3.1 Warming Up Exercises - Basic Inspection and Exploration:

Problem 1 - Data Read, Write and Inspect:

Complete all following Task:

• Dataset for the Task: "bank.csv"

```
import pandas as pd

# 1. Load the dataset
df = pd.read_csv("/content/drive/MyDrive/Modules_Year_2/Concepts-and-Technologies-of-AI/Dataset/Copy-of-bank.csv")

# 2. Check info of the DataFrame and identify:
# (a) Columns with dtypes=object
object_columns = df.select_dtypes(include='object').columns
print("Columns with dtype=object:", object_columns)

# (b) Unique values of those columns
for col in object_columns:
    print(f"Unique values in column '{col}':")
    print(df[col].unique())

# (c) Check for the total number of null values in each column
print("Null values in each column:")
print(df.isnull().sum())

# 3. Drop all the columns with dtypes=object and store in new DataFrame
df_numeric = df.drop(columns=object_columns)

# Save the DataFrame as 'banknumericdata.csv'
df_numeric.to_csv('banknumericdata.csv', index=False)

# 4. Read 'banknumericdata.csv' and find the summary statistics
df_numeric = pd.read_csv('banknumericdata.csv')
print(df_numeric.describe())  # Summary statistics
```

```
Columns with dtype=object: Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'poutcome', 'y'],
      dtype='object')
Unique values in column 'job':
['management' 'technician' 'entrepreneur' 'blue-collar' 'unknown'
 'retired' 'admin.' 'services' 'self-employed' 'unemployed' 'housemaid'
 'student']
Unique values in column 'marital':
['married' 'single' 'divorced']
Unique values in column 'education':
['tertiary' 'secondary' 'unknown' 'primary']
Unique values in column 'default':
['no' 'yes']
Unique values in column 'housing':
['yes' 'no']
Unique values in column 'loan':
['no' 'yes']
Unique values in column 'contact':
['unknown' 'cellular' 'telephone']
Unique values in column 'month':
['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'jan' 'feb' 'mar' 'apr' 'sep']
Unique values in column 'poutcome':
['unknown' 'failure' 'other' 'success']
Unique values in column 'y':
['no' 'yes']
Null values in each column:
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
                age        balance           day      duration      campaign  \
```

```
count  45211.000000   45211.000000  45211.000000  45211.000000  45211.000000
mean      40.936210    1362.272058     15.806419    258.163080      2.763841
std       10.618762    3044.765829      8.322476    257.527812      3.098021
min       18.000000   -8019.000000      1.000000      0.000000      1.000000
25%       33.000000      72.000000      8.000000    103.000000      1.000000
50%       39.000000     448.000000     16.000000    180.000000      2.000000
75%       48.000000    1428.000000     21.000000    319.000000      3.000000
max       95.000000  102127.000000     31.000000   4918.000000     63.000000

              pdays      previous
count  45211.000000  45211.000000
mean      40.197828      0.580323
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Problem 2 - Data Imputations:

Complete all the following Task:

• Dataset for the Task: "medical_student.csv"

```
import pandas as pd

# 1. Load the dataset and import it into a pandas DataFrame
df_medical = pd.read_csv("/content/drive/MyDrive/Modules_Year_2/Concepts-and-Technologies-of-AI/Dataset/Copy-of-bank.csv")

# 2. Check info of the DataFrame and identify columns with missing (null) values
print("Dataset Information:")
print(df_medical.info())

# Identify columns with missing values
missing_values = df_medical.isnull().sum()
print("\nMissing values in each column:")
print(missing_values)

# 3. Handle missing values by filling them with appropriate techniques
# For each column, choose the imputation method based on the data type and distribution

for col in df_medical.columns:
    if df_medical[col].dtype == 'object':  # Categorical columns
        # Fill missing values with the most frequent value (mode)
        df_medical[col] = df_medical[col].fillna(df_medical[col].mode()[0])
    else:  # Numerical columns
        # If numerical, we will use the mean for imputation
        df_medical[col] = df_medical[col].fillna(df_medical[col].mean())

# Explanation:
# - Categorical columns: Using the most frequent value (mode) is common as it reflects the most typical category.
# - Numerical columns: Using the mean is a standard approach unless there are outliers, in which case, median could be better.

# 4. Check for duplicate values and manage them
duplicates = df_medical.duplicated().sum()
print(f"\nNumber of duplicate rows: {duplicates}")

# Remove duplicates if any
df_medical.drop_duplicates(inplace=True)

# Verify if duplicates were removed
print(f"\nNumber of duplicate rows after removal: {df_medical.duplicated().sum()}")
```

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Student ID      180000 non-null  float64
 1   Age             180000 non-null  float64
 2   Gender          180000 non-null  object
 3   Height          180000 non-null  float64
 4   Weight          180000 non-null  float64
 5   Blood Type      180000 non-null  object
 6   BMI             180000 non-null  float64
 7   Temperature     180000 non-null  float64
 8   Heart Rate      180000 non-null  float64
 9   Blood Pressure  180000 non-null  float64
 10  Cholesterol     180000 non-null  float64
 11  Diabetes        180000 non-null  object
 12  Smoking         180000 non-null  object
dtypes: float64(9), object(4)
```

```
memory usage: 19.8+ MB
None

Missing values in each column:
Student ID        20000
Age               20000
Gender            20000
Height            20000
Weight            20000
Blood Type        20000
BMI               20000
Temperature       20000
Heart Rate        20000
Blood Pressure    20000
Cholesterol       20000
Diabetes          20000
Smoking           20000
dtype: int64

Number of duplicate rows: 12572

Number of duplicate rows after removal: 0
```

3.2 Exercises - Data Cleaning and Transformations with ”Titanic Dataset”:

Dataset Used: "titanic.csv"

Problem - 1:

Create a DataFrame that is subsetted for the columns ’Name’, ’Pclass’, ’Sex’, ’Age’, ’Fare’, and ’Survived’. Retain only those rows where ’Pclass’ is equal to 1, representing first-class passengers. What is the mean, median, maximum value, and minimum value of the ’Fare’ column?

```
import pandas as pd

df_titanic = pd.read_csv('/content/drive/MyDrive/Modules_Year_2/Concepts-and-Technologies-of-AI/Dataset/Copy-of-bank.csv')

df_first_class = df_titanic[['Name', 'Pclass', 'Sex', 'Age', 'Fare', 'Survived']]

df_first_class = df_first_class[df_first_class['Pclass'] == 1]

mean_fare = df_first_class['Fare'].mean()
median_fare = df_first_class['Fare'].median()
max_fare = df_first_class['Fare'].max()
min_fare = df_first_class['Fare'].min()

print(f"Mean Fare: {mean_fare}")
print(f"Median Fare: {median_fare}")
print(f"Max Fare: {max_fare}")
print(f"Min Fare: {min_fare}")
```

```
Mean Fare: 84.1546875
Median Fare: 60.287499999999994
Max Fare: 512.3292
Min Fare: 0.0
```

Problem - 2:

How many null values are contained in the ’Age’ column in your subsetted DataFrame? Once you’ve found this out, drop them from your DataFrame.

```
df_first_class = df_titanic[['Name', 'Pclass', 'Sex', 'Age', 'Fare', 'Survived']]

df_first_class = df_first_class[df_first_class['Pclass'] == 1]

missing_age = df_first_class['Age'].isnull().sum()
print(f"Number of missing values in 'Age' column: {missing_age}")

df_first_class_cleaned = df_first_class.dropna(subset=['Age'])

missing_age_after_drop = df_first_class_cleaned['Age'].isnull().sum()
print(f"Number of missing values in 'Age' column after dropping: {missing_age_after_drop}")
```

```
Number of missing values in 'Age' column: 30
Number of missing values in 'Age' column after dropping: 0
```

Problem - 3:

The ’Embarked’ column in the Titanic dataset contains categorical data representing the ports of embarka- tion:

• 'C' for Cherbourg

• 'Q' for Queenstown

• 'S' for Southampton

Task:

1. Use one-hot encoding to convert the 'Embarked' column into separate binary columns ('Embarked C', 'Embarked Q', 'Embarked S').
2. Add these new columns to the original DataFrame.
3. Drop the original 'Embarked' column.
4. Print the first few rows of the modified DataFrame to verify the changes.

```
df_first_class = df_titanic[['Name', 'Pclass', 'Sex', 'Age', 'Fare', 'Survived', 'Embarked']]

df_first_class = df_first_class[df_first_class['Pclass'] == 1]

df_encoded = pd.get_dummies(df_first_class, columns=['Embarked'], prefix='Embarked')

print(df_encoded.head())
```

```
                                                Name  Pclass     Sex   Age  \
1   Cumings, Mrs. John Bradley (Florence Briggs Th...       1  female  38.0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)       1  female  35.0
6                             McCarthy, Mr. Timothy J       1    male  54.0
11                          Bonnell, Miss. Elizabeth       1  female  58.0
23                       Sloper, Mr. William Thompson       1    male  28.0

        Fare  Survived  Embarked_C  Embarked_Q  Embarked_S
1   71.2833         1        True       False       False
3   53.1000         1       False       False        True
6   51.8625         0       False       False        True
11  26.5500         1       False       False        True
23  35.5000         1       False       False        True
```

Problem - 4:

Compare the mean survival rates ('Survived') for the different groups in the 'Sex' column. Draw a visual- ization to show how the survival distributions vary by gender.

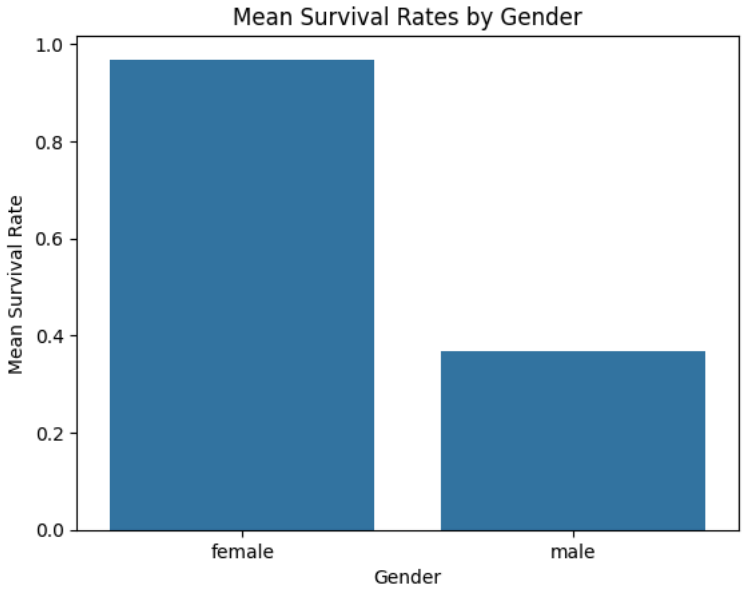
```
import seaborn as sns
import matplotlib.pyplot as plt

df_first_class = df_titanic[['Name', 'Pclass', 'Sex', 'Age', 'Fare', 'Survived']]

df_first_class = df_first_class[df_first_class['Pclass'] == 1]

survival_by_gender = df_first_class.groupby('Sex')['Survived'].mean()

sns.barplot(x=survival_by_gender.index, y=survival_by_gender.values)
plt.title('Mean Survival Rates by Gender')
plt.xlabel('Gender')
plt.ylabel('Mean Survival Rate')
plt.show()
```



Problem - 5:

Draw a visualization that breaks your visualization from Exercise 3 down by the port of embarkation ('Em- barked'). In this instance, compare the ports 'C' (Cherbourg), 'Q' (Queenstown), and 'S' (Southampton).

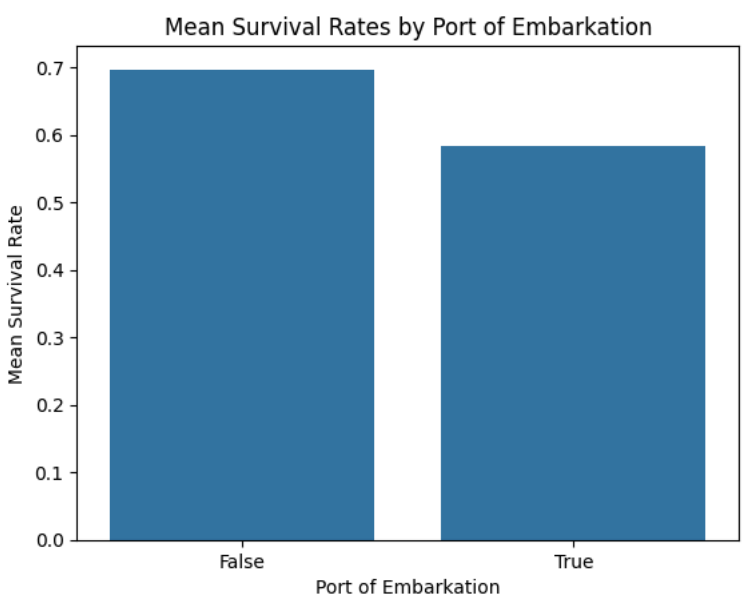
```
df_first_class = df_titanic[['Name', 'Pclass', 'Sex', 'Age', 'Fare', 'Survived', 'Embarked']]

df_first_class = df_first_class[df_first_class['Pclass'] == 1]

df_encoded = pd.get_dummies(df_first_class, columns=['Embarked'], prefix='Embarked')

survival_by_embarked = df_encoded.groupby('Embarked_S')['Survived'].mean()

sns.barplot(x=survival_by_embarked.index, y=survival_by_embarked.values)
plt.title('Mean Survival Rates by Port of Embarkation')
plt.xlabel('Port of Embarkation')
plt.ylabel('Mean Survival Rate')
plt.show()
```



Problem - 6{Optional}:

Show how the survival rates ('Survived') vary by age group and passenger class ('Pclass'). Break up the 'Age' column into five quantiles in your DataFrame, and then compare the means of 'Survived' by class and age group. Draw a visualization using a any plotting library to represent this graphically.

```
df_first_class = df_titanic[['Name', 'Pclass', 'Sex', 'Age', 'Fare', 'Survived']]
```

```
df_first_class = df_first_class[df_first_class['Pclass'] == 1]

df_first_class['Age_Group'] = pd.qcut(df_first_class['Age'], 5)

survival_by_age_class = df_first_class.groupby(['Pclass', 'Age_Group'])['Survived'].mean().reset_index()

pivoted_data = survival_by_age_class.pivot(index='Pclass', columns='Age_Group', values='Survived')

sns.heatmap(pivoted_data, annot=True, cmap='coolwarm')
plt.title('Survival Rates by Age Group and Passenger Class')
plt.xlabel('Age Group')
plt.ylabel('Passenger Class')
plt.show()
```

```
<ipython-input-14-c0a2139fa5d1>:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a futur
  survival_by_age_class = df_first_class.groupby(['Pclass', 'Age_Group'])['Survived'].mean().reset_index()
```