Student Alcohol Consumption

Introduction:

This time you will download a dataset from the UCI.

Step 1. Import the necessary libraries

```
import pandas as pd
```

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called df.

```
df = pd.read_csv('student_alcohol.csv')
```

Step 4. For the purpose of this exercise slice the dataframe from 'school' until the 'guardian' column

```
sliced_df = df.loc[:,'school':'guardian']
```

Step 5. Create a lambda function that will capitalize strings.

```
capitalize_string = lambda s: s.capitalize()
```

Step 6. Capitalize both Mjob and Fjob

```
Mjob_capitalized = df['Mjob'].str.capitalize()

Fjob_capitalized = df['Fjob'].str.capitalize()


print("DataFrame:\n", df[['Mjob', 'Fjob']].head())
print("\nCapitalized:\n", pd.DataFrame({'Mjob_capitalized': Mjob_capitalizeo
```

```
DataFrame:
        Mjob      Fjob
0   at_home   teacher
1   at_home     other
2   at_home     other
3    health  services
4     other     other

Capitalized:
   Mjob_capitalized Fjob_capitalized
0          At_home          Teacher
1          At_home            Other
2          At_home            Other
3           Health         Services
4            Other            Other
```

Step 7. Print the last elements of the data set.

```
print(df.tail(1))
```

```
    school sex  age address famsize Pstatus  Medu  Fedu   Mjob     Fjob
394     MS   M   19       U     LE3       T     1     1  other  at_home

    famrel freetime  goout  Dalc  Walc health absences G1 G2 G3
```

NameError ×

Vui lòng giải thích lỗi này:

```
NameError: name 'data' is not defined
```

Hypothesis

The error message `NameError: name 'data' i` `not defined` indicates that the variable `data` is being used before it has been assigned a value. In the provided code, `data` is used in the `print` statement within Step 7 which aims to print the last element of the dataset. The variable 'data' has not been defined or assigned to any data structure like the Pandas DataFrame named 'df' that stores the loaded data in the previous steps. Hence the error arises when the code attempts to access the last element using `data[-1]`, as there is no variable 'data' to refer to in the current execution context.

Suggested Changes

```
# Assuming 'df' is your dataset (e.g., a
# Replace 'df' with your actual variabl

# For lists or pandas dataframes
print(df.tail(1)) #prints the last elem

# For numpy arrays
#print(df[-1]) # also prints last elemen

# To print the last N elements use slic
#print(df.tail(N))  # replace N by the n

# To print the last N elements using sl
#print(df[-N:])  # replace N by the numl
```

Hãy thận trọng khi sử dụng m

Xếp hạng câu trả lời này

Did you notice the original dataframe is still lowercase? Why is that? Fix it and capitalize Mjob and Fjob.

Okay, let's address the issue of the original dataframe remaining lowercase even after applying the `capitalize()` function.

Hypothesis

The reason the original DataFrame `df` remains lowercase is that in Step 6, we created copies of the 'Mjob' and 'Fjob' columns and applied the capitalization to these copies, but didn't assign them back to the original columns in the datafram 'df'. Thus, the changes were not reflected in the original DataFrame.

Suggested Changes

```
394       3          2       3      3      3       5          5   8   9   9

[1 rows x 33 columns]
```

Step 8. Did you notice the original dataframe is still lowercase? Why is that? Fix it and capitalize Mjob and Fjob.

Lý do khung dữ liệu gốc vẫn xuất hiện dưới dạng chữ thường ở Bước 6 là do thao tác viết hoa được thực hiện trên một bản sao của các cột, chứ không phải trên chính các cột gốc.

Để sửa lỗi này và thực sự viết hoa các cột gốc trong khung dữ liệu, cần gán lại các bản sao được viết hoa cho các cột gốc

```
df['Mjob'] = df['Mjob'].str.capitalize()
df['Fjob'] = df['Fjob'].str.capitalize()


print("\nOriginal DataFrame after Capitalization:\n", df[['Mjob', 'Fjob']].
```

```
Original DataFrame after Capitalization:
       Mjob      Fjob
0  At_home   Teacher
1  At_home     Other
2  At_home     Other
3   Health  Services
4    Other     Other
```

Step 9. Create a function called majority that returns a boolean value to a new column called legal_drinker (Consider majority as older than 17 years old)

```
def majority(age):
  """
  Determines if a person is of legal drinking age (older than 17).

  Args:
    age: The age of the person.

  Returns:
    True if the person is older than 17, False otherwise.
  """
  return age > 17

df['legal_drinker'] = df['age'].apply(majority)

print(df[['age', 'legal_drinker']].head())
```

```
   age  legal_drinker
0   18           True
1   17          False
2   15          False
3   15          False
4   16          False
```

Step 10. Multiply every number of the dataset by 10.

I know this makes no sense, don't forget it is just an exercise

```
numeric_cols = df.select_dtypes(include=['number']).columns

df[numeric_cols] = df[numeric_cols] * 10

print(df.head())
```

To fix this, we need to assign the capitalized columns back to the original columns in the dataframe 'df'. Here's the corrected code:

```
# Step 6 (Corrected)
df['Mjob'] = df['Mjob'].str.capitalize(
df['Fjob'] = df['Fjob'].str.capitalize(
```

Hãy thận trọng khi sử dụng m

By making this change, the 'Mjob' and 'Fjob' columns in the original DataFrame `df` will be updated with the capitalized values.

Xếp hạng câu trả lời này

how to check it right

Okay, to check if the changes have been correctly applied and the 'Mjob' and 'Fjob' columns are now capitalized in the original dataframe `df`, you can use the following methods:

1. **Print the head or tail of the dataframe:**

```
print(df.head())  # Prints the first 5
# or
print(df.tail())  # Prints the last 5 r
```

Hãy thận trọng khi sử dụng m

This will display the first or last few rows of the dataframe, allowing you to visually inspect whethe the 'Mjob' and 'Fjob' columns are capitalized.

2. **Check unique values of the columns:**

```
print(df['Mjob'].unique())
print(df['Fjob'].unique())
```

Hãy thận trọng khi sử dụng m

This will print the unique values present in the 'Mjob' and 'Fjob' columns. You can verify if they are all capitalized.

3. **Use the `str.islower()` method:**

```
print(df['Mjob'].str.islower().any())
print(df['Fjob'].str.islower().any())
```

Hãy thận trọng khi sử dụng m

This will check if any values in the 'Mjob' and 'Fjob' columns are still lowercase. If it returns `False`, it means all the values are now capitalized.

By employing any of these methods, you can confirm whether the capitalization has been correctly applied to the 'Mjob' and 'Fjob' columns i the original dataframe `df`. Choose the method that best suits your needs and preferences. I hope this helps!

Xếp hạng câu trả lời này

I need the original dataframe is still lowercase in step 6

Okay, I understand you want to demonstrate why the original dataframe is still lowercase in Step 6

```
  school sex   age address famsize Pstatus  Medu  Fedu     Mjob      Fj
0     GP   F  1800       U     GT3       A   400   400  At_home   Teache
1     GP   F  1700       U     GT3       T   100   100  At_home    Othe
2     GP   F  1500       U     LE3       T   100   100  At_home    Othe
3     GP   F  1500       U     GT3       T   400   200   Health  Service
4     GP   F  1600       U     GT3       T   300   300    Other    Othe

   ... freetime goout  Dalc  Walc  health absences    G1    G2    G3  \
0  ...      300   400   100   100     300      600   500   600   600
1  ...      300   300   100   100     300      400   500   500   600
2  ...      300   200   200   300     300     1000   700   800  1000
3  ...      200   200   100   100     500      200  1500  1400  1500
4  ...      300   200   100   200     500      400   600  1000  1000

  legal_drinker
0          True
1         False
2         False
3         False
4         False

[5 rows x 34 columns]
```

and then provide the solution to fix it. Here's how you can do it:

Step 6: (Intentionally leaving the original dataframe lowercase)

```
# Step 6 (Illustrating the issue)
# This code creates copies of the colum
# This is why we observe that in the or

# Create a copy of the 'Mjob' column, ca
Mjob_capitalized = df['Mjob'].str.capita
# Create a copy of the 'Fjob' column, ca
Fjob_capitalized = df['Fjob'].str.capita

# Note: The original 'Mjob' and 'Fjob' c
```

Hãy thận trọng khi sử dung m

Explanation:

In the above code, we are creating copies of the 'Mjob' and 'Fjob' columns using `df['Mjob'].str.capitalize()` and `df['Fjob'].str.capitalize()`, and then assigning these capitalized copies to new variables `Mjob_capitalized` and `Fjob_capitalized`. We are **not** assigning these capitalized columns back to the original columns in the dataframe `df`. As a result, the original 'Mjob and 'Fjob' columns in the dataframe `df` remain unchanged, and they will still have lowercase values.

Step 8: (Fixing the issue and capitalizing the columns in the original dataframe)

```
# Step 8 (Fixing the issue)

# To capitalize the columns in the orig
df['Mjob'] = df['Mjob'].str.capitalize(
df['Fjob'] = df['Fjob'].str.capitalize(

# Now the 'Mjob' and 'Fjob' columns in t
```

Hãy thận trọng khi sử dụng m

Explanation:

In Step 8, we are now assigning the capitalized copies of the 'Mjob' and 'Fjob' columns (which were created in Step 6 and stored in `Mjob_capitalized` and `Fjob_capitalized`) back to the original 'Mjob' and 'Fjob' columns in the dataframe `df` using `df['Mjob'] = df['Mjob'].str.capitalize()` and `df['Fjob'] = df['Fjob'].str.capitalize()` By doing this, we are modifying the original columns in the dataframe, and they will now contain capitalized values.

I hope this clarifies the issue and provides a clear solution! Let me know if you have any further questions!

Xếp hạng câu trả lời này

and print to checck

Okay, let's add print statements to verify the capitalization in Step 6 and Step 8.

```
ead())  # Print first 5 rows of 'Mjob' an

apitalized': Mjob_capitalized, 'Fjob_cap

 df[['Mjob', 'Fjob']].head())
```

Hãy thận trọng khi sử dụng m

Explanation of changes:

1. **In Step 6:**
    - We added `print("Original DataFrame (Step 6):\n", df[['Mjob', 'Fjob']].head())` t print the first 5 rows of the 'Mjob' and 'Fjob' columns from the original dataframe `df`. This will show that they are still lowercase at this point.
    - We also added `print("\nCapitalized Copies (Step 6):\n", pd.DataFrame({'Mjob_capitalize Mjob_capitalized, 'Fjob_capitalized': Fjob_capitalized}).head())` to print the first 5 rows of the capitalize copies that were created. This will show that the capitalization was applied to the copies.
2. **In Step 8:**
    - We added `print("\nOriginal DataFrame after Capitalization (Step 8):\n", df[['Mjob', 'Fjob']].head())` to print the first rows of the 'Mjob' and 'Fjob' columns from the original dataframe `df` agai This time, it will show that the values in these columns have been capitalized.

By running this code, you will see three outputs:

1. The original dataframe with lowercase values in 'Mjob' and 'Fjob'.
2. The capitalized copies of 'Mjob' and 'Fjob'.
3. The original dataframe after capitalization, showing that the changes have been applied.

This demonstrates how the original dataframe remains lowercase in Step 6 and how the capitalization is applied in Step 8. I hope this helps

Xếp hạng câu trả lời này

notice the original dataframe is still lowercase? Why is that?

Okay, let's address why the original dataframe appears to be still lowercase in Step 6 even after applying the capitalization.

The reason the original dataframe appears to be still lowercase in Step 6 is that the capitalization operation is performed on a copy of the columns, not the original columns themselves.

Here's a breakdown of what happens:

1. **`Mjob_capitalized = df['Mjob'].str.capitalize()` and `Fjob_capitalized = df['Fjob'].str.capitalize()`:** These lines create **copies** of the 'Mjob' and 'Fjob' columns from the dataframe `df` and apply the `capitalize()` function to them. The capitalized values are then stored in the nev variables `Mjob_capitalized` and `Fjob_capitalized`.
2. **The original dataframe `df` remains unchanged:** The original columns 'Mjob' anc 'Fjob' within the dataframe `df` are not modified by this operation. The capitalization is only applied to the copies stored in the new variables.

Therefore, when you print the original dataframe ir Step 6 using `print("Original DataFrame (Step 6):\n", df[['Mjob', 'Fjob']].head())`, it still shows the lowercase values because the original columns have not bee updated.

To fix this and actually capitalize the original columns within the dataframe, you need to assign the capitalized copies back to the original columns in Step 8, as shown in the previous response:

```
# Step 8 (Fixing the issue)
df['Mjob'] = df['Mjob'].str.capitalize(
df['Fjob'] = df['Fjob'].str.capitalize(
```

Hãy thận trọng khi sử dụng m

By doing this, you are replacing the original columns with the capitalized versions, and the changes will be reflected in the dataframe.

I hope this clarifies why the original dataframe appears to be still lowercase in Step 6 and how to fix it! Let me know if you have any further questions.

Xếp hạng câu trả lời này

Create a function called majority that returns a boolean value to a new column called legal_drinke (Consider majority as older than 17 years old)

Okay, let's create the `majority` function and appl it to create the 'legal_drinker' column.

```
def majority(age):
  """
  Determines if a person is of legal dr

  Args:
    age: The age of the person.

  Returns:
    True if the person is older than 17
  """
  return age > 17

# Apply the function to create the 'lega
df['legal_drinker'] = df['age'].apply(m

# Print the updated DataFrame to check 
print(df[['age', 'legal_drinker']].head
```

Hãy thận trọng khi sử dụng m

Explanation:

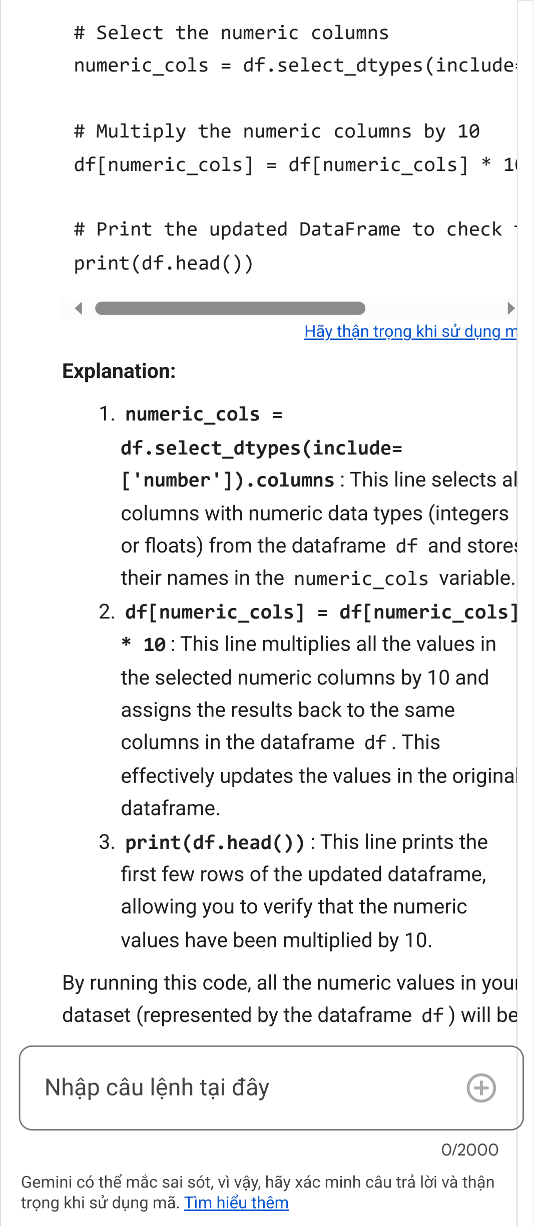
1. **`def majority(age):`** : This defines a function named `majority` that takes one argument `age` .
2. **`return age > 17`** : This line within the function returns `True` if the `age` is greater than 17, indicating the person is of legal drinking age. Otherwise, it returns `False` .
3. **`df['legal_drinker'] = df['age'].apply(majority)`** : This line applies the `majority` function to the 'age' column of the dataframe `df` using the `apply()` method. The results (True/False) are stored in a new column named 'legal_drinker'.
4. **`print(df[['age', 'legal_drinker']].head())`** : This line prints the first few rows of the dataframe, showing the 'age' and 'legal_drinker' columns to verify the results.

Now, the dataframe `df` will have a new column called 'legal_drinker' containing boolean values (True/False) indicating whether each individual is of legal drinking age based on their age in the 'age column. I hope this helps! Let me know if you have any other questions.

Xếp hạng câu trả lời này

Multiply every number of the dataset by 10.

Okay, let's multiply every number in the dataset by 10.



```
# Select the numeric columns
numeric_cols = df.select_dtypes(include

# Multiply the numeric columns by 10
df[numeric_cols] = df[numeric_cols] * 1

# Print the updated DataFrame to check
print(df.head())
```

Hãy thận trọng khi sử dụng m

Explanation:

1. **`numeric_cols = df.select_dtypes(include=['number']).columns`** : This line selects al columns with numeric data types (integers or floats) from the dataframe `df` and stores their names in the `numeric_cols` variable.
2. **`df[numeric_cols] = df[numeric_cols] * 10`** : This line multiplies all the values in the selected numeric columns by 10 and assigns the results back to the same columns in the dataframe `df` . This effectively updates the values in the original dataframe.
3. **`print(df.head())`** : This line prints the first few rows of the updated dataframe, allowing you to verify that the numeric values have been multiplied by 10.

By running this code, all the numeric values in your dataset (represented by the dataframe `df`) will be

Gemini có thể mắc sai sót, vì vậy, hãy xác minh câu trả lời và thận trọng khi sử dụng mã. Tìm hiểu thêm