

Practical 4

About this unit

Practical 4

Practice Lab Assignment

Unit • 100% completed



Lab Assignment

Unit • 100% completed





4.1.1. Pandas - series creation and manipu...

34/56



Write a Python program that takes a list of numbers from the user, creates a Pandas series from it, and then calculates the mean of even and odd numbers separately using the **groupby** and **mean()** operations.

Input Format:

- The user should enter a list of numbers separated by space when prompted.

Output Format:

- The program should display the mean of even and odd numbers separately.
- Each mean value should be displayed with a label indicating whether it corresponds to even or odd numbers.

Sample Test Cases



Explores

seriesMan...



Submit

Debugger

```
1 import pandas as pd
2
3 # Take inputs from the user to
  create a list of numbers
4 numbers = list(map(int,
  input().split()))
5
6
7 # Create a Pandas series from the
  list of numbers
8 series = pd.Series(numbers)
9 # Grouping by even and odd numbers
  and calculating the mean
10 grouped = series.groupby (series % 2
  == 0).mean()
11
12 # Display the mean of even and odd
  numbers with labels
13 grouped.index = ['Even' if is_even
  else 'Odd' for is_even in
  grouped.index]
14 print("Mean of even and odd
  numbers:")
15 print(grouped)
16
```



< Prev

Reset

Submit

Next >



4.1.2. Dictionary to dataframe

03:40

A dictionary of lists has been provided to you in the editor.
Create a DataFrame from the dictionary of lists and perform the listed operations, then display the DataFrame before and after each manipulation.

Create the DataFrame:

- Convert the dictionary to a Pandas DataFrame.

Add a new row:

- Take inputs from the user for the new row data (name, age).
- Add the new row to the DataFrame.
- Display the DataFrame after adding the new row.

Modify a row:

- Modify a specific row by changing the age. Take the row index and new age value from the user.
- Display the DataFrame after modifying the row.

Delete a row:

- Take the row index to be deleted from the user.
- Remove the specified row.
- Display the DataFrame after deleting the row.

Add a new column:

- Add a column **Gender** with values taken from the user.
- Display the DataFrame after adding the new column.

Modify a column:

- Convert names to uppercase.
- Display the DataFrame after modifying the column.

Delete a column:

- Remove the **Age** column.
- Display the DataFrame after deleting the column.

Sample Test Cases



Explores

dataframe...



Submit

Debugger

```
1 import pandas as pd
2
3 # Provided dictionary of lists
4 data = {
5     'Name': ['Alice', 'Bob', 'Charlie'],
6     'Age': [25, 30, 35],
7 }
8
9 # Convert the dictionary to a DataFrame
10 df = pd.DataFrame(data)
11
12 # Display the original DataFrame
13 print("Original DataFrame:")
14 print(df)
15
16 # Adding a new row
17 new_name = input("New name: ")
18 new_age = int(input("New age: "))
19 df.loc[len(df)] = [new_name, new_age]
20
21 # Display the DataFrame after adding a new row
22 print("After adding a row:\n",df)
23
24 # Modifying a row
25 mod_index = int(input("Index of row to modify: "))
26 new_age = int(input("New age: "))
27 if 0 <= mod_index < len(df):
28     df.at[mod_index, 'Age'] = new_age
29 else:
30     print("Invalid index!")
31
32 # Display the DataFrame after modifying a row
33 print("After modifying a row:")
34 print(df)
35
36 # Deleting a row
37 del_index = int(input("Index of row to delete: "))
38 if 0 <= del_index < len(df):
39     df = df.drop(del_index).reset_index(drop=True)
40 else:
41     print("Invalid index!")
42
43 # Display the DataFrame after deleting a row
44 print("After deleting a row:")
45 print(df)
46
47 # Adding a new column
48 genders = input("Enter genders separated by space: ").split()
49 df['Gender'] = genders
50
51 # Display the DataFrame after adding a new column
52 print("After adding a new column:")
53 print(df)
54
55 # Modifying a column
56 df['Name'] = df['Name'].str.upper()
57
58 # Display the DataFrame after modifying a column
59 print("After modifying a column:")
60 print(df)
61
62 # Deleting a column
```


A dictionary of lists has been provided to you in the editor. Create a DataFrame from the dictionary of lists and perform the listed operations, then display the DataFrame before and after each manipulation.

Create the DataFrame:

- Convert the dictionary to a Pandas DataFrame.

Add a new row:

- Take inputs from the user for the new row data (name, age).
- Add the new row to the DataFrame.
- Display the DataFrame after adding the new row.

Modify a row:

- Modify a specific row by changing the age. Take the row index and new age value from the user.
- Display the DataFrame after modifying the row.

Delete a row:

- Take the row index to be deleted from the user.
- Remove the specified row.
- Display the DataFrame after deleting the row.

Add a new column:

- Add a column **Gender** with values taken from the user.
- Display the DataFrame after adding the new column.

Modify a column:

- Convert names to uppercase.
- Display the DataFrame after modifying the column.

Delete a column:

- Remove the **Age** column.
- Display the DataFrame after deleting the column.

Sample Test Cases



Explores

dataframe...

⌕

Submit

Debugger

```
7
8
9 # Convert the dictionary to a
  DataFrame
10 df = pd.DataFrame(data)
11
12 # Display the original DataFrame
13 print("Original DataFrame:")
14 print(df)
15
16 # Adding a new row
17 new_name = input("New name: ")
18 new_age = int(input("New age: "))
19 df.loc[len(df)] = [new_name, new_age]
20
21 # Display the DataFrame after adding
  a new row
22 print("After adding a row:\n",df)
23
24 # Modifying a row
25 mod_index = int(input("Index of row
  to modify: "))
26 new_age = int(input("New age: "))
27 if 0 <= mod_index < len(df):
28     df.at[mod_index, 'Age'] = new_age
29 else:
30     print("Invalid index!")
31
32 # Display the DataFrame after
  modifying a row
33 print("After modifying a row:")
34 print(df)
35
36 # Deleting a row
37 del_index = int(input("Index of row
  to delete: "))
38 if 0 <= del_index < len(df):
39     df =
    df.drop(del_index).reset_index(drop=T
    rue)
40 else:
41     print("Invalid index!")
42
43 # Display the DataFrame after
  deleting a row
44 print("After deleting a row:")
45 print(df)
46
47 # Adding a new column
48 genders = input("Enter genders
  separated by space: ").split()
49 df['Gender'] = genders
50
51 # Display the DataFrame after adding
  a new column
52 print("After adding a new column:")
53 print(df)
54
55 # Modifying a column
56 df['Name'] = df['Name'].str.upper()
57
58 # Display the DataFrame after
  modifying a column
59 print("After modifying a column:")
60 print(df)
61
62 # Deleting a column
63 df.drop('Age', axis=1, inplace=True)
64
65 # Display the DataFrame after
  deleting a column
66 print("After deleting a column:")
67 print(df)
68
```

4.1.3. Student Information

16:50



Write a program to read a text file containing student information (name, age, and grade) using Pandas. Perform the following tasks:

- Display the first five rows of the data frame.
- Calculate the average age of the students(limit the average age up to 2 decimal places).
- Filter out the students who have a grade above a certain threshold(consider the threshold grade is 'B').

Note:

Refer to the displayed test cases for better understanding.

Sample Test Cases



Explorer

studentinf...

studentdat...

Submit

Debugger

```
1 import pandas as pd
2
3 # Read the text file into a DataFrame
4 file = input()
5 data = pd.read_csv(file, sep="\s+",
6                     header=None, names=["Name", "Age",
7                                         "Grade"])
8
9 print("First five rows:")
10 print(data.head())
11 # write your code here..
12 average_age =
13 round(data["Age"].mean(), 2)
14 print("Average age:", average_age)
15
16 print("Students with a grade up to
17 B")
18 filtered_students =
19 data[data["Grade"].isin(['A+', 'A',
20                           'B'])]
21 print(filtered_students)
```



< Prev

Reset

Submit

Next >



4.2.1. Month with the Highest Total Sales

07/26



- Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:
- The CSV file contains the columns: Date, Product, Quantity, Price, and City.
 - Group the data by Month and calculate the total sales for each month.
 - Find the month with the highest total sales and display it.
 - Also, display the total sales for the best month.

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:

The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases



Explores



monthForS...



sales_data...



Submit

Debugger

```
1 import pandas as pd
2
3 # Prompt the user for the file name
4 file_name = input()
5
6 # Load the data
7 df = pd.read_csv(file_name)
8
9 df['Date'] =
pd.to_datetime(df['Date'])
10 df['Month'] =
df['Date'].dt.to_period('M').astype(s
tr)
11 df['Total'] = df['Quantity'] *
df['Price']
12
13 monthly_sales = df.groupby('Month')
['Total'].sum()
14
15 # Find the month with the highest
total sales
16 best_month = monthly_sales.idxmax()
17 highest_sales = monthly_sales.max()
18
19 print(f"Best month: {best_month}")
20 print(f"Total sales:
${highest_sales:.2f}")
21
```



4.2.2. Best Selling Product

21/03



Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the columns: Date, Product, Quantity, Price, and City.
- Find the product that sold the most in terms of quantity sold.
- Display the product that sold the most and the total quantity sold for that product.

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:

The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases



Explores

monthForS...

sales_data...



Submit

Debugger

```
1 import pandas as pd
2
3 # Prompt the user for the file name
4 file_name = input()
5
6 # Load the data
7 df = pd.read_csv(file_name)
8 product_sales = df.groupby('Product')
9 ['Quantity'].sum()
10 # Find the product with the highest
11 total quantity sold
12 best_product = product_sales.idxmax()
13 highest_quantity =
14 product_sales.max()
15
16 # Display the result
17 print(f"Best selling product:
18 {best_product}")
19 print(f"Total quantity sold:
20 {highest_quantity}")
21
```



4.2.3. City that Sold the Most Products

00:57



Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the columns: Date, Product, Quantity, Price, and City.
- Group the data by City and calculate the total quantity of products sold for each city.
- Find the city that sold the most products (based on the total quantity sold).

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:

The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases



Explores



monthForS...



sales_data...



Submit

Debugger

```
1 import pandas as pd
2
3 # Prompt the user for the file name
4 file_name = input()
5
6 # Load the data
7 df = pd.read_csv(file_name)
8
9 # write the code..
10 city_sales = df.groupby('City')
11   ['Quantity'].sum()
12 best_city = city_sales.idxmax()
13
14 # Display the result
15 print(f"City sold the most products:
   {best_city}")
```


4.2.4. Most Frequently Sold Product Pairs

09:17



Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the following columns: Date, Product, Quantity, Price, and City.
- For each date, find all pairs of products that were sold together (i.e., two products sold on the same date).
- Output the product pair/s that was sold most frequently.

Sample Data:

Date	Product	Quantity	Price	City
2025-01-01	Product A	5	20	New York
2025-01-01	Product B	3	15	Los Angeles
2025-01-02	Product A	7	20	New York
2025-01-02	Product C	4	30	Chicago
2025-01-03	Product B	2	15	Chicago
2025-01-03	Product A	8	20	Los Angeles
2025-01-04	Product C	6	30	New York
2025-01-04	Product B	5	15	Los Angeles
2025-01-05	Product A	3	20	Chicago
2025-01-05	Product C	10	30	Los Angeles

Explanation:

Transactions:

- **2025-01-01:** Product A, Product B
- **2025-01-02:** Product A, Product C
- **2025-01-03:** Product B, Product A
- **2025-01-04:** Product C, Product B
- **2025-01-05:** Product A, Product C

Now, let's count how often the pairs of products appear together:

- **Product A and Product B:** Appear in transactions on 2025-01-01 and 2025-01-03.
- **Product A and Product C:** Appear in transactions on 2025-01-02 and 2025-01-05.
- **Product B and Product C:** Appears in transactions on 2025-01-04.

Most Frequent Product Combinations:

- **Product A and Product B** (2 times)
- **Product A and Product C** (2 times)

Note:

The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Explores

frequently...

sales_data...

Submit

Debugger

```
1 import pandas as pd
2 from itertools import combinations
3 from collections import Counter
4
5 # Prompt user to input the file name
6 file_name = input()
7
8 # Read data from the specified CSV
9 # file
10 df = pd.read_csv(file_name)
11
12 # write the code
13 date_products = {}
14
15 for date, group in
16     df.groupby('Date'):
17         products =
18             group['Product'].unique()
19         if len(products) > 1:
20             date_products[date] =
21                 products
22
23 pair_counter = Counter()
24
25 for products in
26     date_products.values():
27         pairs =
28             combinations(sorted(products), 2)
29         pair_counter.update(pairs)
30
31 if pair_counter:
32     max_count =
33         max(pair_counter.values())
34
35     for pair, count in
36         pair_counter.items():
37         if count == max_count:
38             print(f"{pair[0]} and
39                 {pair[1]}: {count} times")
40
41 else:
42     print("No product pairs found.")
43
44 # Output the most frequent product
45 # pairs
```

Sample Test Cases



You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset. For each question, perform necessary data cleaning, transformations, and calculations as required.

1. Display the first 5 rows of the dataset.
2. Display the last 5 rows of the dataset.
3. Get the shape of the dataset (number of rows and columns).
4. Get a summary of the dataset (using .info()).
5. Get basic statistics (mean, standard deviation, etc.) of the dataset using .describe().
6. Check for missing values and display the count of missing values for each column.
7. Fill missing values in the 'Age' column with the median age.
8. Fill missing values in the 'Embarked' column with the most frequent value (mode).
9. Drop the 'Cabin' column due to many missing values.
10. Create a new column, 'FamilySize' by adding the 'SibSp' and 'Parch' columns.

The Titanic dataset contains columns as shown below.

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ti
1	0	3	"Braund, Mr. Owen Harris"	male	22	1	0	A/5 21171.7
2	1	1	"Cumings, Mrs. John Bradley (Florence Briggs Thayer)"	female	38	1	0	33.93
3	1	3	"Heikkinen, Miss. Laina"	female	26	0	0	STON/O2. 33
4	1	1	"Futrelle, Mrs. Jacques Heath (Lily May Peel)"	female	35	1	0	1.53
5	0	3	"Allen, Mr. William Henry"	male	35	0	0	373450.8
6	0	3	"Moran, Mr. James"	male	0	0	0	330877.8
7	0	1	"McCarthy, Mr. Timothy J"	male	54	0	0	17463.51
8	0	3	"Palsson, Master. Gosta Leonard"	male	2	3	1	34990.0
9	1	3	"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)"	female	41	0	0	237733.9
10	1	2	"Nasser, Mrs. Nicholas (Adele Achem)"	female	14	1	0	531.0

Note: Refer to the visible test case for better reference.

titanicData...Submit

ExploresDebugger

```
1 import pandas as pd
2 import numpy as np
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-Dataset.csv')
6
7 # 1. Display the first 5 rows of the dataset
8
9 print(data.head())
10
11 # 2. Display the last 5 rows of the dataset
12
13 print(data.tail())
14
15 # 3. Get the shape of the dataset
16
17 print(data.shape)
18
19 # 4. Get a summary of the dataset (info)
20
21 print(data.info())
22
23 # 5. Get basic statistics of the dataset
24
25 print(data.describe())
26
27 # 6. Check for missing values
28
29 print(data.isnull().sum())
30
31 # 7. Fill missing values in the 'Age' column with the median age
32 median_age = data['Age'].median()
33 data['Age'].fillna(median_age, inplace=True)
34
35 # 8. Fill missing values in the 'Embarked' column with the mode
36 mode_embarked = data['Embarked'].mode()[0]
37 data['Embarked'].fillna(mode_embarked, inplace=True)
38
39 # 9. Drop the 'Cabin' column due to many missing values
40 data.drop('Cabin', axis=1, inplace=True)
41
42 # 10. Create a new column 'FamilySize' by adding 'SibSp' and 'Parch'
43 data['FamilySize'] = data['SibSp'] + data['Parch']
44
```


You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Create a new column 'IsAlone' which is 1 if the passenger is alone (FamilySize = 0), otherwise 0.
2. Convert the 'Sex' column to numeric values (male: 0, female: 1).
3. One-hot encode the 'Embarked' column, dropping the first category.
4. Get the mean age of passengers.
5. Get the median fare of passengers.
6. Get the number of passengers by class.
7. Get the number of passengers by gender.
8. Get the number of passengers by survival status.
9. Calculate the survival rate of passengers.
10. Calculate the survival rate by gender.

The Titanic dataset contains columns as shown below.

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

titanicData...Submit

ExploresDebugger

1import pandas as pd

2import numpy as np

3

4# Load the Titanic dataset

5data = pd.read_csv('Titanic-Dataset.csv')

6data['FamilySize'] = data['SibSp'] +

7data['Parch']

8

9data['Alone'] =

10np.where(data['FamilySize'] == 0, 1,

110)

12

13# 3. Convert 'Sex' to numeric (male:

140, female: 1)

15data['Sex'] =

16data['Sex'].map({'male': 0,

17'female': 1})

18

19# 4. One-hot encode the 'Embarked'

20column, dropping the first category

21data = pd.get_dummies(data, columns=

22['Embarked'], drop_first=True)

23

24# 6. Get the mean age of passengers

25mean_age = data['Age'].mean()

26print(mean_age)

27

28# 7. Get the median fare of

29passengers

30median_fare = data['Fare'].median()

31print(median_fare)

32

33# 8. Get the number of passengers by

34class

35passengers_by_class =

36data['Pclass'].value_counts()

37print(passengers_by_class)

38

39# 9. Get the number of passengers by

40gender

41passengers_by_gender =

42data['Sex'].value_counts().sort_index

43()

44print(passengers_by_gender)

45

46# 10. Get the number of passengers

47by survival status

48passengers_by_survival =

49data['Survived'].value_counts().sort_

50index()

51print(passengers_by_survival)

52

53# 11. Calculate the survival rate

54survival_rate =

55data['Survived'].mean()

56print(survival_rate)

57

58# 12. Calculate the survival rate by

59gender

60survival_rate_by_gender =

61data.groupby('Sex')

62['Survived'].mean()

63print(survival_rate_by_gender)

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Calculate the survival rate by class.
2. Calculate the survival rate by embarkation location (Embarked_S).
3. Calculate the survival rate by family size (FamilySize).
4. Calculate the survival rate by being alone (IsAlone).
5. Get the average fare by passenger class (Pclass).
6. Get the average age by passenger class (Pclass).
7. Get the average age by survival status (Survived).
8. Get the average fare by survival status (Survived).
9. Get the number of survivors by class (Pclass).
10. Get the number of non-survivors by class (Pclass).

The Titanic dataset contains columns as shown below,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ti
1	0	3	"Braund, Mr. Owen Harris"	male	22	1	0	A/S 21171,7
2	1	1	"Cumings, Mrs. John Bradley (Florence Briggs Thayer)"	female	38	1	3	53.1
3	1	3	"Heikkinen, Miss. Laina"	female	26	0	0	STON/O2. 3101282
4	1	1	"Futrelle, Mrs. Jacques Heath (Lily May Peel)"	female	35	0	0	51.0
5	0	3	"Allen, Mr. William Henry"	male	35	0	0	373450,8.0
6	0	3	"Moran, Mr. James"	male	0	0	0	330877,8.4583,Q
7	0	1	"McCarthy, Mr. Timothy J"	male	54	0	0	17463,51.86
8	0	3	"Palsson, Master. Gosta Leonard"	male	2	3	1	34990
9	1	3	"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)"	female	41	0	0	9.0
10	1	2	"Nasser, Mrs. Nicholas (Adele Achem)"	female	14	1	0	

Note: Refer to the visible test case for better reference.

titanicData...

Submit

Explores

Debugger

```
1 import pandas as pd
2 import numpy as np
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-Dataset.csv')
6 data['FamilySize'] = data['SibSp'] +
7 data['Parch']
8 data['IsAlone'] =
9 np.where(data['FamilySize'] > 0, 0,
10 1)
11 data = pd.get_dummies(data, columns=
12 ['Embarked'], drop_first=True)
13
14 print(data.groupby('Pclass').
15 ['Survived'].mean())
16
17 #2. Calculate the survival rate by
18 embarked location (Embarked_S)
19 print(data.groupby('Embarked_S').
20 ['Survived'].mean())
21
22 #3. Calculate the survival rate by
23 family size
24 print(data.groupby('FamilySize').
25 ['Survived'].mean())
26
27 #4. Calculate the survival rate by
28 being alone
29 print(data.groupby('IsAlone').
30 ['Survived'].mean())
31
32 #5. Get the average fare by class
33 print(data.groupby('Pclass').
34 ['Fare'].mean())
35
36 #6. Get the average age by class
37 print(data.groupby('Pclass').
38 ['Age'].mean())
39
40 #7. Get the average age by survival
41 status
42 print(data.groupby('Survived')
43 ['Age'].mean())
44
45 #8. Get the average fare by survival
46 status
47 print(data.groupby('Survived').
48 ['Fare'].mean())
49
50 #9. Get the number of survivors by
51 class (sort by values descending)
52 print(data[data['Survived'] == 1].
53 ['Pclass'].value_counts())
54
55 #10. Get the number of non-survivors
56 by class (sort by values descending)
57 print(data[data['Survived'] == 0].
58 ['Pclass'].value_counts())
59
60
```


You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Get the number of survivors by gender (Sex).
2. Get the number of non-survivors by gender (Sex).
3. Get the number of survivors by embarkation location (Embarked_S).
4. Get the number of non-survivors by embarkation location (Embarked_S).
5. Calculate the percentage of children (Age < 18) who survived.
6. Calculate the percentage of adults (Age >= 18) who survived.
7. Get the median age of survivors.
8. Get the median age of non-survivors.
9. Get the median fare of survivors.
10. Get the median fare of non-survivors.

The Titanic dataset contains columns as shown below;

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

titanicData... Submit

Debugger

```
1 import pandas as pd
2 import numpy as np
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-
Dataset.csv')
6 data = pd.get_dummies(data, columns=
['Embarked'], drop_first=True)
7
8
9 survivors_by_gender =
data[data['Survived'] == 1]
['Sex'].value_counts()
10 print(survivors_by_gender)
11
12 # 2. Get the number of non-survivors-
by-gender
13 non_survivors_by_gender =
data[data['Survived'] == 0]
['Sex'].value_counts()
14 print(non_survivors_by_gender)
15
16 # 3. Get the number of survivors by
embarked location (Embarked_S)
17 survivors_by_embarked_s =
data[data['Survived'] == 1]
['Embarked_S'].value_counts()
18 print(survivors_by_embarked_s)
19
20 # 4. Get the number of non-survivors
by embarked location (Embarked_S)
21 non_survivors_by_embarked_s =
data[data['Survived'] == 0]
['Embarked_S'].value_counts()
22 print(non_survivors_by_embarked_s)
23
24 #5. Percentage of children (Age <
18) who survived
25 children=data [data['Age'] < 18]
26 children_survival_rate=
children['Survived'].mean()
27 print(children_survival_rate)
28
29 #6. Percentage of adults (Age->--18)
who survived
30 adults= data[data['Age'] >=18]
31 adults_survival_rate=
adults['Survived'].mean()
32 print(adults_survival_rate)
33
34 #7. Median age of survivors
35 median_age_survivors =
data[data['Survived'] == 1]
['Age'].median()
36 print(median_age_survivors)
37
38 #8. Median age of non-survivors
39 median_age_non_survivors =
data[data['Survived'] == 0]
['Age'].median()
40 print(median_age_non_survivors)
41
42 #9. Median fare of survivors
43 median_fare_survivors =
data[data['Survived'] == 1]
['Fare'].median()
44 print(median_fare_survivors)
45
46 # 10. Median fare of non-survivors
47 median_fare_non_survivors =
data[data['Survived'] == 0]
['Fare'].median()
48 print(median_fare_non_survivors)
49
50
```

Sample Test Cases

