4.1.1. Pandas - series creation and manipulation

47:04

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 +

Explorer

seriesMa...

Submit

```
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  # Create a Pandas series from the list of numbers
7  data_list=numbers
8  series_from_list=pd.Series(data_list)
9  # Grouping by even and odd numbers and calculating the mean
10 grouped =series_from_list.groupby(series_from_list % 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
```

Debugger

Terminal Test cases

4.1.2. Dictionary to dataframe

48:30

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:

Sample Test Cases +

Explorer

datafram...

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 new_row={'Name': new_name,'Age':new_age}
20 df=pd.concat([df,pd.DataFrame([new_row])],ignore_index=True)
21 # Display the DataFrame after adding a new row
22 print("After adding a row:\n",df)
23
24 # Modifying a row
25 modify_index=int(input("Index of row to modify: "))
```

Terminal Test cases

4.1.2. Dictionary to dataframe

48:30

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:

Sample Test Cases +

Explorer

dataframe...

Submit

Debugger

```
# Deleting a row
delete_index=int(input("Index of row to delete: "))
df=df.drop(delete_index).reset_index(drop=True)
# Display the DataFrame after deleting a row
print("After deleting a row:")
print(df)

# Adding a new column
gender_input=input("Enter genders separated by space: ")
genders=gender_input.split()
df["Gender"]=genders
# Display the DataFrame after adding a new column
print("After adding a new column:")
print(df)

# Modifying a column
df["Name"]=df["Name"].str.upper()
# Display the DataFrame after modifying a column
print("After modifying a column:")
print(df)

# Deleting a column
df=df.drop(columns=['Age'])
# Display the DataFrame after deleting a column
print("After deleting a column:")
print(df)
```

Terminal Test cases

4.1.3. Student Information

05:05

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

studentin... studentdat...

Submit

```python
import pandas as pd

# Read the text file into a DataFrame
file = input()
data = pd.read_csv(file, sep="\s+", header=None, names=["Name", "Age", "Grade"])
print("First five rows:")
print(data.head(5))
# write your code here..
age=round(data['Age'].mean(),2)
print("Average age:",age)
print("Students with a grade up to B")
df=pd.DataFrame(data)
a=df[df['Grade']<='B']
print(a)
```

Debugger

Terminal Test cases

4.2.1. Month with the Highest Total Sales

04:13

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:

Sample Test Cases +

Explorer

monthFor... sales_dat...

Submit

Debugger

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)
df['sales']=df['Quantity'].multiply(df['Price'])
df['month']=pd.to_datetime(df['Date']).dt.strftime("%Y-%m")
# Find the month with the highest total sales
best_month =df.groupby('month')['sales'].sum().idxmax()
highest_sales = df['sales'].sum()

print(f"Best month: {best_month}")
print(f"Total sales: ${highest_sales:.2f}")

```

Terminal Test cases

4.2.2. Best Selling Product

25:34

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 +

Explorer

monthFor... sales_dat...

Submit

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)


# Find the product with the highest total quantity sold
product_sales = df.groupby("Product")["Quantity"].sum()
best_product = product_sales.idxmax()
highest_quantity = product_sales.max()

# Display the result
print(f"Best selling product: {best_product}")
print(f"Total quantity sold: {highest_quantity}")

```

Terminal Test cases

Debugger

< Prev Reset Submit Next >

4.2.3. City that Sold the Most Products

02:00

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 +

Explorer

monthFor... sales_dat...

Submit

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)

# write the code..
city_sales = df.groupby("City")["Quantity"].sum()


best_city = city_sales.idxmax()
# Display the result
print(f"City sold the most products: {best_city}")

```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

4.2.4. Most Frequently Sold Product Pairs

24:43

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:

Sample Test Cases +

Explorer

frequentl... sales_dat...

Submit

```python
import pandas as pd
from itertools import combinations
from collections import Counter

# Prompt user to input the file name
file_name = input()

# Read data from the specified CSV file
df = pd.read_csv(file_name)

# write the code

date_products = {}

for date, group in df.groupby('Date'):
    products = group['Product'].unique()
    if len(products) > 1:
        date_products[date] = products

pair_counter = Counter()

for products in date_products.values():
    pairs = combinations(sorted(products), 2)
    pair_counter.update(pairs)

```

Terminal Test cases

Debugger

Prev Reset Submit Next

4.2.4. Most Frequently Sold Product Pairs

24:43

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:

Sample Test Cases +

Explorer

frequentI... sales_dat...

Submit

Debugger

```
10
11  # write the code
12
13  date_products = {}
14
15  for date, group in df.groupby('Date'):
16      products = group['Product'].unique()
17      if len(products) > 1:
18          date_products[date] = products
19
20  pair_counter = Counter()
21
22  for products in date_products.values():
23      pairs = combinations(sorted(products), 2)
24      pair_counter.update(pairs)
25
26  if pair_counter:
27      max_count = max(pair_counter.values())
28
29      for pair, count in pair_counter.items():
30          if count == max_count:
31              print(f"{pair[0]} and {pair[1]}: {count} times")
32  else:
33      print("No product pairs found.")
34  # Output the most frequent product pairs
```

Terminal Test cases

4.2.5. Titanic Dataset Analysis and Data Cleaning

17:59

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,

Passengerld	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')

print(data.head())

# 2. Display the last 5 rows of the dataset
print(data.tail())

# 3. Get the shape of the dataset
print(data.shape)

# 4. Get a summary of the dataset (info)
print(data.info())

# 5. Get basic statistics of the dataset
print(data.describe())

# 6. Check for missing values
print(data.isnull().sum())

# 7. Fill missing values in the 'Age' column with the median age
median_age = data['Age'].median()
```

Terminal Test cases

‹ Prev Reset Submit Next ›

4.2.5. Titanic Dataset Analysis and Data Cleaning

17:59

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,

Passengerld	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```
# 4. Get a summary of the dataset (info)
print(data.info())

# 5. Get basic statistics of the dataset
print(data.describe())

# 6. Check for missing values
print(data.isnull().sum())

# 7. Fill missing values in the 'Age' column with the median age
median_age = data['Age'].median()
data['Age'].fillna(median_age, inplace=True)

# 8. Fill missing values in the 'Embarked' column with the mode
mode_embarked = data['Embarked'].mode()[0]
data['Embarked'].fillna(mode_embarked, inplace=True)

# 9. Drop the 'Cabin' column due to many missing values
data.drop('Cabin', axis=1, inplace=True)

# 10. Create a new column 'FamilySize' by adding 'SibSp' and 'Parch'
data['FamilySize'] = data['SibSp'] + data['Parch']
```

Terminal Test cases

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

18:54

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,

Passengerld	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data['FamilySize'] = data['SibSp'] + data['Parch']

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

# 2. Convert 'Sex' to numeric (male: 0, female: 1)
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# 3. One-hot encode the 'Embarked' column
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# 4. Get the mean age of passengers
mean_age = data['Age'].mean()
print(mean_age)

# 5. Get the median fare of passengers
median_fare = data['Fare'].median()
print(median_fare)

# 6. Get the number of passengers by class
passengers_by_class = data['Pclass'].value_counts()
```

Terminal Test cases

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

18:54

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,

Passengerld	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```
19
20  # 5. Get the median fare of passengers
21  median_fare = data['Fare'].median()
22  print(median_fare)
23
24  # 6. Get the number of passengers by class
25  passengers_by_class = data['Pclass'].value_counts()
26  print(passengers_by_class)
27
28  # 7. Get the number of passengers by gender
29  passengers_by_gender = data['Sex'].value_counts().sort_index()
30  print(passengers_by_gender)
31
32  # 8. Get the number of passengers by survival status
33  passengers_by_survival = data['Survived'].value_counts().sort_index()
34  print(passengers_by_survival)
35
36  # 9. Calculate the survival rate
37  survival_rate = data['Survived'].mean()
38  print(survival_rate)
39
40  # 10. Calculate the survival rate by gender
41  survival_rate_by_gender = data.groupby('Sex')['Survived'].mean()
42  print(survival_rate_by_gender)
43
```

Terminal Test cases

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3

08:18

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,

Passengerid	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases +

Explorer titanicDat... Submit Debugger

```
13  print(data.groupby('Embarked_S')['Survived'].mean())
14
15  # 3. Calculate the survival rate by family size
16  print(data.groupby('FamilySize')['Survived'].mean())
17
18  # 4. Calculate the survival rate by being alone
19  print(data.groupby('IsAlone')['Survived'].mean())
20
21  # 5. Get the average fare by class
22  print(data.groupby('Pclass')['Fare'].mean())
23
24  # 6. Get the average age by class
25  print(data.groupby('Pclass')['Age'].mean())
26
27  # 7. Get the average age by survival status
28  print(data.groupby('Survived')['Age'].mean())
29
30  # 8. Get the average fare by survival status
31  print(data.groupby('Survived')['Fare'].mean())
32
33  # 9. Get the number of survivors by class
34  print(data[data['Survived']==1]['Pclass'].value_counts())
35
36  # 10. Get the number of non-survivors by class
37  print(data[data['Survived'] == 0]['Pclass'].value_counts())
```

Terminal Test cases

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

30 28

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,

Passengerld	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)


survivors_by_gender = data[data['Survived'] == 1]['Sex'].value_counts()
print(survivors_by_gender)

# 2. Get the number of non-survivors by gender
non_survivors_by_gender = data[data['Survived'] == 0]
['Sex'].value_counts()
print(non_survivors_by_gender)

# 3. Get the number of survivors by embarked location
survivors_by_embarked_s = data[data['Survived'] == 1]
['Embarked_S'].value_counts()
print(survivors_by_embarked_s)

# 4. Get the number of non-survivors by embarked location
non_survivors_by_embarked_s = data[data['Survived'] == 0]
['Embarked_S'].value_counts()
print(non_survivors_by_embarked_s)
```

Terminal Test cases

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

30:28

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,

Passengerld	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```python
# 5. Calculate the percentage of children (Age < 18) who survived
children = data[data['Age'] < 18]
children_survival_rate = children['Survived'].mean()
print(children_survival_rate)

# 6. Calculate the percentage of adults (Age >= 18) who survived
adults = data[data['Age'] >= 18]
adults_survival_rate = adults['Survived'].mean()
print(adults_survival_rate)

# 7. Get the median age of survivors
median_age_survivors = data[data['Survived'] == 1]['Age'].median()
print(median_age_survivors)

# 8. Get the median age of non-survivors
median_age_non_survivors = data[data['Survived'] == 0]['Age'].median()
print(median_age_non_survivors)

# 9. Get the median fare of survivors
median_fare_survivors = data[data['Survived'] == 1]['Fare'].median()
print(median_fare_survivors)

# 10. Get the median fare of non-survivors
median_fare_non_survivors = data[data['Survived'] == 0]['Fare'].median()
print(median_fare_non_survivors)
```

Terminal Test cases