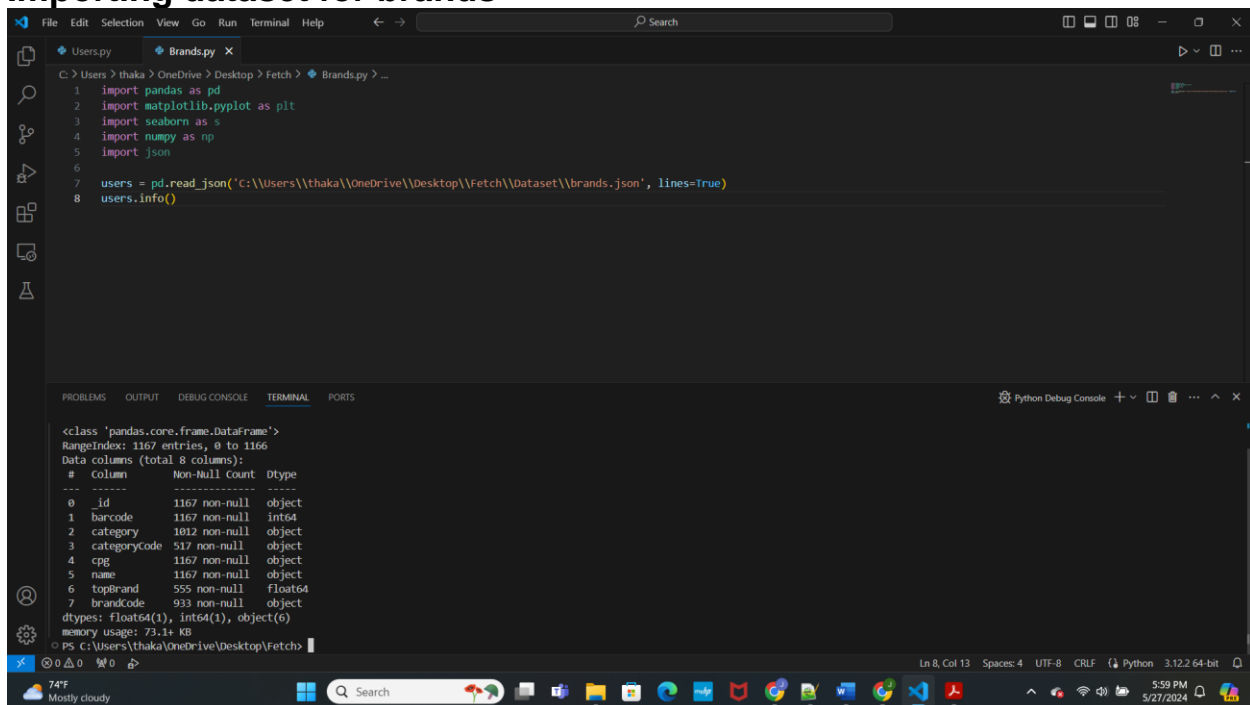


Importing dataset for brands



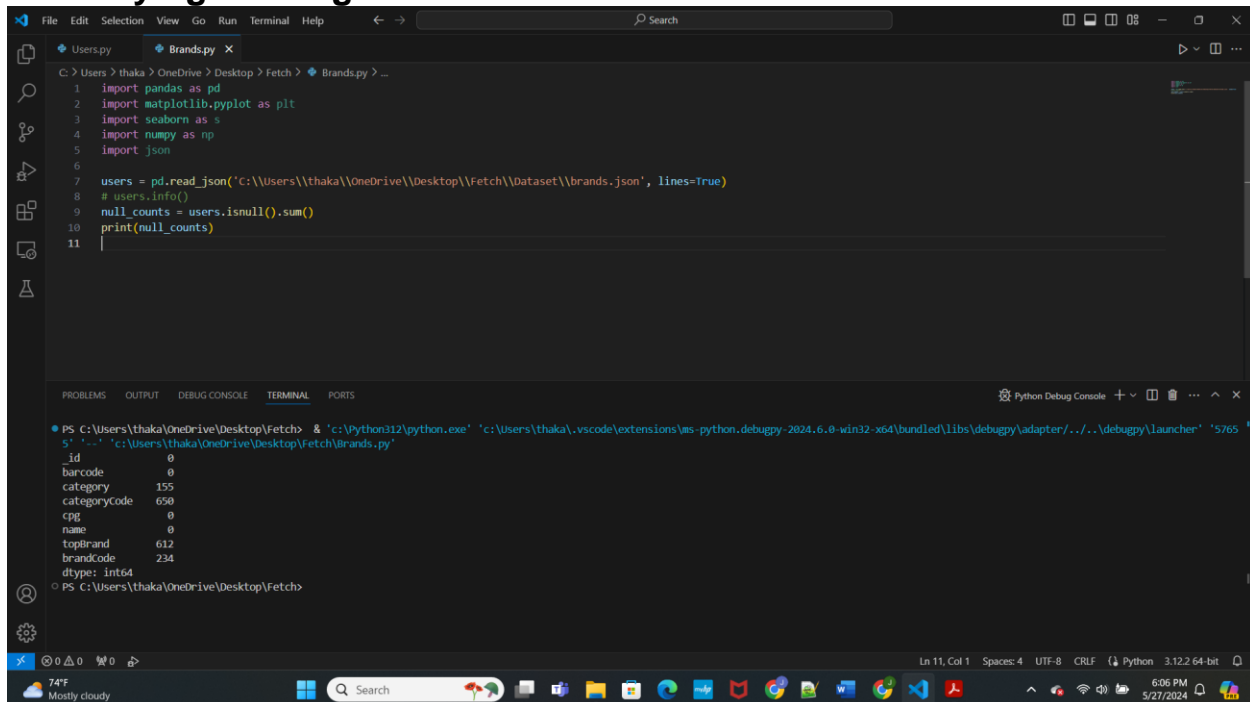
The screenshot shows a VS Code editor with a file named `Brands.py` open. The code imports `pandas`, `matplotlib.pyplot`, `seaborn`, `numpy`, and `json`. It then reads a JSON file from a local path and displays the DataFrame information in the terminal.

```
C:\Users> thaka > OneDrive > Desktop > Fetch > Brands.py > ...  
1 import pandas as pd  
2 import matplotlib.pyplot as plt  
3 import seaborn as s  
4 import numpy as np  
5 import json  
6  
7 users = pd.read_json('C:\\Users\\thaka\\OneDrive\\Desktop\\Fetch\\Dataset\\brands.json', lines=True)  
8 users.info()
```

The terminal output shows the DataFrame information:

```
<class 'pandas.core.frame.DataFrame'  
RangeIndex: 1167 entries, 0 to 1166  
Data columns (total 8 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   _id          1167 non-null   object  
1   barcode      1167 non-null   int64  
2   category     1012 non-null   object  
3   categoryCode  517 non-null    object  
4   cpg          1167 non-null   object  
5   name         1167 non-null   object  
6   topBrand     555 non-null    float64  
7   brandCode    933 non-null    object  
dtypes: float64(1), int64(1), object(6)  
memory usage: 73.1+ KB  
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

Quantifying missing data



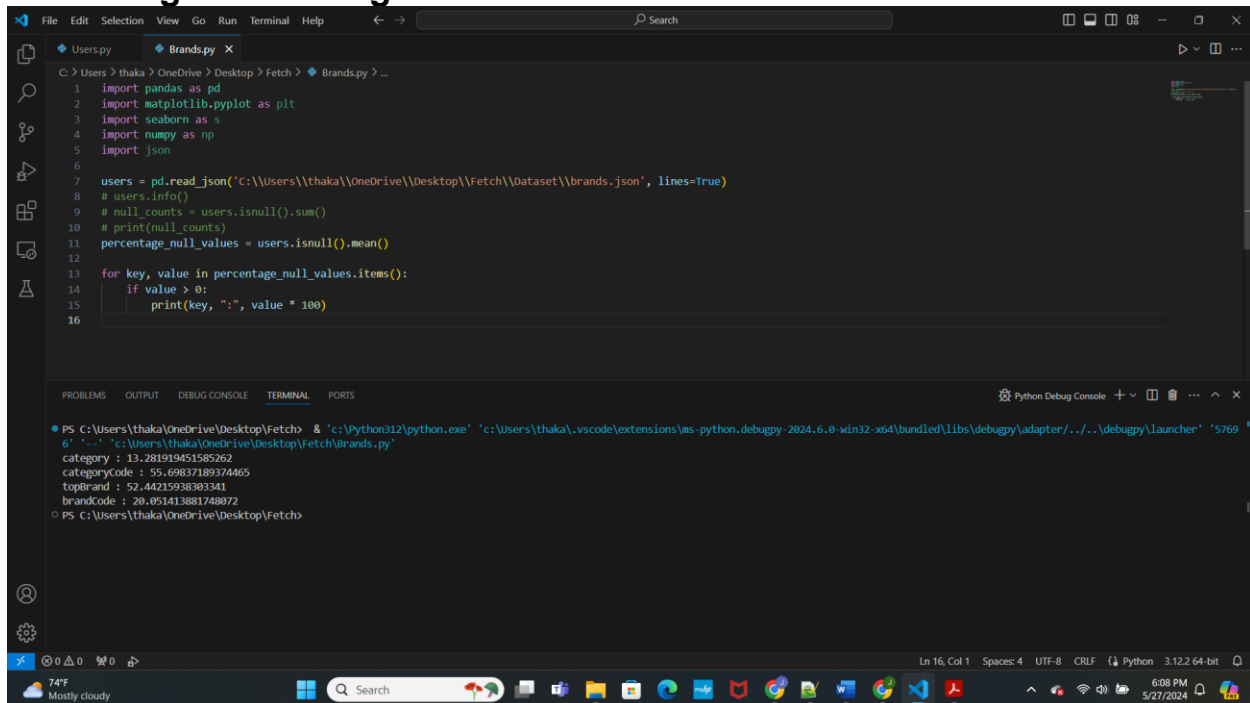
The screenshot shows the same VS Code editor with the `Brands.py` file. The code now includes an additional line to calculate the sum of null counts for each column and print the result. The terminal output shows the command to run the script and the resulting DataFrame information.

```
C:\Users> thaka > OneDrive > Desktop > Fetch > Brands.py > ...  
1 import pandas as pd  
2 import matplotlib.pyplot as plt  
3 import seaborn as s  
4 import numpy as np  
5 import json  
6  
7 users = pd.read_json('C:\\Users\\thaka\\OneDrive\\Desktop\\Fetch\\Dataset\\brands.json', lines=True)  
8 # users.info()  
9 null_counts = users.isnull().sum()  
10 print(null_counts)  
11
```

The terminal output shows the command to run the script and the resulting DataFrame information:

```
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' '5765'  
5' '-.' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Brands.py'  
_id          0  
barcode      0  
category     155  
categoryCode 650  
cpg          0  
name         0  
topBrand     612  
brandCode    234  
dtype: int64  
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

Percentage of missing values in variables



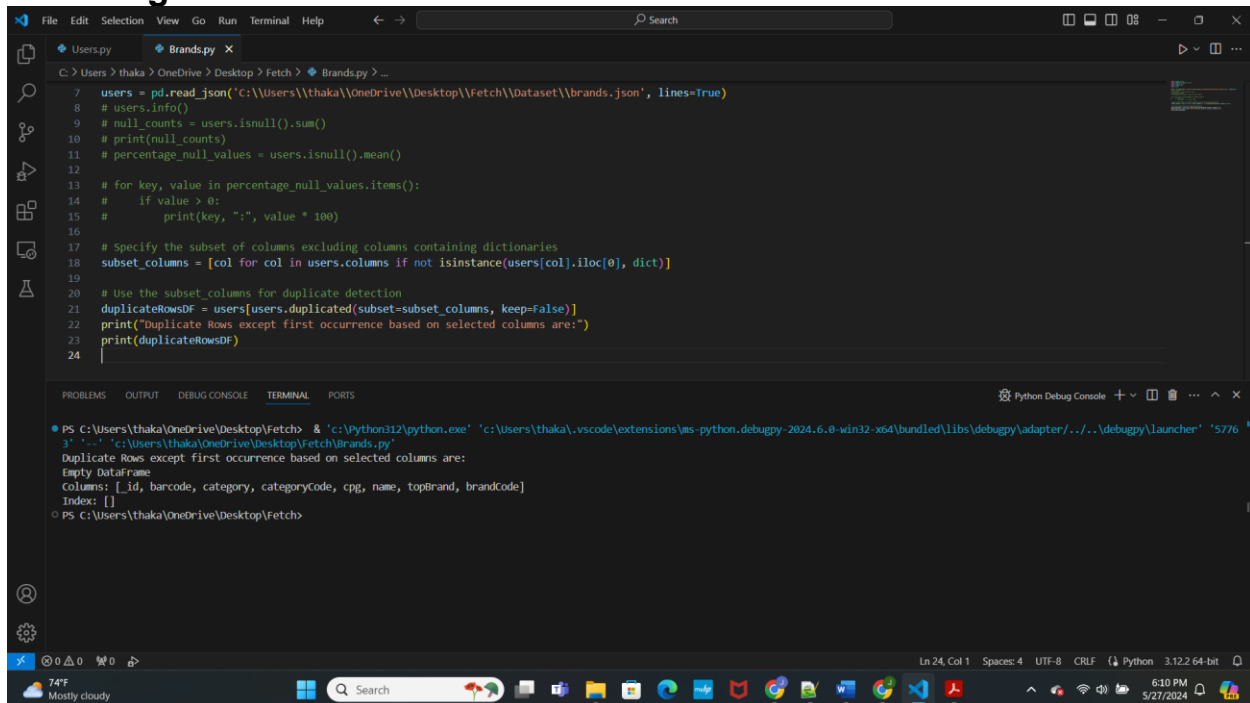
The screenshot shows a VS Code editor with a file named `Brands.py`. The script imports `pandas`, `matplotlib.pyplot`, `seaborn`, `numpy`, and `json`. It reads a JSON file `brands.json` into a DataFrame `users`. It then calculates the null counts for each column and the percentage of null values. The results are printed for columns where the percentage of null values is greater than 0.

```
C:\Users\thaka> OneDrive\Desktop\Fetch> Brands.py ...
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as s
4 import numpy as np
5 import json
6
7 users = pd.read_json('C:\\Users\\thaka\\OneDrive\\Desktop\\Fetch\\Dataset\\brands.json', lines=True)
8 # users.info()
9 # null_counts = users.isnull().sum()
10 # print(null_counts)
11 percentage_null_values = users.isnull().mean()
12
13 for key, value in percentage_null_values.items():
14     if value > 0:
15         print(key, ":", value * 100)
16
```

The terminal output shows the execution of the script, displaying the null counts and the percentage of null 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' '5769'
6 '-' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Brands.py'
category : 13.281919451585262
categoryCode : 55.69837189374465
topBrand : 52.44215938303341
brandCode : 20.051413881748072
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

Checking for redundant records



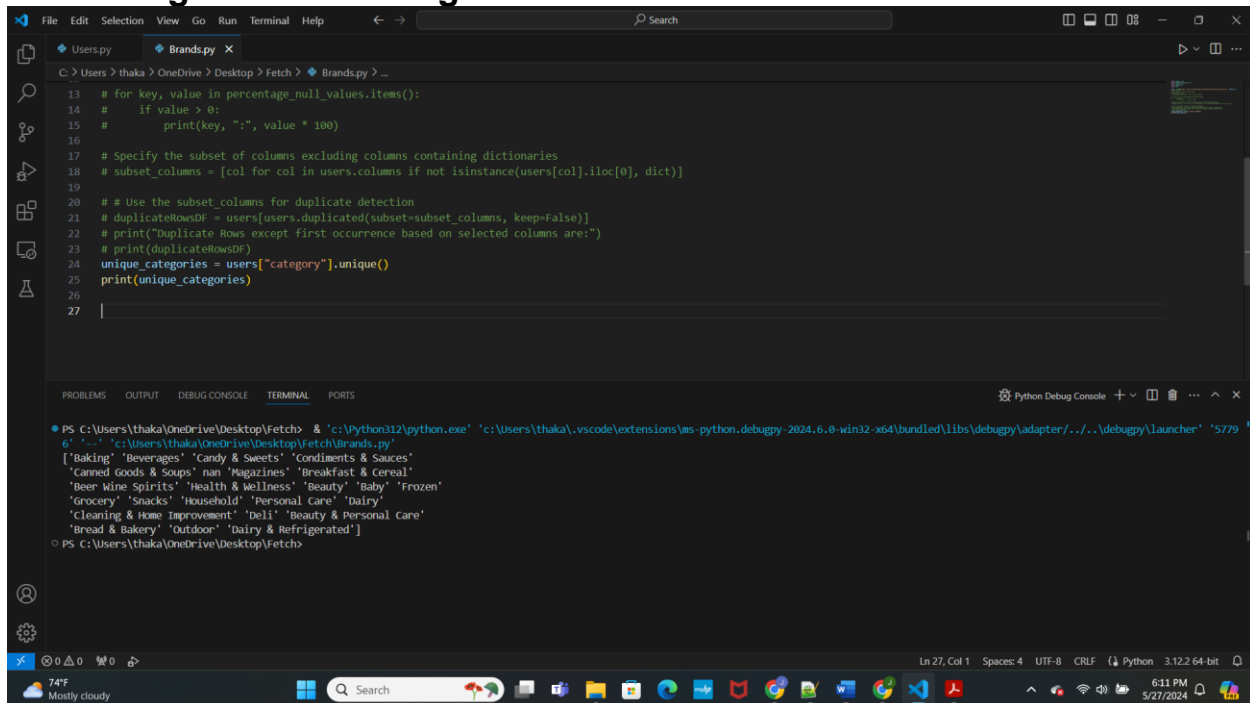
The screenshot shows a VS Code editor with a file named `Brands.py`. The script reads a JSON file `brands.json` into a DataFrame `users`. It then identifies duplicate rows based on a subset of columns (excluding columns containing dictionaries). The results are printed, showing the duplicate rows and the columns used for detection.

```
C:\Users\thaka> OneDrive\Desktop\Fetch> Brands.py ...
7 users = pd.read_json('C:\\Users\\thaka\\OneDrive\\Desktop\\Fetch\\Dataset\\brands.json', lines=True)
8 # users.info()
9 # null_counts = users.isnull().sum()
10 # print(null_counts)
11 # percentage_null_values = users.isnull().mean()
12
13 # for key, value in percentage_null_values.items():
14 #     if value > 0:
15 #         print(key, ":", value * 100)
16
17 # Specify the subset of columns excluding columns containing dictionaries
18 subset_columns = [col for col in users.columns if not isinstance(users[col].iloc[0], dict)]
19
20 # Use the subset_columns for duplicate detection
21 duplicateRowsDF = users[users.duplicated(subset=subset_columns, keep=False)]
22 print("Duplicate Rows except first occurrence based on selected columns are:")
23 print(duplicateRowsDF)
24
```

The terminal output shows the execution of the script, displaying the duplicate rows and the columns used for detection:

```
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' '5776'
3' '-' 'c:\Users\thaka\OneDrive\Desktop\Fetch\Brands.py'
Duplicate Rows except first occurrence based on selected columns are:
Empty DataFrame
Columns: f_id, barcode, category, categoryCode, cpg, name, topBrand, brandCode
Index: []
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

Examining values of categorical variables

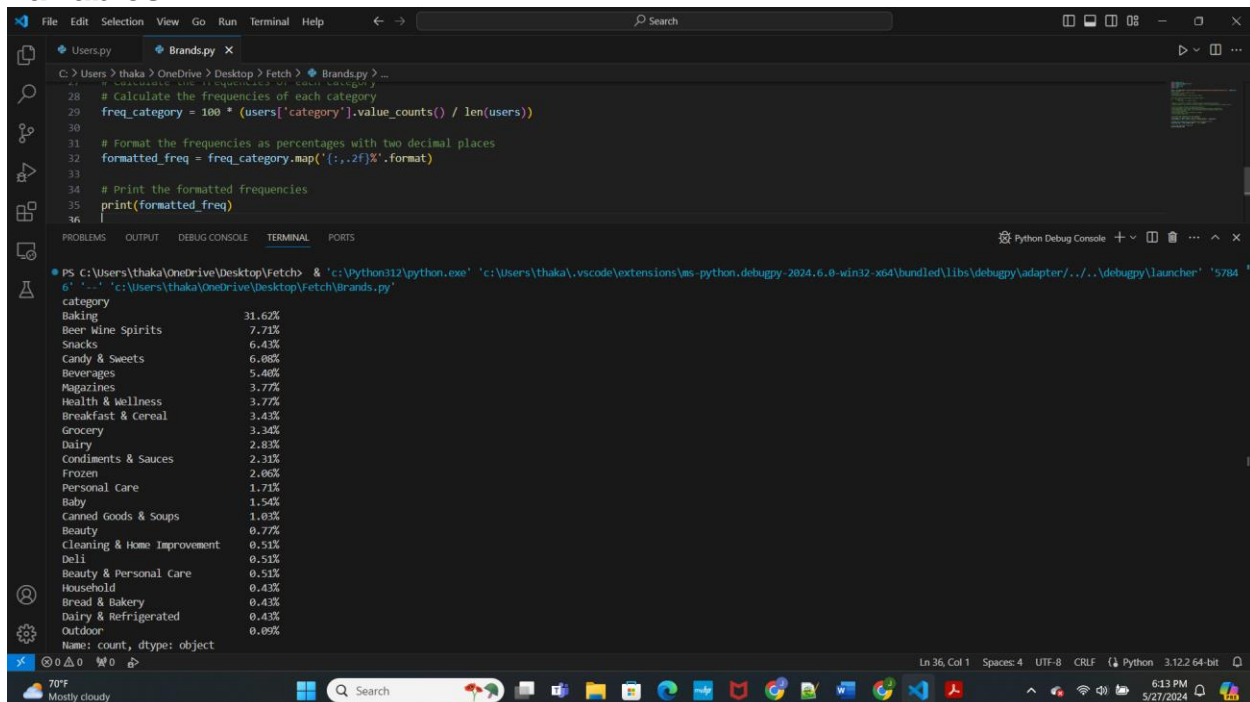


The screenshot shows a VS Code editor with a file named `Brands.py`. The script processes a dataset of brands, removing null values, identifying duplicates, and listing unique categories. The terminal output shows the unique categories for the 'category' variable.

```
13 # for key, value in percentage_null_values.items():
14 #     if value > 0:
15 #         print(key, ":", value * 100)
16
17 # Specify the subset of columns excluding columns containing dictionaries
18 # subset_columns = [col for col in users.columns if not isinstance(users[col].iloc[0], dict)]
19
20 # Use the subset_columns for duplicate detection
21 # duplicateRowsDF = users[users.duplicated(subset=subset_columns, keep=False)]
22 # print("Duplicate Rows except first occurrence based on selected columns are:")
23 # print(duplicateRowsDF)
24 unique_categories = users["category"].unique()
25 print(unique_categories)
26
27 |
```

```
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' '5779'
6' ... 'c:\Users\thaka\OneDrive\Desktop\Fetch\Brands.py'
['Baking', 'Beverages', 'Candy & Sweets', 'Condiments & Sauces',
 'Canned Goods & Soups', 'Magazines', 'Breakfast & Cereal',
 'Beer Wine Spirits', 'Health & Wellness', 'Beauty', 'Baby', 'Frozen',
 'Grocery', 'Snacks', 'Household', 'Personal Care', 'Dairy',
 'Cleaning & Home Improvement', 'Deli', 'Beauty & Personal Care',
 'Bread & Bakery', 'Outdoor', 'Dairy & Refrigerated']
PS C:\Users\thaka\OneDrive\Desktop\Fetch>
```

Examining percentage of different category values for categorical variables



The screenshot shows a VS Code editor with a file named `Brands.py`. The script calculates the percentage of each category in the dataset. The terminal output displays a list of categories with their corresponding percentages.

```
27 # Calculate the frequencies of each category
28 # Calculate the frequencies of each category
29 freq_category = 100 * (users["category"].value_counts() / len(users))
30
31 # Format the frequencies as percentages with two decimal places
32 formatted_freq = freq_category.map('{:.2f}%'.format)
33
34 # Print the formatted frequencies
35 print(formatted_freq)
36
37 |
```

```
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' '5784'
6' ... 'c:\Users\thaka\OneDrive\Desktop\Fetch\Brands.py'
category
Baking                31.62%
Beer Wine Spirits     7.71%
Snacks                6.43%
Candy & Sweets        6.08%
Beverages             5.40%
Magazines             3.77%
Health & Wellness     3.77%
Breakfast & Cereal    3.43%
Grocery              3.24%
Dairy                2.83%
Condiments & Sauces  2.31%
Frozen               2.06%
Personal Care        1.71%
Baby                1.54%
Canned Goods & Soups 1.03%
Beauty              0.77%
Cleaning & Home Improvement 0.51%
Deli                0.51%
Beauty & Personal Care 0.51%
Household           0.43%
Bread & Bakery       0.43%
Dairy & Refrigerated 0.43%
Outdoor             0.09%
Name: count, dtype: object
```