

Exploratory Data Analysis Using Python - Lab Manual

Prepared By : Vasudevan T V

List of Programs

0 - Get the Largest Number from a List	2
0 - Remove Duplicates from a List	2
0 - Conversion of Tuple to Dictionary	3
0 - Merging Dictionaries	3
0 - Checking Common Member in Lists	4
0 - Find Earlier Date	4
0 - Subtract 5 Days from Current Date	5
0 - File Copy Operation	5
0 - Capitalise Each Word in a File	5
0 - Search and Replace in a File	6
0 - Count Number of Lines in a File	6
0 - Retrieve Lines Having Two Consecutive 1's	7
0 - Matrix Operations Using Vectorisation	7
1 - Introduction to Python Libraries for Data Analysis(Pandas,NumPy,Matplotlib) . .	8
2 - Data Importing and Basic Exploration Using Pandas	14
3 - Handling Missing Data and Data Imputation in Python (Pandas)	17
9 - Univariate and Bivariate Analysis	24

0 - Get the Largest Number from a List

Write a python program to get the largest number from a list.

Program - Version 1

```
-----  
list=[10,20,30]  
print("List:", list)  
print("The largest element is:", max(list))
```

Program - Version 2

```
-----  
list1=[]  
while True :  
    print("Enter an integer to the list:")  
    element = int(input())  
    list1.append(element)  
    print("Is it end of list?(y/n)")  
    flag = input()  
    if flag == "y" :  
        break  
print("The List =", list1)  
print("Maximum Value in this list:", max(list1))
```

0 - Remove Duplicates from a List

Write a python program to remove duplicates from a list.

Program - Version 1

```
-----  
list1=[20,20,10,30,30]  
print("Duplicated List:", list1)  
list2=[]  
for i in list1:  
    if i not in list2:  
        list2.append(i)  
print("Deduplicated List:", list2)
```

Program - Version 2

```
-----  
list1 = [9,9,5,6,3,4,1,9,2,4,2]
```

```
print("Original List:", list1)
list2 = list(set(list1))
print("Duplicates Removed List : ", list2)
```

0 - Conversion of Tuple to Dictionary

Write a python program to convert a tuple to a dictionary.

Program

```
-----  
tuple = (('Rajesh', 75), ('Vasudevan', 80), ('Thomas', 85))  
print("Tuple=",tuple)  
dictionary = dict(tuple)  
print("Dictionary=",dictionary)
```

0 - Merging Dictionaries

Write a python program to merge two python dictionaries.

*Hint - Use update() method or ** operator or | operator*

Program - Version 1

```
-----  
dict1 = {'a': 1, 'b': 2}  
print("Dictionary 1=",dict1)  
dict2 = {'c': 3, 'd': 4}  
print("Dictionary 2=",dict2)  
  
# Merging dict2 into dict1  
dict1.update(dict2)  
  
print(dict1)
```

Program - Version 2

```
-----  
dict1 = {'a': 1, 'b': 2}  
print("Dictionary 1=",dict1)  
dict2 = {'c': 3, 'd': 4}  
print("Dictionary 2=",dict2)
```

```

# Merging dictionaries
merged_dictionary = {**dict1, **dict2}

print("Merged Dictionary=", merged_dictionary)

Program - Version 3
-----
dict1 = {'a': 1, 'b': 2}
print("Dictionary 1=",dict1)
dict2 = {'c': 3, 'd': 4}
print("Dictionary 2=",dict2)

# Merging dictionaries
merged_dictionary = dict1|dict2

print("Merged Dictionary=", merged_dictionary)

```

0 - Checking Common Member in Lists

Write a python program that takes two lists and returns true if they have at least one common member.

```

Program
-----
list1=[10,20,30]
# list is converted to set to make the elements unique
# and to perform set operations
set1=set(list1)
list2=[40,50,60]
set2=set(list2)
result=set1 & set2
# If there is no common element, then result = set()
if (result== set()):
    print("False")
else :
    print("True")

```

0 - Find Earlier Date

Write a python program to determine which one is the earlier date from the two given dates.

```
Program
-----
date1="2025-01-26"
date2="2025-08-15"
if date1<date2 :
    print("Earlier date is", date1)
elif date1>date2 :
    print("Earlier date is", date2)
else:
    print ("Two dates are equal")
```

0 - Subtract 5 Days from Current Date

Write a python program to subtract 5 days from current date.

```
Program
-----
import datetime
today=datetime.date.today()
no_of_days=datetime.timedelta(days=5)
print(today-no_of_days)
```

0 - File Copy Operation

Write a Python program to open a file and copy the contents to another file.

```
Program
-----
source_file=open("file1.txt", "r")
contents=source_file.read()
source_file.close()
destination_file=open("file2.txt", "w")
destination_file.write(contents)
destination_file.close()
```

0 - Capitalise Each Word in a File

Write a python program to capitalise each word in a file.

Hint - Use str.upper() method

Program

```
-----
file=open("file1.txt","r")
contents=file.read()
new_contents=contents.upper()
file.close()
file=open("file1.txt","w")
file.write(new_contents)
file.close()
```

0 - Search and Replace in a File

Write a Python program to search for a word in a file and replace it with another word.

Hint - Use str.replace("old text","new text") method

Program

```
-----
file=open("file1.txt","r")
contents=file.read()
new_contents=contents.replace("Python", "C")
file.close()
file=open("file1.txt","w")
file.write(new_contents)
file.close()
```

0 - Count Number of Lines in a File

Write a python program to count number of lines in a file.

Program

```
-----
count = 0
file = open("file1.txt", "r")
for line in file:
    count = count + 1
print("Number of Lines:", count)
```

0 - Retrieve Lines Having Two Consecutive 1's

Write a python program to retrieve lines having two consecutive 1's.

Program

```
-----
lines_with_consecutive_ones = []
file = open("file1.txt", "r")
for line in file:
    if '11' in line: # Check for two consecutive '1's
        lines_with_consecutive_ones.append(line.strip()) # remove leading
# and trailing
# white space
print(lines_with_consecutive_ones)
print("Lines containing two consecutive '1's:")
for line in lines_with_consecutive_ones:
    print(line)
```

0 - Matrix Operations Using Vectorisation

Write python programs to perform the following matrix operations using vectorisation.
(Use matrices of order 3 X 3)

1. addition
2. subtraction
3. multiplication
4. scalar multiplication
5. transpose

Program

```
-----
import numpy
matrix1=numpy.matrix([[1,2,3],[1,1,1],[1,1,1]])
matrix2=numpy.matrix([[1,0,0],[0,1,0],[0,0,1]])
print("Matrix 1=\n",matrix1)
print("Matrix 2=\n",matrix2)
```

```

matrix3 = numpy.add(matrix1,matrix2)
matrix4 = numpy.subtract(matrix1,matrix2)
matrix5 = numpy.matmul(matrix1,matrix2)
matrix6 = numpy.transpose(matrix1)
print("Matrix 1 + Matrix 2=\n", matrix3)
print("Matrix 1 - Matrix 2=\n", matrix4)
print("Matrix 1 * Matrix 2=\n", matrix5)
print("2 * Matrix 1=\n", 2*matrix1)
print("Transpose of Matrix 1=\n", matrix6)

```

1 - Introduction to Python Libraries for Data Analysis(Pandas,NumPy,Matplotlib)

You can use the Titanic dataset for practice:

[https://raw.githubusercontent.com/datasciencedojo/datasets /master/titanic.csv](https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv)

This contains details of passengers on the British ship Titanic that sank on 15 April,1912.

PassengerID: A unique identifier for each passenger.

Survived: Survival Status (1 = Survived, 0 = Not Survived)

Pclass: Ticket class (1 = Upper, 2 = Middle, 3 = Lower).

Name: Name of the passenger.

Sex: Gender of the passenger.

Age: Age of the passenger.

SibSp: Number of siblings or spouses aboard.

Parch: Number of parents or children aboard.

Ticket: Ticket number.

Fare: Passenger fare.

Cabin: Cabin number (if available).

Embarked: Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).

1. Create a NumPy array with values from 1 to 50.

Find its: Mean, Maximum, Minimum, Standard deviation

2. Load the Titanic dataset using NumPy Find: Number of passengers, Number of survivors, Average age of passengers

3. Using Matplotlib:

(i) Plot a histogram of the Fare column in Titanic dataset.

- (ii) Plot a bar chart showing the number of passengers by Pclass.
- (iii) Plot a line chart showing numbers from 1 to 10 and their squares.

Program 1

```
-----  
# Imports the numpy module, used for numeric computing  
import numpy  
  
# Create a NumPy array with values from 1 to 50  
# Generates an array with values starting from 1 up to, but not including, 51  
array = numpy.arange(1, 51)  
  
# Calculate the mean  
mean_value = numpy.mean(array)  
  
# Calculate the maximum  
max_value = numpy.max(array)  
  
# Calculate the minimum  
min_value = numpy.min(array)  
  
# Calculate the standard deviation  
std_deviation = numpy.std(array)  
  
# Output the results  
print("Array:", array)  
print("Mean:", mean_value)  
print("Maximum:", max_value)  
print("Minimum:", min_value)  
print("Standard Deviation:", "% .2f" % std_deviation)
```

Output

```
-----  
Mean: 25.5  
Maximum: 50  
Minimum: 1  
Standard Deviation: 14.43
```

Program 2

```
-----  
# Imports the numpy module, used for numeric computing  
import numpy
```

```

# Imports the pandas module, used for data manipulation and analysis
import pandas

# Load Titanic dataset
titanic_data = pandas.read_csv('titanic.csv')

# Converting DataFrame to NumPy array for analysis
data_array = titanic_data.to_numpy()

# data_array.shape[0] returns the number of rows in a numpy array
# data_array.shape[1] returns the number of columns in a numpy array
# data_array.shape returns the number of rows and columns in a numpy array
# Number of passengers
number_of_passengers = data_array.shape[0]

# Number of survivors (assuming 'Survived' column indicates 0 for No
# and 1 for Yes)
# Assuming 'Survived' is in the second column
number_of_survivors = numpy.sum(data_array[:, 1])

# Retrieve the passenger ages from sixth column of the numpy array
ages = data_array[:, 5]

# Convert ages to float
ages = ages.astype(float)

# Calculate the average age while ignoring NaN values
average_age = numpy.nanmean(ages)

# Output results
print("Number of passengers:", number_of_passengers)
print("Number of survivors:", number_of_survivors)
# Prints with precision 2 after decimal point
print("Average age of passengers:", "% .2f" % average_age)

```

Output

```

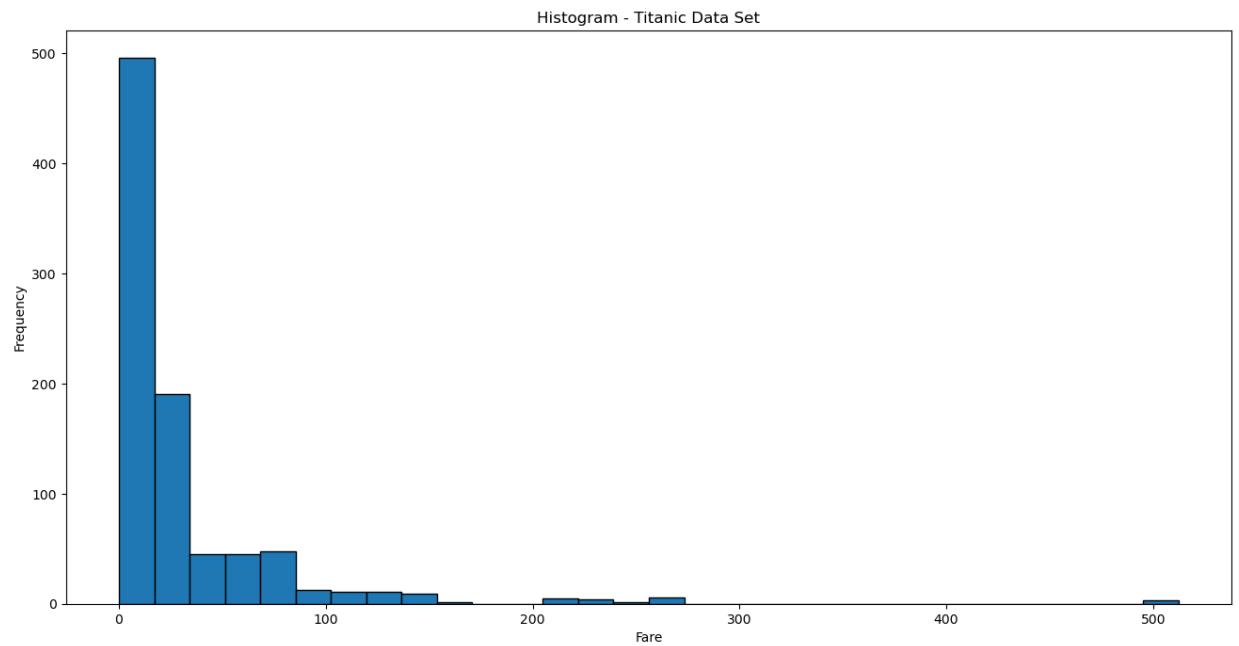
Number of passengers: 891
Number of survivors: 342
Average age of passengers: 29.70

```

Program 3(i)

```
-----  
# import pandas module used for data manipulation and analysis  
import pandas  
# imports pyplot, a module used in the package matplotlib to plot various  
# figures  
from matplotlib import pyplot  
# load the Titanic dataset  
titanic_data = pandas.read_csv('titanic.csv')  
# plots the histogram of fare attribute  
# edge colour is set as black  
# number of bins is set as 30 ( By default bins = 10 )  
titanic_data['Fare'].plot(kind='hist', edgecolor="black", bins=30)  
# sets the title  
pyplot.title("Histogram - Titanic Data Set")  
# sets the x-axis label  
pyplot.xlabel('Fare')  
# shows the histogram  
pyplot.show()
```

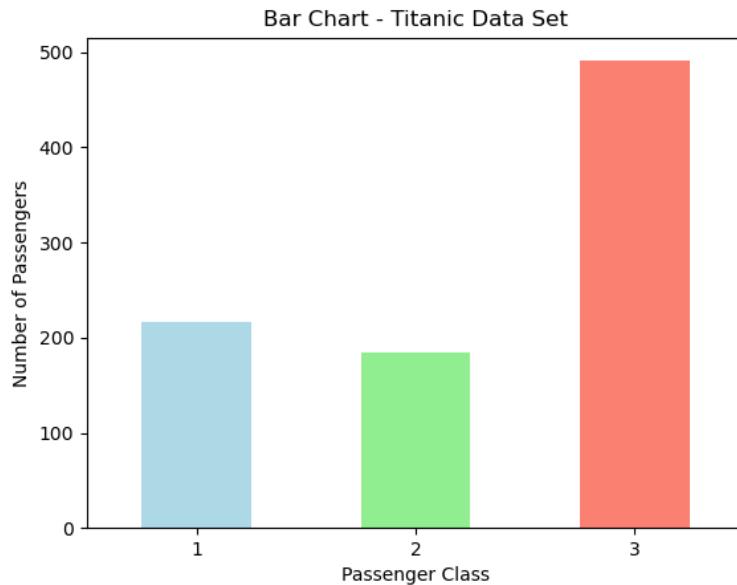
Output



Program 3(ii)

```
-----  
# import pandas module used for data manipulation and analysis  
import pandas  
# imports pyplot, a module used in the package matplotlib to plot various  
# figures  
from matplotlib import pyplot  
# Load the Titanic dataset  
titanic_data = pandas.read_csv('titanic.csv')  
# Count the number of passengers by Pclass sorted by Pclass order  
pclass_counts = titanic_data['Pclass'].value_counts().reindex([1,2,3])  
# plots the bar chart with different colours for each Pclass  
pclass_counts.plot(kind='bar', color=['lightblue', 'lightgreen', 'salmon'])  
# sets the title  
pyplot.title('Bar Chart - Titanic Data Set')  
# sets the x-axis label  
pyplot.xlabel('Passenger Class')  
# sets the y-axis label  
pyplot.ylabel('Number of Passengers')  
# displays x-axis tick labels horizontally( default behaviour is vertical )  
pyplot.xticks(rotation='horizontal')  
# shows the bar chart  
pyplot.show()
```

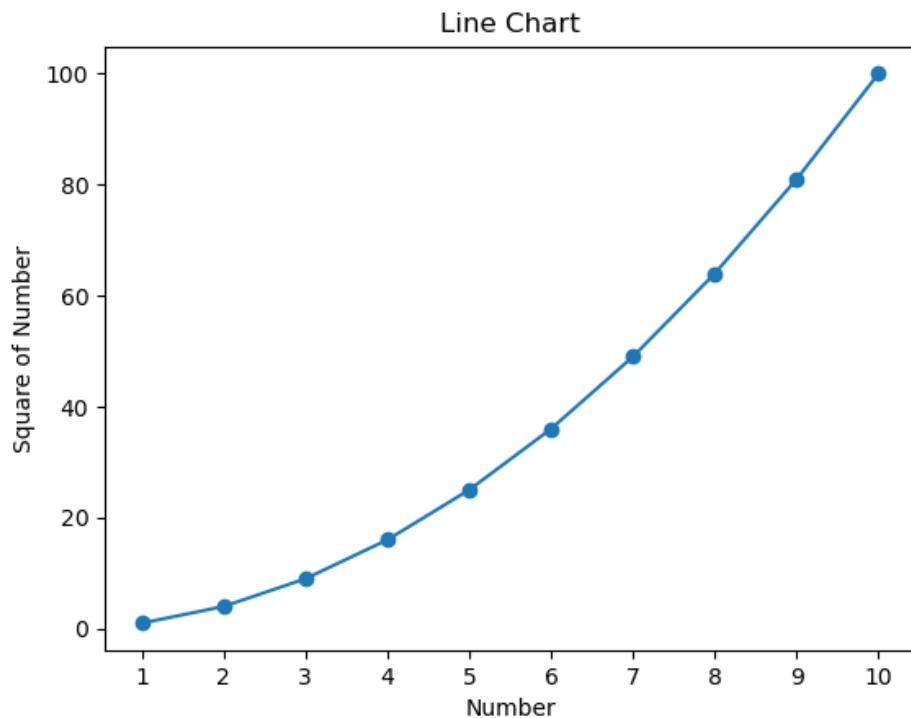
Output



Program 3(iii)

```
-----  
# imports pyplot, a module used in the package matplotlib to plot various  
# figures  
from matplotlib import pyplot  
# Data: numbers from 1 to 10 stored in a list x  
x = list(range(1, 11))  
# Calculate their squares stored in a list y  
y = [i ** 2 for i in x]  
# plots the line chart, adding circular markers at each data point  
pyplot.plot(x, y, marker='o')  
# Adding title and labels  
pyplot.title('Line Chart')  
pyplot.xlabel('Number')  
pyplot.ylabel('Square of Number')  
# Set x-ticks to the numbers  
pyplot.xticks(x)  
# Show the plot  
pyplot.show()
```

Output



2 - Data Importing and Basic Exploration Using Pandas

Load the Titanic dataset.

1. Print (i) Number of rows and columns (ii) Column data types (iii) Number of missing values in each column (iv) Basic descriptive statistics
2. Find (i) The number of passengers in each class (ii) The average age of passengers (iii) The number of survivors

Program 1

```
-----  
# Imports the pandas module, used for data manipulation and analysis  
import pandas  
  
# Load Titanic dataset into a data frame  
titanic_data = pandas.read_csv('titanic.csv')  
  
# titanic_data.shape returns the number of rows and columns in the data  
# frame  
number_of_rows, number_of_columns = titanic_data.shape  
  
# Print number of rows and columns  
print("Titanic data set")  
print("-----")  
print("Number of rows:", number_of_rows)  
print("Number of columns:", number_of_columns)  
  
# Print column data types  
print("Column data types")  
print("-----")  
print(titanic_data.dtypes)  
  
# Calculate and print the number of missing values in each column  
# titanic_data.isnull() creates a DataFrame of the same shape as  
# titanic_data, but with True for missing values and False for non-missing  
# values  
# .sum() counts the number of True values (i.e., missing values) for each  
# column  
missing_values = titanic_data.isnull().sum()  
print("Number of missing values")  
print("-----")
```

```
print(missing_values)

# Print basic descriptive statistics
print("Basic Descriptive Statistics")
print("-----")
descriptive_stats = titanic_data.describe()
print(descriptive_stats)
```

Output

```
-----  
Titanic data set  
-----  
Number of rows: 891  
Number of columns: 12  
Column data types  
-----  
PassengerId      int64  
Survived         int64  
Pclass           int64  
Name             object  
Sex              object  
Age              float64  
SibSp            int64  
Parch            int64  
Ticket           object  
Fare             float64  
Cabin            object  
Embarked         object  
dtype: object  
Number of missing values  
-----  
PassengerId      0  
Survived         0  
Pclass           0  
Name             0  
Sex              0  
Age              177  
SibSp            0  
Parch            0  
Ticket           0  
Fare             0  
Cabin            687  
Embarked         2  
dtype: int64
```

Basic Descriptive Statistics							
	PassengerId	Survived	Pclass	...	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	...	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	...	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	...	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	...	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	...	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	...	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	...	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	...	8.000000	6.000000	512.329200

Program 2

```
-----
# Imports the pandas module, used for data manipulation and analysis
import pandas
# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv('titanic.csv')
# Count the number of passengers in each class ordered by Pclass
class_passenger_count = titanic_data['Pclass'].value_counts().
reindex([1,2,3])
# Calculate the average age
average_age = titanic_data['Age'].mean()
# Calculate the number of survivors
number_of_survivors = titanic_data['Survived'].sum()
# Output the results
print("Number of passengers in each class")
print("-----")
print(class_passenger_count)
# Prints with precision 2 after decimal point
print("Average age of passengers:","%.2f" % average_age)
print("Number of survivors:", number_of_survivors)
```

Output

```
-----
Number of passengers in each class
-----
Pclass
1    216
2    184
3    491
Name: count, dtype: int64
Average age of passengers: 29.70
Number of survivors: 342
```

3 - Handling Missing Data and Data Imputation in Python (Pandas)

Load the Titanic dataset.

1. Display all columns with missing values and the % of missing data.
2. Drop all rows where the Age is missing.
Report the number of rows before and after.
3. (i) Impute missing Age with median.
(ii) Impute missing Embarked with most frequent value (mode).
(iii) Fill missing Cabin with a placeholder like "Unknown".

Program 1

```
-----
# Imports the pandas module, used for data manipulation and analysis
import pandas

# Store the url of the titanic data set
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv(url)

# Calculate missing values and percentage
missing_data = titanic_data.isnull().sum()
# len(titanic_data) returns the number of rows in the data frame
missing_percentage = (missing_data / len(titanic_data)) * 100

# Create a DataFrame to hold the missing data info
missing_info = pandas.DataFrame({'Missing Values': missing_data,
'Percentage': missing_percentage})

# Filter columns with missing values
missing_info = missing_info[missing_info['Missing Values'] > 0]

# Display the results
print(missing_info)
```

Output

	Missing Values	Percentage
Age	177	19.865320
Cabin	687	77.104377
Embarked	2	0.224467

Program 2

```
-----
# Imports the pandas module, used for data manipulation and analysis
import pandas

# Store the url of the titanic data set
url = 'https://raw.githubusercontent.com/datasets/master/titanic.csv'
# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv(url)

# len(titanic_data) returns the number of rows in the data frame
# Get the number of rows before dropping
rows_before = len(titanic_data)

# Drop rows where 'Age' is missing
titanic_data_cleaned = titanic_data.dropna(subset=['Age'])

# Get the number of rows after dropping
rows_after = len(titanic_data_cleaned)

# Set options to display the full DataFrame, otherwise display will be
# truncated
pandas.set_option('display.max_rows', None)          # Show all rows
pandas.set_option('display.max_columns', None)        # Show all columns

# Display the results
print("Titanic data before dropping")
print("-----")
print(titanic_data)
print("-----")
print("Number of rows before dropping = ", rows_before)
print("-----")
print("Titanic data after dropping")
print("-----")
print(titanic_data_cleaned)
print("-----")
print("Number of rows after dropping = ", rows_after)
print("-----")
```

```
Output ( Displaying data frames as truncated )
```

```
Titanic data before dropping
-----
   PassengerId  Survived  Pclass
0            1         0      3
1            2         1      1
2            3         1      3
3            4         1      1
4            5         0      3
..          ...
886        887         0      2
887        888         1      1
888        889         0      3
889        890         1      1
890        891         0      3
[891 rows x 12 columns]
Number of rows before dropping = 891
-----
Titanic data after dropping
-----
   PassengerId  Survived  Pclass
0            1         0      3
1            2         1      1
2            3         1      3
3            4         1      1
4            5         0      3
..          ...
885        886         0      3
886        887         0      2
887        888         1      1
888        889         1      1
889        890         0      3
890        891         0      3
[714 rows x 12 columns]
Number of rows after dropping = 714
```

Program 3(i)

```
# Imports the pandas module, used for data manipulation and analysis
import pandas

# Store the url of the titanic data set
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'

# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv(url)

# Set options to display the full DataFrame, otherwise display will be
# truncated
pandas.set_option('display.max_rows', None)      # Show all rows
pandas.set_option('display.max_columns', None)    # Show all columns

# Display the results
print("Titanic data before imputing Age")
print("-----")
print(titanic_data)
```

```
print("-----")  
  
# Calculate the median age  
median_age = titanic_data['Age'].median()  
  
# Impute missing values with the median age  
# inplace=True modifies the original data frame  
# inplace=False creates a new data frame with changes applied  
titanic_data['Age'].fillna(median_age, inplace=True)  
  
# Display the results  
print("Titanic data after imputing Age")  
print("-----")  
print(titanic_data)  
print("-----")  
  
Output ( Displaying data frames as truncated )  
-----
```

```
Titanic data before imputing Age
```

```
-----  
   PassengerId  Survived  Pclass  ...      Fare Cabin Embarked  
0            1         0       3  ...    7.2500   NaN      S  
1            2         1       1  ...  71.2833  C85      C  
2            3         1       3  ...   7.9250   NaN      S  
3            4         1       1  ...  53.1000  C123     S  
4            5         0       3  ...   8.0500   NaN      S  
..          ...       ...  ...      ...   ...     ...  
886         887         0       2  ...  13.0000   NaN      S  
887         888         1       1  ...  30.0000  B42      S  
888         889         0       3  ...  23.4500   NaN      S  
889         890         1       1  ...  30.0000  C148     C  
890         891         0       3  ...   7.7500   NaN      Q  
[891 rows x 12 columns]
```

```
Titanic data after imputing Age
```

```
-----  
   PassengerId  Survived  Pclass  ...      Fare Cabin Embarked  
0            1         0       3  ...    7.2500   NaN      S  
1            2         1       1  ...  71.2833  C85      C  
2            3         1       3  ...   7.9250   NaN      S  
3            4         1       1  ...  53.1000  C123     S  
4            5         0       3  ...   8.0500   NaN      S  
..          ...       ...  ...      ...   ...     ...  
886         887         0       2  ...  13.0000   NaN      S  
887         888         1       1  ...  30.0000  B42      S  
888         889         0       3  ...  23.4500   NaN      S  
889         890         1       1  ...  30.0000  C148     C  
890         891         0       3  ...   7.7500   NaN      Q  
[891 rows x 12 columns]
```

```
Program 3(ii)
```

```
-----  
# Imports the pandas module, used for data manipulation and analysis  
import pandas  
  
# Store the url of the titanic data set  
url = 'https://raw.githubusercontent.com/datasets/master/titanic.csv'
```

```

/titanic.csv'
# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv(url)

# Set options to display the full DataFrame, otherwise display will be
# truncated
pandas.set_option('display.max_rows', None)          # Show all rows
pandas.set_option('display.max_columns', None)        # Show all columns

# Display the results
print("Titanic data before imputing Embarked")
print("-----")
print(titanic_data)
print("-----")

# Calculate the mode for the 'Embarked' column
# Get the first mode value, as mode() can return multiple values

mode_embarked = titanic_data['Embarked'].mode()[0]

# Impute missing embarked with the most frequent value
titanic_data['Embarked'].fillna(mode_embarked, inplace=True)

# Display the results
print("Titanic data after imputing Embarked")
print("-----")
print(titanic_data)
print("-----")

Output ( Displaying data frames as truncated )
-----

```

```

Titanic data before imputing Embarked
-----
   PassengerId  Survived  Pclass
0            1         0      3
1            2         1      1
2            3         1      3
3            4         1      1
4            5         0      3
..          ...
886        887         0      2
887        888         1      1
888        889         0      3
889        890         1      1
890        891         0      3
[891 rows x 12 columns]
-----
Titanic data after imputing Embarked
-----
   PassengerId  Survived  Pclass
0            1         0      3
1            2         1      1
2            3         1      3
3            4         1      1
4            5         0      3
..          ...
886        887         0      2
887        888         1      1
888        889         0      3
889        890         1      1
890        891         0      3
[891 rows x 12 columns]
-----
```

Program 3(iii)

```

# Imports the pandas module, used for data manipulation and analysis
import pandas

# Store the url of the titanic data set
url = 'https://raw.githubusercontent.com/datasets/master/titanic.csv'
# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv(url)

# Set options to display the full DataFrame, otherwise display will be
# truncated
pandas.set_option('display.max_rows', None)           # Show all rows
pandas.set_option('display.max_columns', None)         # Show all columns

# Display the results
print("Titanic data before imputing Cabin")
print("-----")
print(titanic_data)
print("-----")

# Fill missing values in 'Cabin' with the placeholder 'Unknown'
titanic_data['Cabin'].fillna('Unknown', inplace=True)

# Display the results
```

```

print("Titanic data after imputing Cabin")
print("-----")
print(titanic_data)
print("-----")

```

Output (Displaying data frames as truncated)

Titanic data before imputing Cabin												
	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
..
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

[891 rows x 12 columns]

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

[891 rows x 12 columns]

9 - Univariate and Bivariate Analysis

Using the titanic data set :

1. Find out
 - (a) What is the distribution of passenger ages?
 - (b) What is the most common passenger class?
2. Analyze the relationship between Survived and Sex using a bar chart. What inference can you make?
3. How is the fare distributed across different passenger classes? Which class has the highest median fare?

Program 1 (a)

```

# Imports the pandas module, used for data manipulation and analysis
import pandas

# imports pyplot, a module used in the package matplotlib to plot
# various figures
from matplotlib import pyplot

# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv('titanic.csv')

# Plotting the distribution of passenger ages using a histogram
titanic_data['Age'].plot(kind='hist', color='skyblue', edgecolor="black",
bins=30)

# sets the title
pyplot.title('Distribution of Passenger Ages')

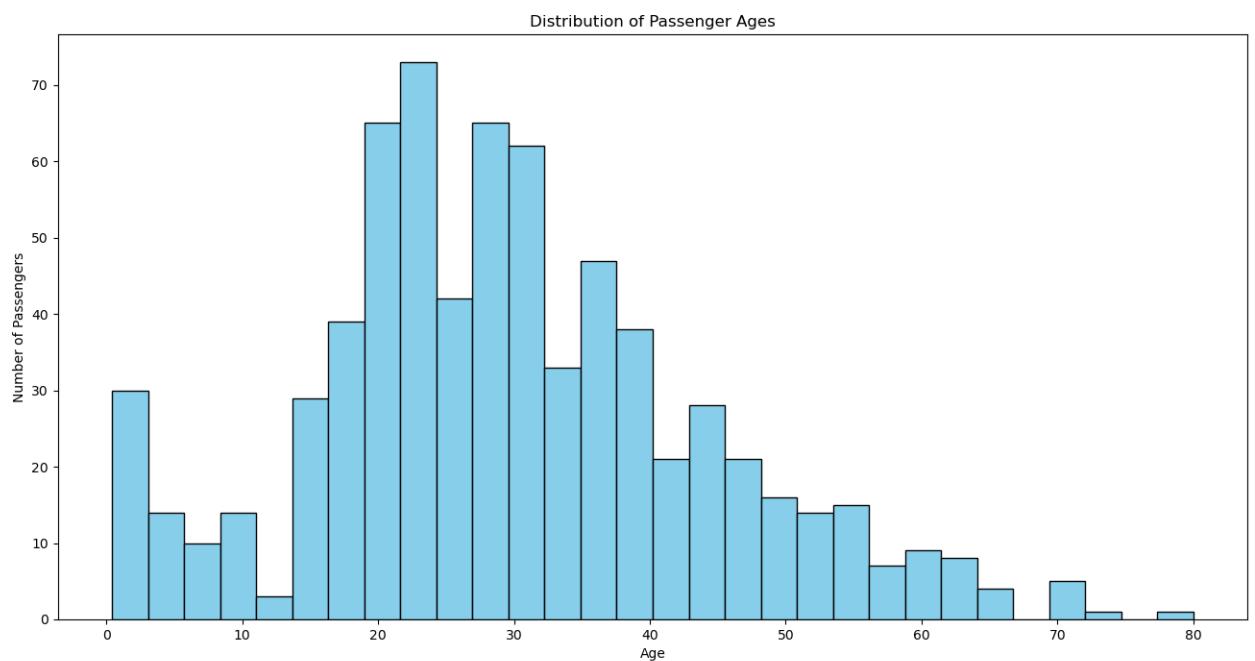
# sets the x-axis label
pyplot.xlabel('Age')

# sets the y-axis label
pyplot.ylabel('Number of Passengers')

# shows the histogram
pyplot.show()

```

Output



Program 1(b)

```
-----  
# Imports the pandas module, used for data manipulation and analysis  
import pandas  
# Load Titanic dataset into a data frame  
titanic_data = pandas.read_csv('titanic.csv')  
# Calculating the most common passenger class  
# mode()[0] get the first mode value, as mode() can return multiple values  
most_common_class = titanic_data['Pclass'].mode()[0]  
# Display the result  
print("Most common passenger class:", most_common_class)
```

Output

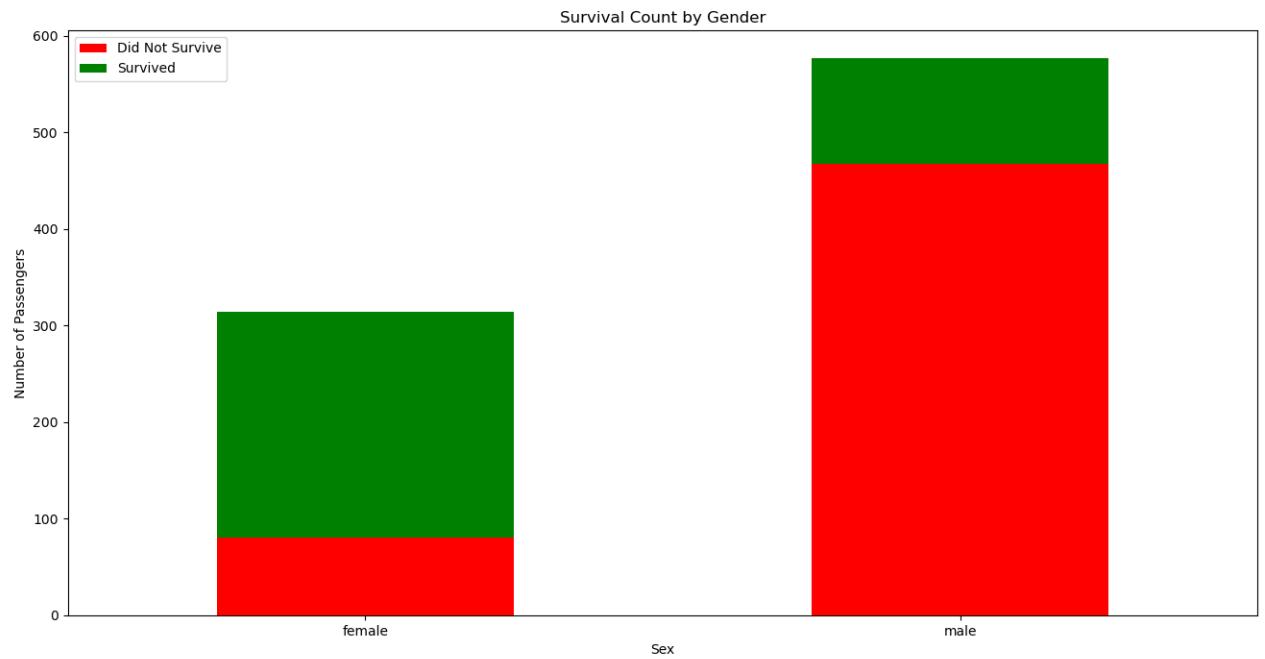
```
-----  
Most common passenger class: 3
```

Program 2

```
-----  
# Imports the pandas module, used for data manipulation and analysis  
import pandas  
# imports pyplot, a module used in the package matplotlib to plot various  
# figures  
from matplotlib import pyplot  
# Load Titanic dataset into a data frame  
titanic_data = pandas.read_csv('titanic.csv')  
# Count the number of survivors and non-survivors by sex  
# The groupby() function groups the data by "Sex" and "Survived"  
# size() counts the occurrences  
# The resulting counts are unstacked to create a Data Frame suitable for  
# plotting  
survival_sex = titanic_data.groupby(['Sex', 'Survived']).size().unstack()  
# Plotting the bar chart as a stacked one  
survival_sex.plot(kind='bar', stacked=True, color=['red', 'green'])  
# sets the title  
pyplot.title('Survival Count by Gender')  
# sets the x-axis label  
pyplot.xlabel('Sex')  
# sets the y-axis label  
pyplot.ylabel('Number of Passengers')  
# displays x-axis tick labels horizontally (default behaviour is vertical)  
pyplot.xticks(rotation='horizontal')  
# sets the legend
```

```
pyplot.legend(['Did Not Survive', 'Survived'])  
# shows the bar chart  
pyplot.show()
```

Output



Inference

During the Titanic evacuation, women were favored compared to men, as lifeboats were prioritised for women

Program 3

```
# Imports the pandas module, used for data manipulation and analysis  
import pandas  
# imports pyplot, a module used in the package matplotlib to plot various  
# figures  
from matplotlib import pyplot  
# Load Titanic dataset into a data frame  
titanic_data = pandas.read_csv('titanic.csv')
```

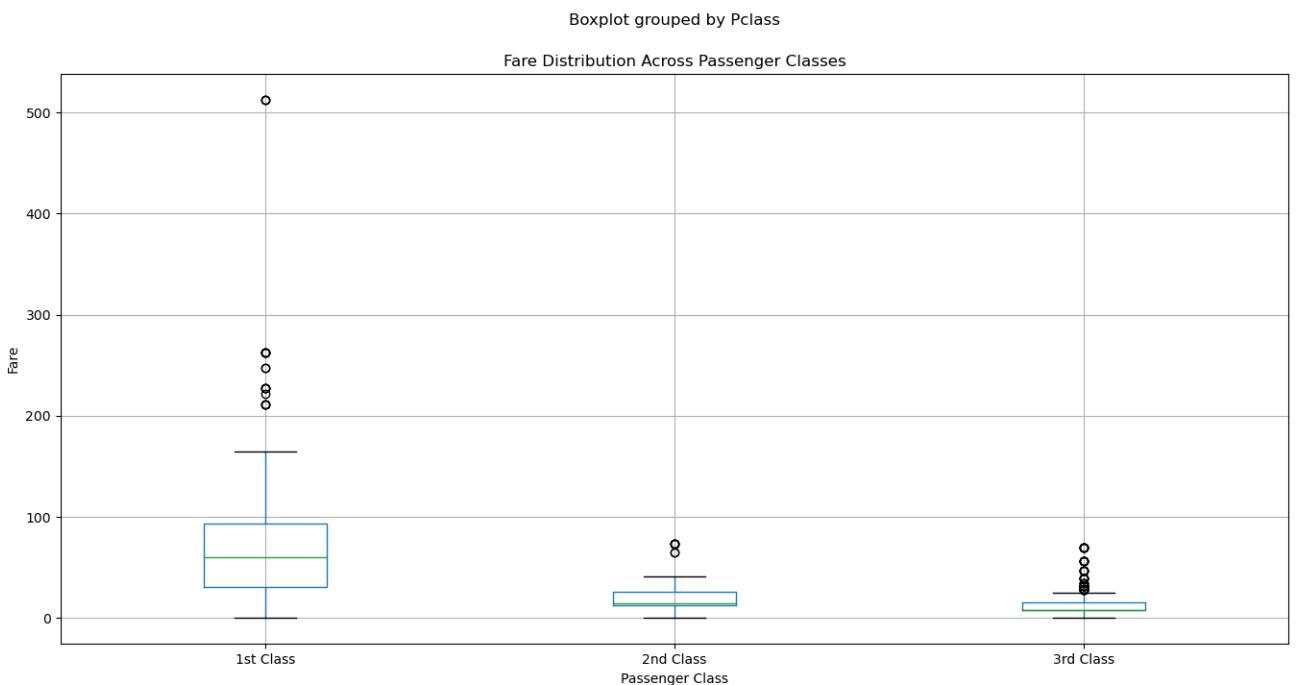
```

# Plotting the distribution of fares
titanic_data.boxplot(column='Fare', by='Pclass')
# sets the title
pyplot.title('Fare Distribution Across Passenger Classes')
# sets the x-axis label
pyplot.xlabel('Passenger Class')
# sets the y-axis label
pyplot.ylabel('Fare')
# sets x-axis ticks along with labels
pyplot.xticks([1, 2, 3], ['1st Class', '2nd Class', '3rd Class'])
# shows the plot
pyplot.show()

# Calculate the median fare for each passenger class
median_fare_by_class = titanic_data.groupby('Pclass')['Fare'].median()
# idxmax() is used to find the index (i.e., the class number) of the
# maximum median fare
print("Passenger Class with highest median fare:", median_fare_by_class.
idxmax())

```

Output



```
Passenger Class with highest median fare: 1
```