4.1.1. Pandas - series creation and manipulation

16:18

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=pd.Series(data_list)
9
10 # Grouping by even and odd numbers and calculating the mean
11 grouped = series.groupby(series % 2 == 0).mean()
12
13 # Display the mean of even and odd numbers with labels
14 grouped.index = ['Even' if is_even else 'Odd' for is_even in
   grouped.index]
15 print("Mean of even and odd numbers:")
16 print(grouped)
17
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

4.1.2. Dictionary to dataframe

42:27

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:

Sample Test Cases +

Explorer

dataframe...

Submit

Debugger

```python
import pandas as pd

# Provided dictionary of lists
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
}

# Convert the dictionary to a DataFrame
df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Adding a new row
new_name =input("New name: ")
new_age=int(input("New age: "))
new_row={'Name': new_name,'Age':new_age}
df=pd.concat([df,pd.DataFrame([new_row])],ignore_index=True)
# Display the DataFrame after adding a new row
print("After adding a row:\n",df)

# Modifying a row
modify_index=int(input("Index of row to modify: "))
new_age_mod=int(input("New age: "))
df.loc[modify_index,"Age"] = new_age_mod
# Display the DataFrame after modifying a row
print("After modifying a row:")
```

Terminal Test cases

4.1.3. Student Information

00:46

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

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+", header=None, names=["Name", "Age",
   "Grade"])
6  print("First five rows:")
7  print(data.head(5))
8
9  # write your code here..
10 age=round(data['Age'].mean(),2)
11 print("Average age:",age)
12 print("Students with a grade up to B")
13 df=pd.DataFrame(data)
14 a=df[df['Grade']<='B']
15 print(a)
16
```

Terminal Test cases

4.2.1. Month with the Highest Total Sales

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 +

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)
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

03:02

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

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)


# 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

4.2.3. City that Sold the Most Products

02:20

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}")

```

Terminal Test cases

Debugger

4.2.4. Most Frequently Sold Product Pairs

41:17

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

Sample Test Cases +

Explorer

frequentI... 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)


date_groups = df.groupby('Date')['Product'].apply(list)
pair_counter = Counter()

for products in date_groups:
    unique_pairs = combinations(sorted(set(products)), 2)
    pair_counter.update(unique_pairs)

max_freq = max(pair_counter.values())

for pair, count in pair_counter.items():
    if count == max_freq:
        print(f"{pair[0]} and {pair[1]}: {count} times")
```

Terminal Test cases

Debugger

4.2.5. Titanic Dataset Analysis and Data Cleaning

12:31

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')

# 1. Display the first 5 rows of the dataset
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()
data['Age'].fillna(median_age, inplace=True)
# 8. Fill missing values in the 'Embarked' column with the mode
mode_embarked = data['Embarked'].mode()[0]
```

Terminal Test cases

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

16:31

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,

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

Sample Data:

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']

# 1. Create a new column 'IsAlone' (1 if alone, 0 otherwise)
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# 2. Convert 'Sex' to numeric (male: 0, female: 1)
data = pd.get_dummies(data, columns= ['Embarked'],drop_first=True)
mean_age = data['Age'].mean()
print(mean_age)
# 3. One-hot encode the 'Embarked' column
median_fare = data['Fare'].median()
print(median_fare)
# 4. Get the mean age of passengers
passengers_by_class = data['Pclass'].value_counts()
print(passengers_by_class)


# 5. Get the median fare of passengers



# 6. Get the number of passengers by class

```

Terminal Test cases

CODETANTRA Home 202401070223@mitaoe.ac.in Support Logout

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3

15:38

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 Data:

Sample Test Cases +

Explorer titanicDat... 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['FamilySize'] = data['SibSp'] + data['Parch']
7  data['IsAlone'] = np.where(data['FamilySize'] > 0, 0, 1)
8  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
9
10 print(data.groupby('Pclass') ['Survived'].mean())
11
12 #2. Calculate the survival rate by embarked location (Embarked_S)
13
14 print(data.groupby('Embarked_S') ['Survived'].mean())
15
16 #3. Calculate the survival rate by family size
17
18 print(data.groupby('FamilySize') ['Survived'].mean())
19
20 #4. Calculate the survival rate by being alone
21
22 print(data.groupby('IsAlone') ['Survived'].mean())
23
24 #5. Get the average fare by class
25
26 print(data.groupby('Pclass') ['Fare'].mean())
27
28 #6. Get the average age by class
29
```

Terminal Test cases