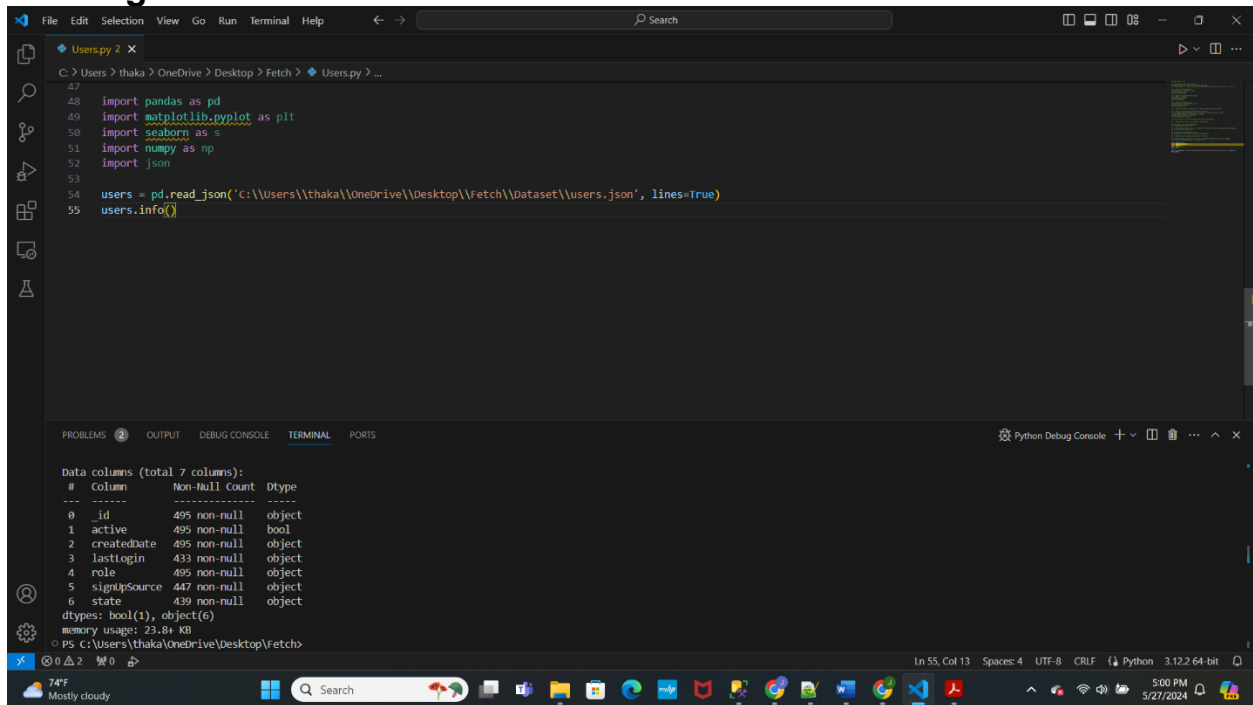


# Loading dataset for Users



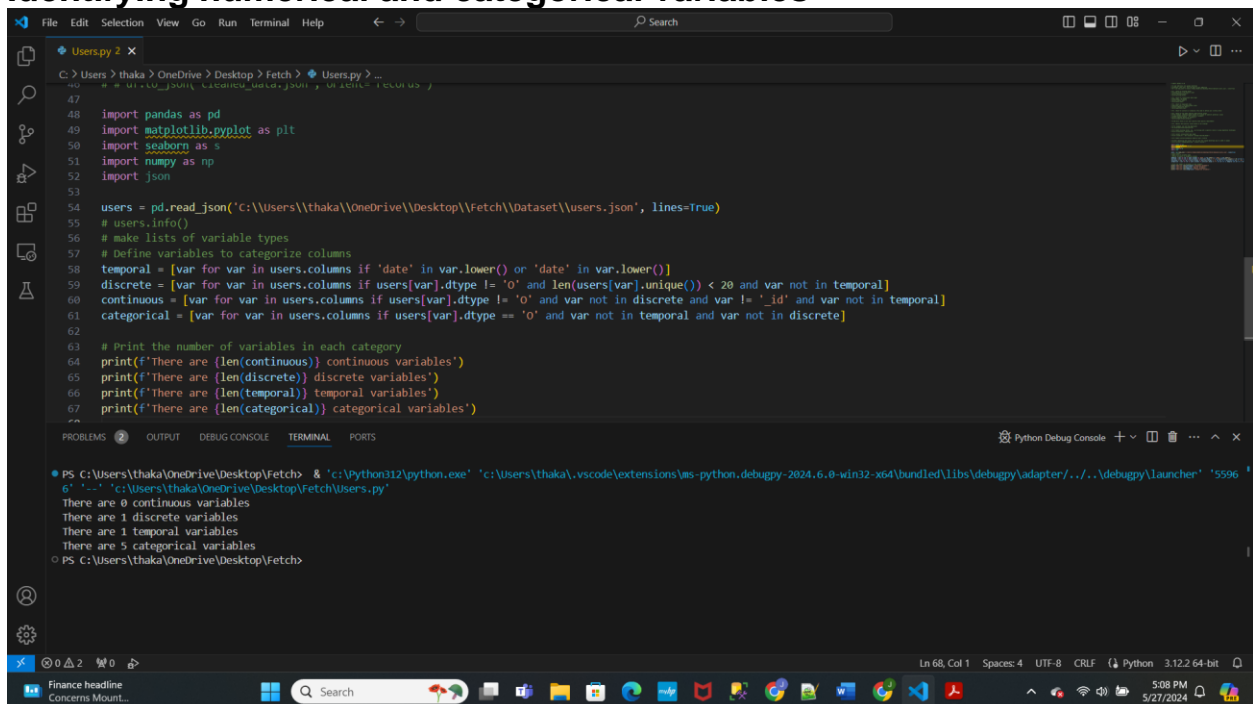
The screenshot shows a VS Code editor with a file named `Users.py` open. The code imports `pandas`, `matplotlib.pyplot`, `seaborn`, `numpy`, and `json`. It then reads a JSON file from the path `C:\Users\thaka\OneDrive\Desktop\Fetch\Dataset\users.json` using `pd.read_json` with `lines=True`. The `users.info()` method is called to display the dataset's structure.

```
47
48 import pandas as pd
49 import matplotlib.pyplot as plt
50 import seaborn as s
51 import numpy as np
52 import json
53
54 users = pd.read_json('C:\\Users\\thaka\\OneDrive\\Desktop\\Fetch\\Dataset\\users.json', lines=True)
55 users.info()
```

The terminal output shows the following information:

```
Data columns (total 7 columns):
# Column Non-Null Count Dtype
---
0 _id 495 non-null object
1 active 495 non-null bool
2 createdAt 495 non-null object
3 lastLogin 433 non-null object
4 role 495 non-null object
5 signUpSource 447 non-null object
6 state 439 non-null object
dtypes: bool(1), object(6)
memory usage: 23.8+ KB
```

# Identifying numerical and categorical variables



The screenshot shows the same VS Code editor with the `Users.py` file. The code has been updated to identify the types of variables in the dataset. It defines three lists: `temporal`, `discrete`, and `continuous`. It then prints the number of variables in each category.

```
47
48 import pandas as pd
49 import matplotlib.pyplot as plt
50 import seaborn as s
51 import numpy as np
52 import json
53
54 users = pd.read_json('C:\\Users\\thaka\\OneDrive\\Desktop\\Fetch\\Dataset\\users.json', lines=True)
55 # users.info()
56 # make lists of variable types
57 # define variables to categorize columns
58 temporal = [var for var in users.columns if 'date' in var.lower() or 'time' in var.lower()]
59 discrete = [var for var in users.columns if users[var].dtype != 'O' and len(users[var].unique()) < 20 and var not in temporal]
60 continuous = [var for var in users.columns if users[var].dtype != 'O' and var not in discrete and var != '_id' and var not in temporal]
61 categorical = [var for var in users.columns if users[var].dtype == 'O' and var not in temporal and var not in discrete]
62
63 # Print the number of variables in each category
64 print(f'There are {len(continuous)} continuous variables')
65 print(f'There are {len(discrete)} discrete variables')
66 print(f'There are {len(temporal)} temporal variables')
67 print(f'There are {len(categorical)} categorical variables')
68
```

The terminal output shows the following results:

```
PS C:\Users\thaka\OneDrive\Desktop\Fetch> & 'c:\Python312\python.exe' 'c:\Users\thaka\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher' '5596'
6
There are 0 continuous variables
There are 1 discrete variables
There are 1 temporal variables
There are 5 categorical variables
```

The screenshot shows a VS Code editor window with a Python script named `Users.py`. The script performs the following actions:

- Converts the `lastloginDate` column to a string type.
- Displays the first few rows of the DataFrame.
- Classifies variables into three categories: `temporal`, `discrete`, and `categorical`.
- Prints the count of variables in each category.

The terminal output shows the execution of the script, displaying the counts for each category:

```
PS C:\Users\thaka\OneDrive\Desktop\Fetch> & 'c:\python312\python.exe' 'c:\Users\thaka\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher' '5619' 1'...' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Users.py'
[active]
[createdDate]
['id', 'lastlogin', 'role', 'signupSource', 'state']
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

## Quantifying missing data

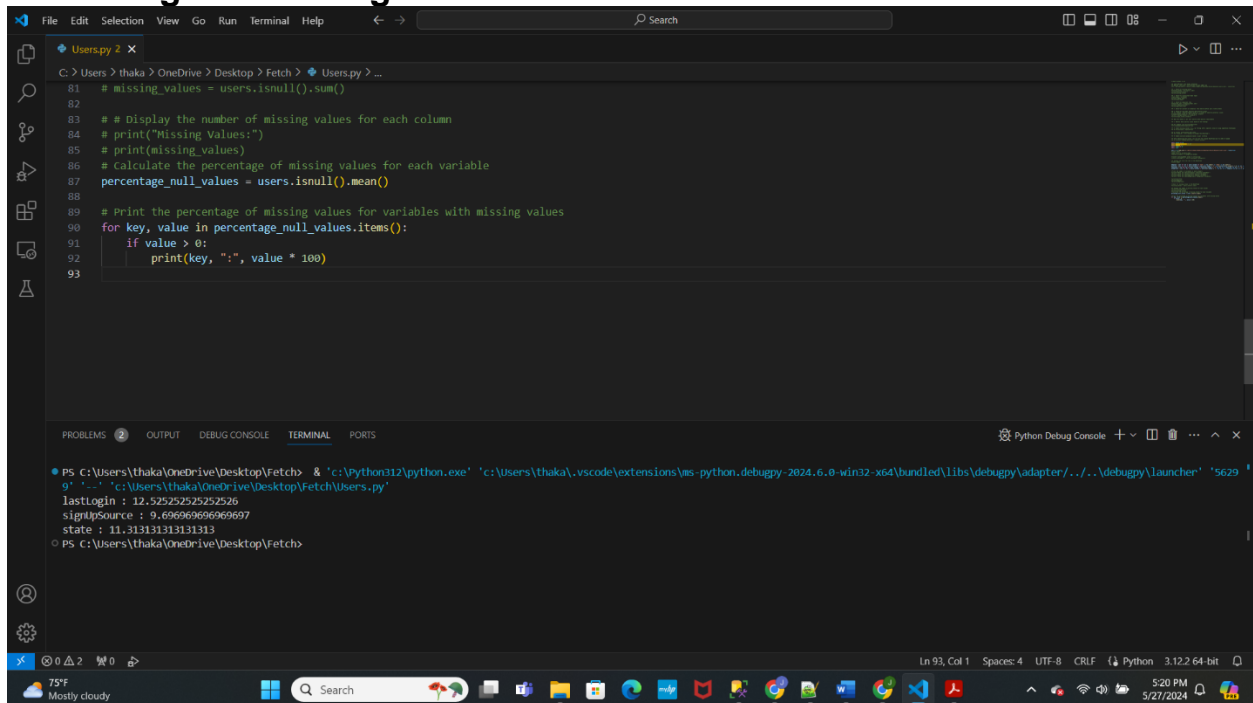
The screenshot shows a VS Code editor window with a Python script named `Users.py`. The script performs the following actions:

- Checks for missing values in the DataFrame using `users.isnull().sum()`.
- Displays the number of missing values for each column.

The terminal output shows the execution of the script, displaying the number of missing values for each column:

```
PS C:\Users\thaka\OneDrive\Desktop\Fetch> & 'c:\python312\python.exe' 'c:\Users\thaka\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher' '5623' 1'...' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Users.py'
Missing Values:
_id          0
active       0
createdDate  0
lastlogin    62
role         0
signupSource 48
state        56
dtype: int64
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

## Percentage of missing values in variables



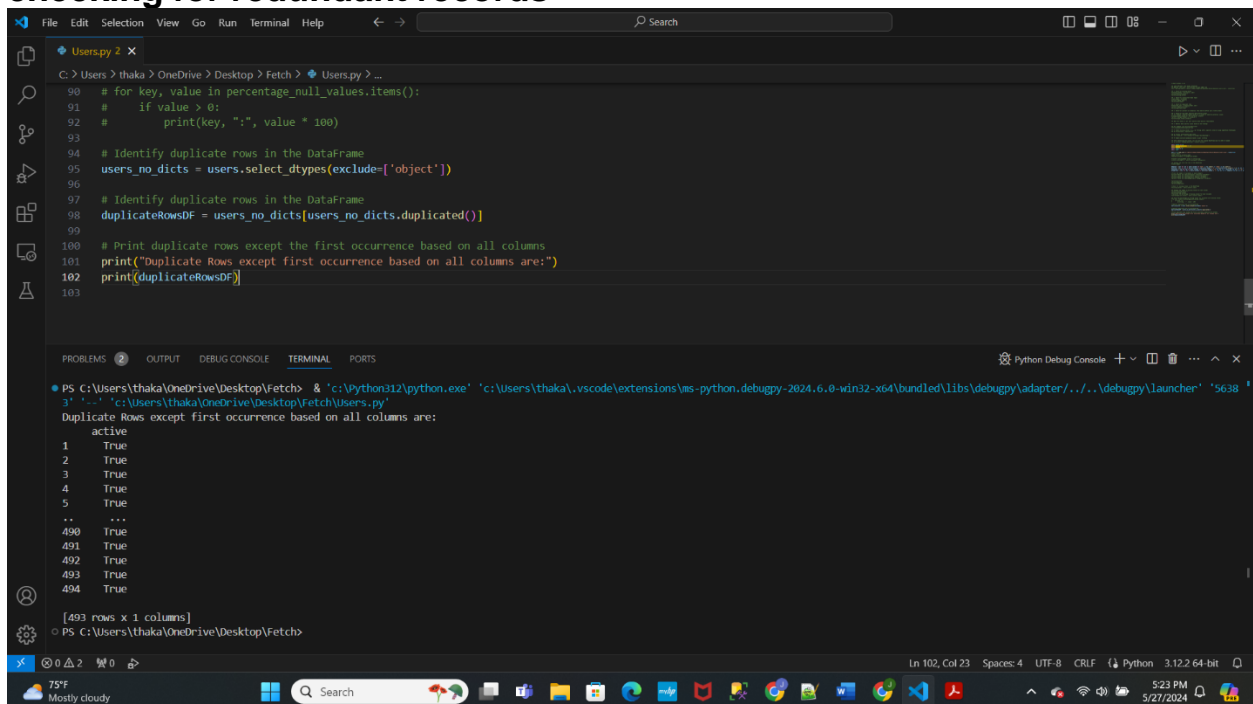
The screenshot shows a VS Code editor with a Python file named `Users.py`. The script calculates the percentage of missing values for each variable in a dataset. The terminal output shows the execution of the script, displaying the percentage of missing values for each variable.

```
C:\Users> thaka > OneDrive > Desktop > Fetch > Users.py > ...
81 # missing_values = users.isnull().sum()
82
83 ## Display the number of missing values for each column
84 # print("Missing Values:")
85 # print(missing_values)
86 # calculate the percentage of missing values for each variable
87 percentage_null_values = users.isnull().mean()
88
89 # Print the percentage of missing values for variables with missing values
90 for key, value in percentage_null_values.items():
91     if value > 0:
92         print(key, ":", value * 100)
93
```

Terminal Output:

```
PS C:\Users\thaka\OneDrive\Desktop\Fetch> & 'c:\Python312\python.exe' 'c:\Users\thaka\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher' '5629'
9' '-' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Users.py'
lastLogin : 12.525252525252526
signtSource : 9.696969696969697
state : 11.313131313131313
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

## checking for redundant records



The screenshot shows a VS Code editor with a Python file named `Users.py`. The script identifies duplicate rows in a DataFrame and prints the duplicate rows. The terminal output shows the execution of the script, displaying the duplicate rows.

```
90 # for key, value in percentage_null_values.items():
91 #     if value > 0:
92 #         print(key, ":", value * 100)
93
94 # Identify duplicate rows in the DataFrame
95 users_no_dicts = users.select_dtypes(exclude=['object'])
96
97 # Identify duplicate rows in the DataFrame
98 duplicateRowsDF = users_no_dicts[users_no_dicts.duplicated()]
99
100 # Print duplicate rows except the first occurrence based on all columns
101 print("Duplicate Rows except first occurrence based on all columns are:")
102 print(duplicateRowsDF)
103
```

Terminal Output:

```
PS C:\Users\thaka\OneDrive\Desktop\Fetch> & 'c:\Python312\python.exe' 'c:\Users\thaka\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher' '5638'
3' '-' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Users.py'
Duplicate Rows except first occurrence based on all columns are:
  active
1    True
2    True
3    True
4    True
5    True
..
490   True
491   True
492   True
493   True
494   True

[493 rows x 1 columns]
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

# Unique values of categorical variables

The image displays two screenshots of a VS Code editor window, showing the process of finding unique values in categorical variables using pandas.

**Top Screenshot:**

- The editor shows a file named `Users.py` with the following code:

```
95 # users_no_dicts = users.select_dtypes(exclude=['object'])
96
97 # Identify duplicate rows in the Dataframe
98 # duplicateRowsDF = users_no_dicts[users_no_dicts.duplicated()]
99
100 # Print duplicate rows except the first occurrence based on all columns
101 # print("Duplicate Rows except first occurrence based on all columns are:")
102 # print(duplicateRowsDF)
103 # Retrieve the unique values from the "role" column
104 unique_roles = users["role"].unique()
105
106 # Print the unique values
107 print("Unique values in the 'role' column:")
108 print(unique_roles)
109
```
- The terminal output shows the execution of the code:

```
PS C:\Users\thaka\OneDrive\Desktop\Fetch> & 'c:\Python312\python.exe' 'c:\Users\thaka\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher' '5642'
77 '...' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Users.py'
Unique values in the 'role' column:
['consumer' 'fetch-staff']
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

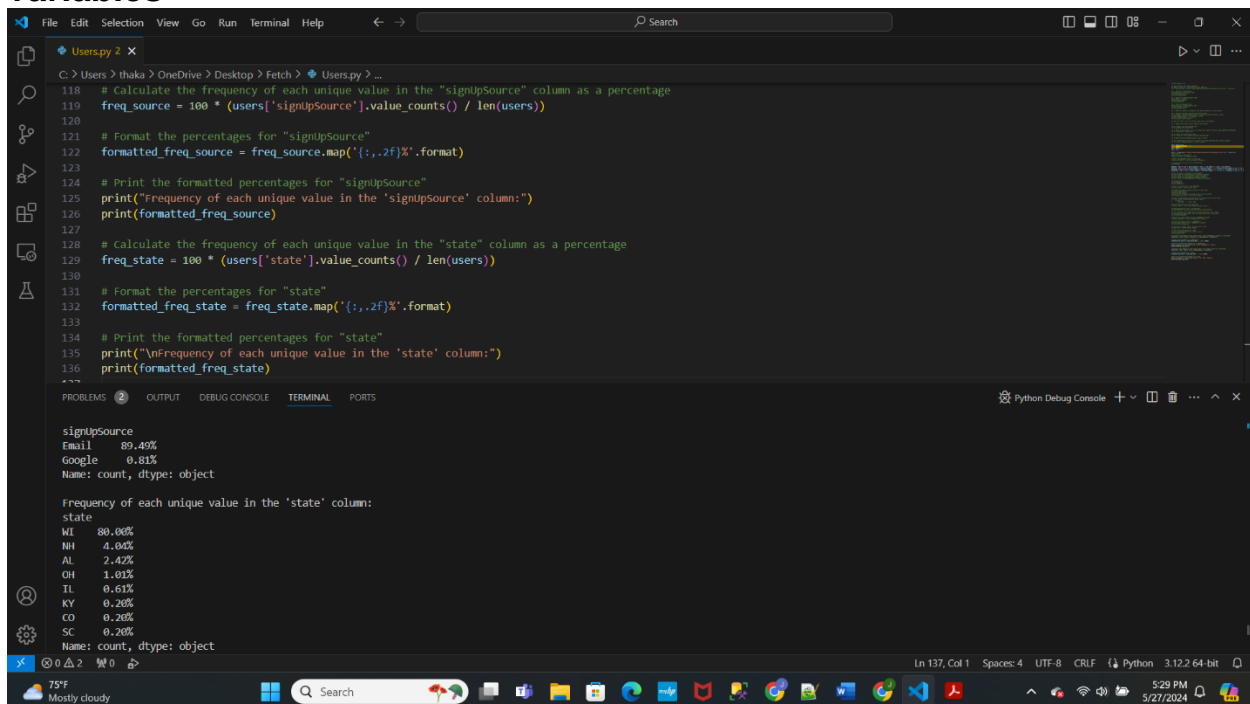
**Bottom Screenshot:**

- The editor shows the same file `Users.py` with the following code:

```
102 # print(duplicateRowsDF)
103
104 # Retrieve the unique values from the "signupSource" column
105 unique_signupSource = users["signupSource"].unique()
106
107 # Print the unique values for "signupSource"
108 print("Unique values in the 'signupSource' column:")
109 print(unique_signupSource)
110
111 # Retrieve the unique values from the "state" column
112 unique_state = users["state"].unique()
113
114 # Print the unique values for "state"
115 print("Unique values in the 'state' column:")
116 print(unique_state)
117
```
- The terminal output shows the execution of the code:

```
PS C:\Users\thaka\OneDrive\Desktop\Fetch> & 'c:\Python312\python.exe' 'c:\Users\thaka\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher' '5647'
81 '...' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Users.py'
Unique values in the 'signupSource' column:
['email' 'google' nan]
Unique values in the 'state' column:
['MI' 'KY' 'AL' 'CO' 'IL' nan 'OH' 'SC' 'NI']
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

## Examined percentage of different category values for categorical variables



The screenshot shows a Visual Studio Code editor window with a Python file named `Users.py`. The script calculates the percentage of unique values for two categorical variables: `signupSource` and `state`. The output is displayed in the terminal panel.

```
C:\Users> thaka > OneDrive > Desktop > Fetch > Users.py > ...
118 # Calculate the frequency of each unique value in the "signupSource" column as a percentage
119 freq_source = 100 * (users['signupSource'].value_counts() / len(users))
120
121 # Format the percentages for "signupSource"
122 formatted_freq_source = freq_source.map('{:,.2f}%'.format)
123
124 # Print the formatted percentages for "signupSource"
125 print("Frequency of each unique value in the 'signupSource' column:")
126 print(formatted_freq_source)
127
128 # Calculate the frequency of each unique value in the "state" column as a percentage
129 freq_state = 100 * (users['state'].value_counts() / len(users))
130
131 # Format the percentages for "state"
132 formatted_freq_state = freq_state.map('{:,.2f}%'.format)
133
134 # Print the formatted percentages for "state"
135 print("\nFrequency of each unique value in the 'state' column:")
136 print(formatted_freq_state)
```

Terminal Output:

```
signupSource
Email    89.45%
Google   0.81%
Name: count, dtype: object

Frequency of each unique value in the 'state' column:
state
WI    80.06%
NH    4.04%
AL    2.42%
OH    1.81%
IL    0.61%
KY    0.26%
CO    0.26%
SC    0.26%
Name: count, dtype: object
```

VS Code status bar: Ln 137, Col 1 | Spaces: 4 | UTF-8 | CRLF | Python 3.12.2 64-bit

Windows taskbar: 75°F Mostly cloudy | Search | 5:29 PM 5/27/2024