5CS037 - Concepts and Technologies of AI.
Worksheet-2: Exploratory Data Analysis with Pandas -Part-1.

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

November 19, 2025

1 Instructions

{Disclaimer:Exploratory Data Analysis is designed to be two part exercise and this is Part 1, which mostly focuses on use of Pandas for efficient data cleaning and data transformation operation.}

This worksheet contains programming exercises on Data cleaning and Data Transformation with pandas based on the material discussed from the slides.This is a graded exercise and are to be completed on your own and is compulsory to submit.
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: Getting Started with Pandas.

——————– You have to think about one thing: what's that information worth?– (Moneyball) ———————-

2 Getting Started 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 Building Blocks of Pandas:

1. Data Structure - Series:

Sample Code from Slide - 15 - Creating a Simple Series.

```
import
pandas as
pd #
Creating a
simple
Series
data =
[10, 20,
30, 40]
series =
pd.Series(
data)
print(seri
es)
```

2. Data Structure - Index:

Sample Code from Slide - 16 to 18 - Types of Index.

```
# Default Index import pandas as pd
series = pd.Series([10, 20, 30])
print(series.index)
# Output: RangeIndex(start=0, stop=3, step=1) # User Defined:
series = pd.Series([10, 20, 30],index=['a','b','c']) print(series) #
Output:
# a 10
# b 20
# c 30
#datestime index
dates = pd.date_range('2023-01-01', periods=3) series =
pd.Series([10, 20, 30], index=dates) print(series) # Output:
# 2023-01-01 10
# 2023-01-02 20
# 2023-01-03 30
```

Sample Code from Slide - 19 - Acess and Reset Index.

```
#Access print(series.index) # Set or Reset Index
series.index = ['x', 'y', 'z'] # Series
# For DataFrame df = pd.DataFrame({'A': [1, 2]}, index=['row1', 'row2'])
df.reset_index(inplace=True)
# Converts the index into a column
```

3. Data Structure - DataFrames.

Sample Code from Slide - 22 - Creating DataFrames.

```
#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'])
```

4. DataFrames - Loading Data to DataFrames.

Sample Code from Slide - 23 - Loading Data To DataFrames.

```
#Importing Data from file import
pandas as pd
# path to your dataset must be given to built in read_csv("Your path") function.
dataset = pd.read_csv("/data/Week02/bank.csv")
dataset.head() dataset.tail() dataset.info()
# Run the above code and observe the output.
```

3. Data Structure - DataFrames.

Sample Code from Slide - 22 - Creating DataFrames.

```
#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'])
```

5. DataFrames - Writing DataFrames to CSV.

Sample Code from Slide - 24 - Writing DataFrames to CSV.

```
#Importing Data from file import
pandas as pd data = {'Name':
['Alice', 'Bob', 'Charlie'],'City':
['New York', 'San Francisco', 'Los
Angeles']} df =
pd.DataFrame(data) # creating a
DataFrame #Writing DataFrame
to csv.
df.to_csv('output.csv',
index=False) # Run the above
code and observe the output.
```

2.2 Basic Operation on Data: Data Inspection and Exploration:

1. First Data Inspection and Exploration:

Sample Code from Slide - 27 to 28 - First Data Inspection and Exploration.

```
import pandas as pd #
Sample DataFrame data
= {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
} df = pd.DataFrame(data) #
View the first two rows
print(df.head(2)) # View the last
row
print(df.tail(1)) # DataFrame
information print(df.info())
# Summary statistics print(df.describe())
# Check dimensions of the DataFrame
print(f"The DataFrame has {df.shape[0]} rows and {df.shape[1]} columns.")
# Access the 'Age' column print(df['Age'])
# Select rows by numerical index print(df.iloc[0]) # First row #
Select rows by condition print(df.loc[df['Age'] > 30]) # Rows
where Age > 30
```

Understanding `DataFrame.info()`

DataFrame.info():

The `DataFrame.info()` methodprovidesaconcisesummaryofaDataFrame,whichisparticularly usefulforgettinganoverviewofitsstructure.
OutputComponents:

1. ClassType:ShowsthattheobjectisapandasDataFrame.
2. RangeIndex:IndicatesthenumberofrowsintheDataFrame.
3. ColumnInformation:
    - Columnnames
    - Datatypes(int64,float64,object,etc.).
    - Non-nullcounts(usefulforspottingmissingvalues).
4. MemoryUsage:DisplaystheapproximatememoryusageoftheDataFrame.

SampleCodeandOutputExplainationfordf.info().

```
import pandas as pd #
Sample DataFrame data
= {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, None],
    'Salary': [50000, 60000, 55000]
} df = pd.DataFrame(data)
# Check info df.info() #-------------------------------------------------------------------
#************************OUTPUT Explaination**************************
#-------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'> # The object type RangeIndex: 3 entries, 0 to
2 # Number of rows
Data columns (total 3 columns): # Number of columns
 # Column Non-Null Count Dtype
--- ------ ------------- -----
  0    Name 3 non-null object # All rows have values
  1    Age 2 non-null float64 # One missing value
  2    Salary 3 non-null int64 # All rows have values dtypes: float64(1), int64(1), object(1) # Data types memory usage: 200.0+
 bytes # Memory used
```

Understanding DataFrame.describe()

SummaryStatisticswithdf.statistics():

The DataFrame.describe() methodprovidessummarystatisticsfornumericalcolumnsina DataFramebydefault.
OutputComponents:

1. ount:Numberofnon-nullvalues.
2. Mean:Theaveragevalue.
3. StandardDeviation(std):Measureofdatadispersion.
4. Minimum(min):Thesmallestvalue.
5. 25%,50%,75%:Percentilevalues(25th,50th/median,and75th).
6. Maximum(max):Thelargestvalue.

Sample Code and Output Explaination for df.describe().

```
# Generate descriptive statistics import
pandas as pd # Sample DataFrame data =
{
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, None],
    'Salary': [50000, 60000, 55000]
} df = pd.DataFrame(data) # Check summary statistics
df.describe() # Generate descriptive statistics
#---------------------------------------------------------------------#*************************OUTPUT
Explaination*************************
#---------------------------------------------------------------------
                Age Salary
count 2.000000 3.000000 # Non-null values mean
27.500000 55000.000000 # Average values std 3.535534
5000.000000 # Dispersion of values min 25.000000
50000.000000 # Minimum values
25% 26.250000 52500.000000 # 25th percentile 50%
27.500000 55000.000000 # Median
75% 28.750000 57500.000000 # 75th percentile max
30.000000 60000.000000 # Maximum values
```

2. Filtering and Modifying Data:

Sample Code from Slide - 29 - Filtering Rows and Columns.

```
import pandas as pd df =
pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})
# Filter rows where Age > 28 filtered_rows =
df[df['Age'] > 28] print(filtered_rows)
#Select Specific Columns
# Select only 'Name' and 'Salary' columns selected_columns =
df[['Name', 'Salary']] print(selected_columns)
```

Sample Code from Slide - 30 - Droping and Adding a Columns.

```
# Drop the 'Salary' column df_without_salary =
df.drop(columns=['Salary']) print(df_without_salary)
# Drop the row with index 1 (Bob) df_without_row =
df.drop(index=1) print(df_without_row)
# Add a new column for Bonus df['Bonus'] =
df['Salary'] * 0.1 print(df)
```

2.3 Basic Operation on Data - Data Wrangling - Common Data Cleaning operations:

1. Handling Missing Values:

Sample Code from Slide - 34 - Handling Missing values - Adding Missing Values.

```
# Adding Some Missing Values import pandas as pd from sklearn.datasets import load_iris import numpy as np iris = load_iris()
# Load the Iris dataset iris_df = pd.DataFrame(data=np.c_[iris['data'], iris['target']], columns=iris['feature_names'] + [' target'])
np.random.seed(42) # Introduce missing values randomly mask =
np.random.rand(*iris_df.shape) < 0.1 # 10% iris_df[mask] = np.nan
print("Missing Values in Iris Dataset:") print(iris_df.isnull().sum())
```

Sample Code from Slide - 35 - Handling Missing values - Techniques for Filling Missing Values.

```
# Filling missing values with forward fill (ffill), mean, median, and 0 iris_df_ffill = iris_df.ffill()
iris_df_mean = iris_df.fillna(iris_df.mean()) iris_df_median =
iris_df.fillna(iris_df.median())
```

```
iris_df_zero = iris_df.fillna(0) # Expand iris_df with filled columns iris_df_expanded = pd.concat([iris_df,
iris_df_ffill.add_suffix('_ffill'), iris_df_mean.add_suffix('
_mean'),iris_df_median.add_suffix('_median'),iris_df_zero.add_suffix('_zero')], axis=1)
# Display the head of the expanded DataFrame print("\nDataset after Filling
Missing Values:") print(iris_df_expanded.head())
```

2. **Some Common Operation performed for cleaning data.**

Sample Code from Slide - 36 to 37 - Some Common Operations on Data Cleaning.

```
#------------------------------------------------------------------# -----------Trimming Whitespaces:------
--------------------------#------------------------------------------------------------------df =
pd.DataFrame({'Name': ['Alice ', 'Bob '], 'Age': [25, 30]}) df['Name'] = df['Name'].str.strip()
#------------------------------------------------------------------
#-------------Changing Datatype:-----------------------------------#--------------------------------------
---------------------------df = pd.DataFrame({'Age': ['25', '30', '35']}) # Change 'Age' column
data type to integer df['Age'] = df['Age'].astype(int) print(df) #----------------------------------
-------------------------------#-------------Renaming Columns:-----------------------------------#-------
-----------------------------------------------------------
# Rename columns df = pd.DataFrame({'Name': ['Alice', 'Bob'], 'Age': [25, 30]}) df =
df.rename(columns={'Name': 'Full Name', 'Age': 'Years'}) print(df) #----------------------------
-------------------------------------#-------------Removing Duplicates:----------------------------------
#------------------------------------------------------------------
# Remove duplicate rows df = pd.DataFrame({'Name': ['Alice', 'Bob', 'Alice'], 'Age': [25, 30, 25]})
df = df.drop_duplicates() print(df)
```

3. **Data Transformation - DataFrame Reshaping.**

Sample Code from Slide - 39 to 40 - DataFrame Reshaping - Pivot and Melt.

```
#------------------Pivoting--------------------------------------import pandas as pd # Sample DataFrame
data = {'Date': ['2024-01-01', '2024-01-01', '2024-01-02', '2024-01-02'],
      'City': ['Kathmandu', 'Pokhara', 'Kathmandu', 'Pokhara'],
      'Temperature': [15, 18, 16, 19]} df =
pd.DataFrame(data)
# Pivot: Reshape data to show cities as columns
pivoted_df = df.pivot(index='Date', columns='City', values='Temperature') print(pivoted_df) #----------------------------
--------------------------------------------
#--------------------------Melting----------------------------------------
#-------------------------------------------------------------------------
# Melt: Convert wide data back to long format melted_df =
pd.melt(pivoted_df.reset_index(), id_vars=['Date'], var_name='City',
value_name='Temperature')
print(melted_df)
```

4. **Data Transformation - Data Scaling .**

Sample Code from Slide - 41 - Data Transformation - Min-Max Scaling.

```
import pandas as pd from sklearn.datasets import load_iris iris = load_iris() # Load the Iris
dataset iris_df = pd.DataFrame(data=iris['data'], columns=iris['feature_names'])
# Min-Max Scaling using Pandas iris_minmax_scaled = (iris_df - iris_df.min()) / (iris_df.max() -
iris_df.min()) print("Original Iris DataFrame:") print(iris_df.head()) print("\nMin-Max Scaled Iris
DataFrame:") print(iris_minmax_scaled.head()) # Display scaled data
```

5. Data Transformation - Handling Categorical Variables:

Sample Code from Slide - 42 - Handling Categorical Variables - Ordinal or Label Encoding.

```
import pandas as pd
# Sample DataFrame with ordinal categories df = pd.DataFrame({'Category': ['Low', 'Medium',
'High', 'Low', 'High']})
# Ordinal encoding using map ordinal_mapping = {'Low': 1, 'Medium': 2,
'High': 3} df['Category_Ordinal'] = df['Category'].map(ordinal_mapping)
print(df)
```

Sample Code from Slide - 43 - Handling Categorical Variables - One Hot Encoding.

```
import pandas as pd
df_municipalities = pd.DataFrame({'Municipality': ['Kathmandu', 'Bhaktapur', 'Lalitpur', 'Madhyapur
    Thimi', 'Kirtipur']}) one_hot_encoding = pd.get_dummies(df_municipalities['Municipality'],
prefix='Municipality') df_encoded = pd.concat([df_municipalities, one_hot_encoding], axis=1) print(df_encoded)#
Display the result
```

6. Merging and joining DataFrames:

Sample Code from Slide - 44 - Merging and Joining DataFrames - Concatenation.

```
import pandas as pd # Sample
DataFrames df1 =
pd.DataFrame({'A': [1, 2], 'B': [3,
4]}) df2 = pd.DataFrame({'A': [5, 6],
'B': [7, 8]}) # Row-wise
concatenation
combined_rows = pd.concat([df1,
df2], axis=0) print("Row-wise
concatenation:")
print(combined_rows) # Column-
wise concatenation
combined_cols = pd.concat([df1,
df2], axis=1) print("\nColumn-wise
```

```
concatenation:")
print(combined_cols)
```

Sample Code from Slide - 46 - Merging and Joining DataFrames - Merge.

```
# Sample DataFrames df1 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']}) df2
= pd.DataFrame({'ID': [2, 3, 4], 'Score': [85, 90, 88]})
# Inner join inner_merged = pd.merge(df1, df2, on='ID', how='inner')
print("Inner Join:") print(inner_merged) # Left join left_merged =
pd.merge(df1, df2, on='ID', how='left') print("\nLeft Join:")
print(left_merged) # Outer join outer_merged = pd.merge(df1, df2,
on='ID', how='outer') print("\nOuter Join:") print(outer_merged)
```

3 To - Do - Task

Please Complete all the problem listed below.

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"

1. Load the provided dataset and import in pandas DataFrame.
2. Check info of the DataFrame and identify following:
   (a) columns with dtypes=object
   (b) unique values of those columns.
   (c) check for the total number of null values in each column.
3. Drop all the columns with dtypes object and store in new DataFrame, also write the DataFrame in ".csv" with name "banknumericdata.csv"
4. Read "banknumericdata.csv" and Find the summary statistics.

Problem 2 - Data Imputations:

Complete all the following Task:

- Dataset for the Task: "medical_student.csv"

1. Load the provided dataset and import in pandas DataFrame.
2. Check info of the DataFrame and identify column with missing (null) values.
3. For the column with missing values fill the values using various techniques we discussed above. Try to explain why did you select the particular methods for particular column.
4. Check for any duplicate values present in Dataset and do necessary to manage the duplicate items. {Hint: dataset.duplicated.sum()}

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?

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.

Problem - 3:

The 'Embarked' column in the Titanic dataset contains categorical data representing the ports of embarkation:

- '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.

Problem - 4:

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

Problem - 5:

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

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.

To - Do - Task Please Complete all the problem listed below. 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"

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_csv("//content/drive/MyDrive/concept of Technology of AI/Copy of bank.csv")
print(df.head())
```

```
   age           job  marital  education default  balance housing loan  \
0   58    management  married   tertiary      no     2143     yes   no
1   44    technician   single  secondary      no       29     yes   no
2   33  entrepreneur  married  secondary      no        2     yes  yes
3   47   blue-collar  married    unknown      no     1506     yes   no
4   33       unknown   single    unknown      no        1      no   no

   contact  day month  duration  campaign  pdays  previous poutcome   y
0  unknown    5   may       261         1     -1         0  unknown  no
1  unknown    5   may       151         1     -1         0  unknown  no
2  unknown    5   may        76         1     -1         0  unknown  no
3  unknown    5   may        92         1     -1         0  unknown  no
4  unknown    5   may       198         1     -1         0  unknown  no
```

2.Check info of the DataFrame and identify following: (a) columns with dtypes=object

2.Check info of the DataFrame and identify following: (a) columns with dtypes=object

```
col = df.select_dtypes(include='object').columns
col
```

```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'poutcome', 'y'],
      dtype='object')
```

(b) unique values of those columns.

```
for x in col:
    print(df[x].unique())
```

```
['management' 'technician' 'entrepreneur' 'blue-collar' 'unknown'
 'retired' 'admin.' 'services' 'self-employed' 'unemployed' 'housemaid'
 'student']
['married' 'single' 'divorced']
['tertiary' 'secondary' 'unknown' 'primary']
['no' 'yes']
['yes' 'no']
['no' 'yes']
['unknown' 'cellular' 'telephone']
['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'jan' 'feb' 'mar' 'apr' 'sep']
['unknown' 'failure' 'other' 'success']
['no' 'yes']
```

(c) check for the total number of null values in each column.

```
df.isnull().sum()
```

	0
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

3.Drop all the columns with dtypes object and store in new DataFrame, also write the DataFrame in ".csv" with name "banknumericdata.csv"

```
km = df.copy()
df_numeric = km.drop(columns=col)
df_numeric.to_csv("banknumericdata.csv", index=False)
df_numeric.head()
```

	age	balance	day	duration	campaign	pdays	previous
0	58	2143	5	261	1	-1	0
1	44	29	5	151	1	-1	0
2	33	2	5	76	1	-1	0
3	47	1506	5	92	1	-1	0
4	33	1	5	198	1	-1	0

Next steps: Generate code with df_numeric New interactive sheet

4.Read "banknumericdata.csv" and Find the summary statistics.

```
bnk = pd.read_csv("/content/banknumericdata.csv");
bnk.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

```
med = pd.read_csv("/content/drive/MyDrive/concept of Technology of AI/Copy of medical_students_dataset.csv")
med.head()
```

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate	Blood Pressure	Cholesterol	Diabetes	Smoking
0	1.0	18.0	Female	161.777924	72.354947	O	27.645835	NaN	95.0	109.0	203.0	No	NaN
1	2.0	NaN	Male	152.069157	47.630941	B	NaN	98.714977	93.0	104.0	163.0	No	No
2	3.0	32.0	Female	182.537664	55.741083	A	16.729017	98.260293	76.0	130.0	216.0	Yes	No
3	NaN	30.0	Male	182.112867	63.332207	B	19.096042	98.839605	99.0	112.0	141.0	No	Yes
4	5.0	23.0	Female	NaN	46.234173	O	NaN	98.480008	95.0	NaN	231.0	No	No

2.Check info of the DataFrame and identify column with missing (null) values.

2.Check info of the DataFrame and identify column with missing (null) values.

```
med.isnull().sum()
```

	0
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

3.For the column with missing values fill the values using various techniques we discussed above. Try to explain why did you select the particular methods for particular column.

```
med.describe()
```

	Student ID	Age	Height	Weight	BMI	Temperature	Heart Rate	Blood Pressure	Cholesterol
count	180000.000000	180000.000000	180000.000000	180000.000000	180000.000000	180000.000000	180000.000000	180000.000000	180000.000000
mean	49974.042078	26.021561	174.947103	69.971585	23.338869	98.600948	79.503767	114.558033	184.486361
std	28879.641657	4.890528	14.447560	17.322574	7.033554	0.500530	11.540755	14.403353	37.559678
min	1.000000	18.000000	150.000041	40.000578	10.074837	96.397835	60.000000	90.000000	120.000000
25%	24971.750000	22.000000	162.476110	54.969838	17.858396	98.264750	70.000000	102.000000	152.000000
50%	49943.500000	26.000000	174.899914	69.979384	22.671401	98.599654	80.000000	115.000000	184.000000
75%	74986.000000	30.000000	187.464417	84.980097	27.997487	98.940543	90.000000	127.000000	217.000000
max	100000.000000	34.000000	199.998639	99.999907	44.355113	100.824857	99.000000	139.000000	249.000000

The data for age, temperature, and BMI appears approximately normally distributed. The standard deviations are low, and the means and medians are very close to each other. Additionally, the quartiles are evenly spaced. Therefore, I will impute missing values in age, temperature, and BMI using the mean.

```
med['Age'] = med['Age'].fillna(med['Age'].mean())
med['Temperature'] = med['Temperature'].fillna(med['Temperature'].mean())
med['BMI'] = med['BMI'].fillna(med['BMI'].mean())
med.isnull().sum()
```

...

	0
Student ID	20000
Age	0
Gender	20000
Height	20000
Weight	20000
Blood Type	20000
BMI	0
Temperature	0
Heart Rate	20000
Blood Pressure	20000
Cholesterol	20000
Diabetes	20000
Smoking	20000

dtype: int64

The data for height, weight, heart rate, blood pressure, and cholesterol is skewed, with relatively high standard deviations. The quartiles are unevenly spaced, indicating asymmetry in the distribution. Therefore, I will impute missing values for these variables using the median. Additionally, the large difference between the minimum and maximum values suggests the presence of outliers in the data.

```
med.describe()
```

	Student ID	Age	Height	Weight	BMI	Temperature	Heart Rate	Blood Pressure	Cholesterol
count	180000.000000	200000.000000	180000.000000	180000.000000	200000.000000	200000.000000	180000.000000	180000.000000	180000.000000
mean	49974.042078	26.021561	174.947103	69.971585	23.338869	98.600948	79.503767	114.558033	184.486361
std	28879.641657	4.639561	14.447560	17.322574	6.672613	0.474844	11.540755	14.403353	37.559678
min	1.000000	18.000000	150.000041	40.000578	10.074837	96.397835	60.000000	90.000000	120.000000
25%	24971.750000	22.000000	162.476110	54.969838	18.382809	98.306875	70.000000	102.000000	152.000000
50%	49943.500000	26.021561	174.899914	69.979384	23.338869	98.600948	80.000000	115.000000	184.000000
75%	74986.000000	30.000000	187.464417	84.980097	27.255521	98.897102	90.000000	127.000000	217.000000
max	100000.000000	34.000000	199.998639	99.999907	44.355113	100.824857	99.000000	139.000000	249.000000

4.Check for any duplicate values present in Dataset and do necessary to manage the duplicate items. {Hint: dataset.duplicated.sum()}

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?

```
tic = pd.read_csv("/content/drive/MyDrive/concept of Technology of AI/Copy of Titanic-Dataset.csv")
tic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
subset = tic[['Name','Sex','Pclass','Age','Fare','Survived']]
first_class = subset[subset['Pclass'] == 1]
```

```
subset = tic[['Name','Sex','Pclass','Age','Fare','Survived']]
first_class = subset[subset['Pclass'] == 1]
print(first_class['Fare'].describe())
```

```
count    216.000000
mean      84.154687
std       78.380373
min        0.000000
25%       30.923950
50%       60.287500
75%       93.500000
max      512.329200
Name: Fare, dtype: float64
```

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.

```
tic1 = tic.copy()
tic1 = tic1.dropna(subset=['Age'])
tic1.isnull().sum()
```

	0
PassengerId	0
Survived	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:

Use one-hot encoding to convert the 'Embarked' column into separate binary columns ('Embarked C', 'Embarked Q', 'Embarked S'). Add these new columns to the original DataFrame.

```
tic2 = tic.copy()
tic2.head()

tic2["C"] = np.where(tic["Embarked"] == "C", 1, 0)
tic2["Q"] = np.where(tic["Embarked"] == "Q", 1, 0)
tic2["S"] = np.where(tic["Embarked"] == "S", 1, 0)
tic2.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	C	Q	S
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0	0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1	0	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	0	1

3.Drop the original 'Embarked' column.

```
tic2.drop(columns = ["Embarked"])
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	C	Q	S
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	0	0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	1	0	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	0	0	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	0	0	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	0	0	1
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	0	0	1
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	0	0	1
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	1	0	0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	0	1	0

891 rows × 14 columns

4.Print the first few rows of the modified DataFrame to verify the changes.

4.Print the first few rows of the modified DataFrame to verify the changes.

```
tic2.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	C	Q	S
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0	0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1	0	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	0	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0	0	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0	0	1

Problem - 4:

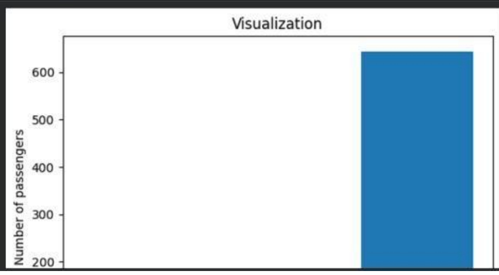
Compare the mean survival rates ('Survived') for the different groups in the 'Sex' column. Draw a visual- ization to show how the survival distributions vary by gender.

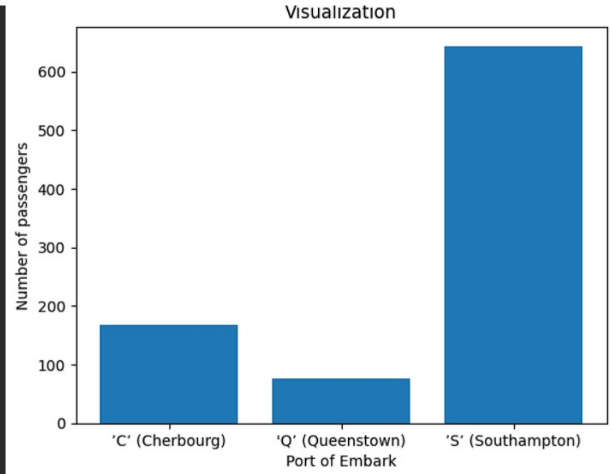
```
# Number of records
print(len(tic2))
# No.of survived people
print(tic2["Survived"].sum())

mean = tic2["Survived"].sum() / len(tic2)
print(mean)
```

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).

```
name = ["'C' (Cherbourg)","'Q' (Queenstown)","'S' (Southampton)"]
value = [tic2["C"].sum(),tic2["Q"].sum(),tic2["S"].sum()]

plt.bar(name,value)
plt.xlabel("Port of Embark")
plt.ylabel("Number of passengers")
plt.title("Visualization")
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

```
tic2["Age"].describe()
```

	Age
count	714.000000
mean	29.699118
std	14.526497
min	0.420000
25%	20.125000
50%	28.000000
75%	38.000000
max	80.000000

dtype: float64

————————— The - End —————————-