df['fare_per_year'] = df['Fare'] / df['Age']

# Display the new column with the first few rows
print(df[['Name', 'Fare', 'Age', 'fare_per_year']].head())
```

```
                                                Name     Fare   Age  \
0                            Braund, Mr. Owen Harris   7.2500  22.0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  71.2833  38.0
2                             Heikkinen, Miss. Laina   7.9250  26.0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  53.1000  35.0
4                           Allen, Mr. William Henry   8.0500  35.0

   fare_per_year
0       0.329545
1       1.875876
2       0.304808
3       1.517143
4       0.230000
```

2. Subset rows where fare per year is higher than 5, assigning this to high fare age. 3. Sort high fare age by descending fare per year, assigning this to high fare age srt. 4. Select only the Name and fare per year columns of high fare age srt and save the result as result. 5. Look at the result.

```python
# 2: Subset rows where fare_per_year > 5
high_fare_age = df[df['fare_per_year'] > 5]

# 3: Sort by descending fare_per_year
high_fare_age_srt = high_fare_age.sort_values(by='fare_per_year', ascending=False)

# 4: Select only Name and fare_per_year
result = high_fare_age_srt[['Name', 'fare_per_year']]
print(result)
```

```
                                                  Name  fare_per_year
305                      Allison, Master. Hudson Trevor     164.728261
297                        Allison, Miss. Helen Loraine      75.775000
386                     Goodwin, Master. Sidney Leonard      46.900000
164                        Panula, Master. Eino Viljami      39.687500
183                           Becker, Master. Richard F      39.000000
..                                                  ...            ...
348          Coutts, Master. William Loch "William"       5.300000
31   Spencer, Mrs. William Augustus (Marie Eugenie)       5.232886
205                          Strom, Miss. Telma Matilda       5.231250
813                 Andersson, Miss. Ebba Iris Alfrida       5.212500
480                     Goodwin, Master. Harold Victor       5.211111

[71 rows x 2 columns]
```

Which adult male passenger (age ≥ 18 and Sex is 'male') paid the highest fare relative to their class?

To answer the question perform following operations:

1. Add a column to the Titanic dataset, fare per class, containing the fare divided by the passenger class i.e. Fare / Pclass.
2. Subset rows where the passenger is male (Sex is "male") and an adult (Age is greater than or equal to 18), assigning this to adult males.
3. Sort adult males by descending fare per class, assigning this to adult males srt.
4. Select only the Name, Age, and fare per class columns of adult males sr and save the result as result.
5. Look at the result.

```python
# 1. Add fare_per_class column
df['fare_per_class'] = df['Fare'] / df['Pclass']

# 2. Subset adult male passengers (Age ≥ 18 and Sex == 'male')
adult_males = df[(df['Age'] >= 18) & (df['Sex'] == 'male')]

# 3. Sort by descending fare_per_class
adult_males_srt = adult_males.sort_values(by='fare_per_class', ascending=False)

# 4. Select only Name, Age, and fare_per_class
result = adult_males_srt[['Name', 'Age', 'fare_per_class']]
print(result)
```

```
                                  Name   Age  fare_per_class
737                 Lesurer, Mr. Gustave J  35.0        512.3292
679  Cardeza, Mr. Thomas Drake Martinez  36.0        512.3292
27         Fortune, Mr. Charles Alexander  19.0        263.0000
438                     Fortune, Mr. Mark  64.0        263.0000
118              Baxter, Mr. Quigg Edmond  24.0        247.5208
..                                    ...   ...             ...
179                  Leonard, Mr. Lionel  36.0          0.0000
732                Knight, Mr. Robert J  28.0          0.0000
822      Reuchlin, Jonkheer. John George  38.0          0.0000
806                Andrews, Mr. Thomas Jr  39.0          0.0000
815                     Fry, Mr. Richard  28.0          0.0000

[519 rows x 3 columns]
```

3.3 Exploratory Data Analysis with Group-by Method Practice Exercise: Based on the dataset Answer the following question:

What percent of the total fare revenue came from each passenger class? To answer the question perform following operation:

1. Calculate the total Fare paid across all passengers in the Titanic dataset.

```python
df['Fare'] = df['Fare'].fillna(0)
total_fare = df['Fare'].sum()
print("Total Fare Revenue:", total_fare)
```

```
Total Fare Revenue: 28693.9493
```

2. Subset for passengers in first class (Pclass is 1) and calculate their total fare.

```
first_class_passengers = df[df['Pclass'] == 1]
total_fare_first_class = first_class_passengers['Fare'].sum()
print("Total Fare from First Class passengers:", total_fare_first_class)
```

```
Total Fare from First Class passengers: 18177.4125
```

3. Do the same for second class (Pclass is 2) and third class (Pclass is 3).

```
# Subset passengers in Second Class
second_class_passengers = df[df['Pclass'] == 2]
total_fare_second_class = second_class_passengers['Fare'].sum()
print("Total Fare from Second Class passengers:", total_fare_second_class)

# Subset passengers in Third Class
third_class_passengers = df[df['Pclass'] == 3]
total_fare_third_class = third_class_passengers['Fare'].sum()
print("Total Fare from Third Class passengers:", total_fare_third_class)
```

```
Total Fare from Second Class passengers: 3801.8417
Total Fare from Third Class passengers: 6714.6951
```

4. Combine the fare totals from first, second, and third classes into a list.

```
# Combine fare totals into a list
fare_totals = [total_fare_first_class, total_fare_second_class, total_fare_third_class]
print("Fare totals by class:", fare_totals)
```

```
Fare totals by class: [np.float64(18177.4125), np.float64(3801.8417), np.float64(6714.6951)]
```

5. Divide the totals for each class by the overall total fare to get the proportion of fare revenue by class.

```
fare_proportion = [total_fare_first_class / total_fare,
                   total_fare_second_class / total_fare,
                   total_fare_third_class / total_fare]

print("Proportion of total fare revenue by class:")
print(f"First Class: {fare_proportion[0]:.2f}")
print(f"Second Class: {fare_proportion[1]:.2f}")
print(f"Third Class: {fare_proportion[2]:.2f}")
fare_percentage = [p * 100 for p in fare_proportion]
print("Percentage of total fare revenue by class:", fare_percentage)
```

```
Proportion of total fare revenue by class:
First Class: 0.63
Second Class: 0.13
Third Class: 0.23
Percentage of total fare revenue by class: [np.float64(63.349287718996564), np.float64(13.24962855496507), np.float64(23.401083726038365)]
```

Based on the dataset Answer the following question:

What percent of the total number of passengers on the Titanic belonged to each age group (e.g., child, adult, senior)?

To answer the question perform following operation:

1. Create a new column, age group, that categorizes passengers into "child" (age < 18), "adult" (age 18{64), and "senior" (age 65 and above).

```
def categorize_age(age):
    if age < 18:
        return 'child'
    elif age < 65:
        return 'adult'
    else:
        return 'senior'
df['age_group'] = df['Age'].apply(categorize_age)
print(df[['Name', 'Age', 'age_group']].head())
```

```
                                                Name   Age age_group
0                            Braund, Mr. Owen Harris  22.0     adult
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  38.0     adult
2                             Heikkinen, Miss. Laina  26.0     adult
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  35.0     adult
4                           Allen, Mr. William Henry  35.0     adult
```

2. Calculate the total number of passengers on the Titanic.

```
total_passengers = len(df)
print("Total number of passengers:", total_passengers)
```

```
Total number of passengers: 891
```

3. Count the number of passengers in each age group.

```
age_group_counts = df['age_group'].value_counts()
print("Number of passengers in each age group:")
print(age_group_counts)
```

```
Number of passengers in each age group:
age_group
adult     767
child     113
senior     11
Name: count, dtype: int64
```

4. Divide the count of each age group by the total number of passengers to get the proportion of passengers in each age group.

```
age_group_proportion = age_group_counts / total_passengers
print("Proportion of passengers in each age group:")
print(age_group_proportion)
```

```
Proportion of passengers in each age group:
age_group
adult     0.860831
child     0.126824
senior    0.012346
Name: count, dtype: float64
```

5. Display the proportion as a percentage.

```
age_group_percentage = age_group_proportion * 100

print("Percentage of passengers in each age group:")
print(age_group_percentage.round(2))
```

```
Percentage of passengers in each age group:
age_group
adult     86.08
child     12.68
senior     1.23
Name: count, dtype: float64
```

The - End