Theory Activity No. 1

Name: Om Vilas Kale

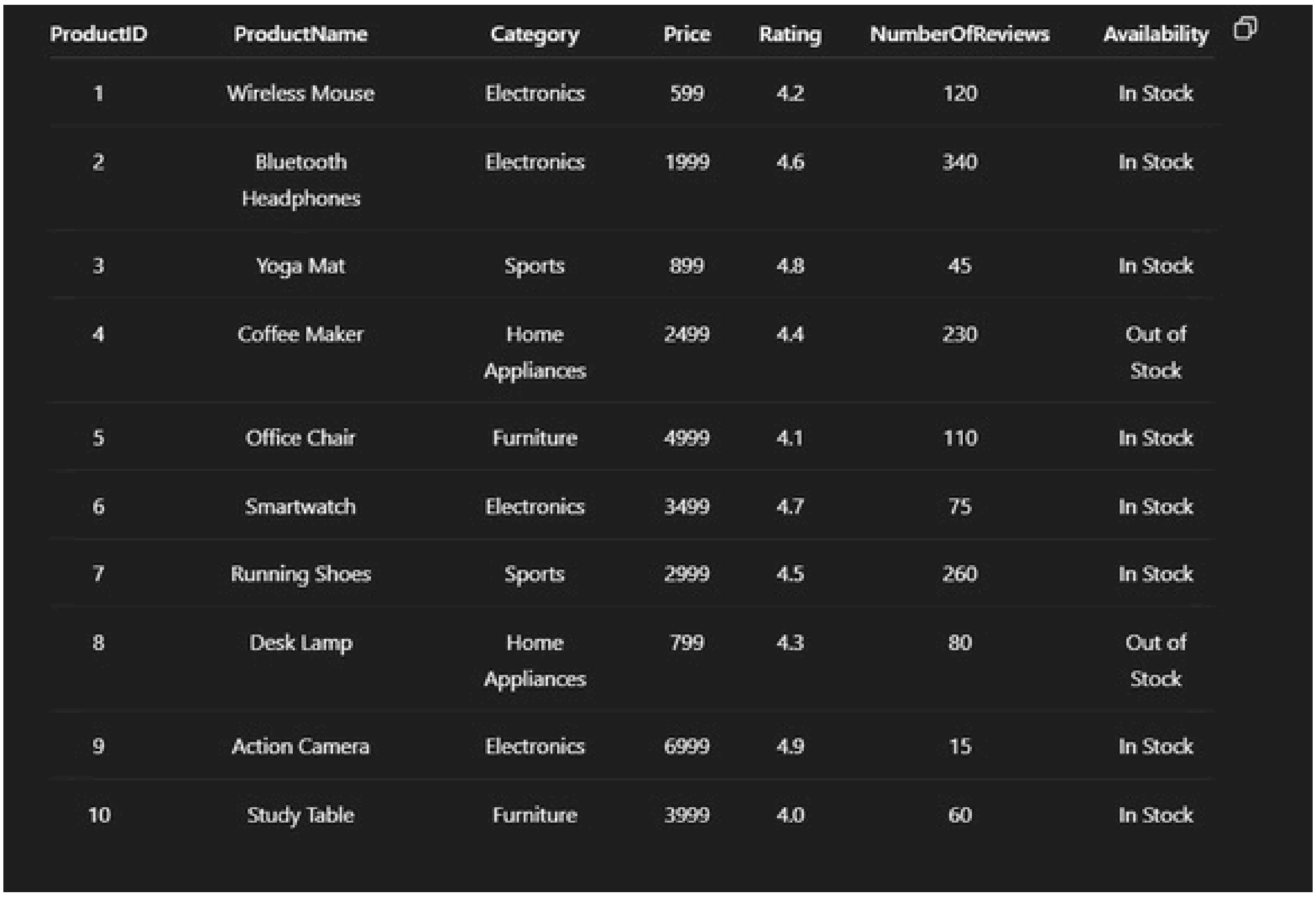
PRN : 202401040118

Roll No : CS2-18

Div : CS2

Dataset : Amazon Product Dataset

Dataset URL : Amazon-Product-Dataset



ProductID	ProductName	Category	Price	Rating	NumberOfReviews	Availability
1	Wireless Mouse	Electronics	599	4.2	120	In Stock
2	Bluetooth Headphones	Electronics	1999	4.6	340	In Stock
3	Yoga Mat	Sports	899	4.8	45	In Stock
4	Coffee Maker	Home Appliances	2499	4.4	230	Out of Stock
5	Office Chair	Furniture	4999	4.1	110	In Stock
6	Smartwatch	Electronics	3499	4.7	75	In Stock
7	Running Shoes	Sports	2999	4.5	260	In Stock
8	Desk Lamp	Home Appliances	799	4.3	80	Out of Stock
9	Action Camera	Electronics	6999	4.9	15	In Stock
10	Study Table	Furniture	3999	4.0	60	In Stock

First, your imports:

import pandas as pd import numpy as np

df = pd.read_csv('amazon_products.csv')

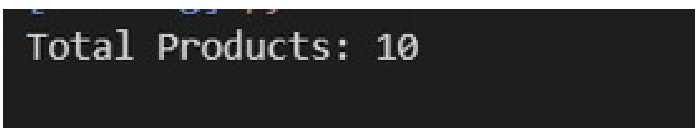
Problem Statements

1. Total number of products:

Solution:

total_products = len(df) print("Total number of products:", total_products)

Output:



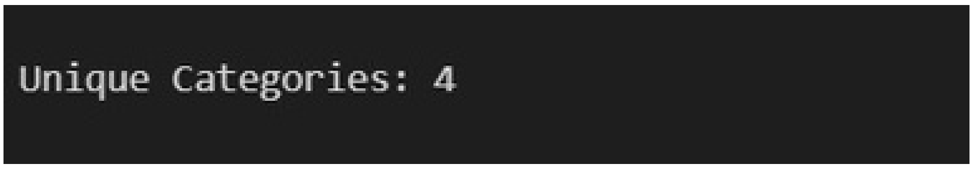
```
Total Products: 10
```

1. Unique product categories:

Solution:

unique_categories = df['Category'].nunique() print("Number of unique categories:", unique_categories)

Output:



```
Unique Categories: 4
```

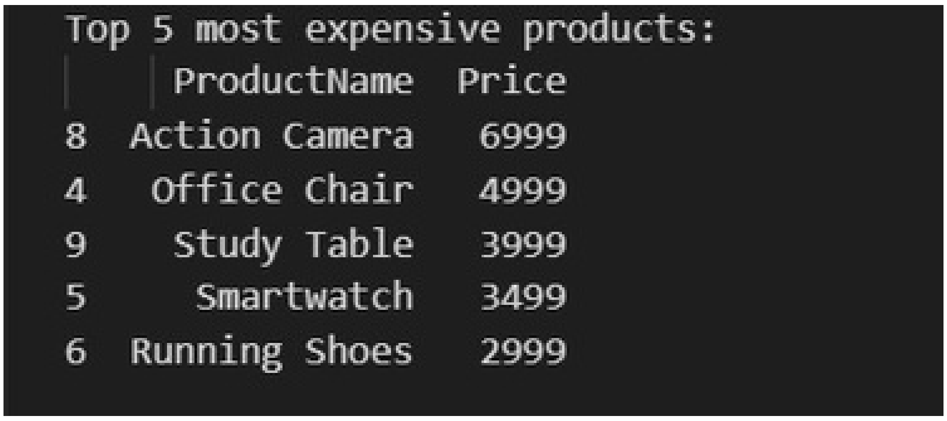
1. Top 5 most expensive products:

Solution:

top5_expensive = df.sort_values(by='Price', ascending=False).head(5)

print("Top 5 most expensive products:") print(top5_expensive[['ProductName', 'Price']])

Output:



```
Top 5 most expensive products:
     ProductName  Price
8  Action Camera   6999
4   Office Chair   4999
9    Study Table   3999
5     Smartwatch   3499
6  Running Shoes   2999
```

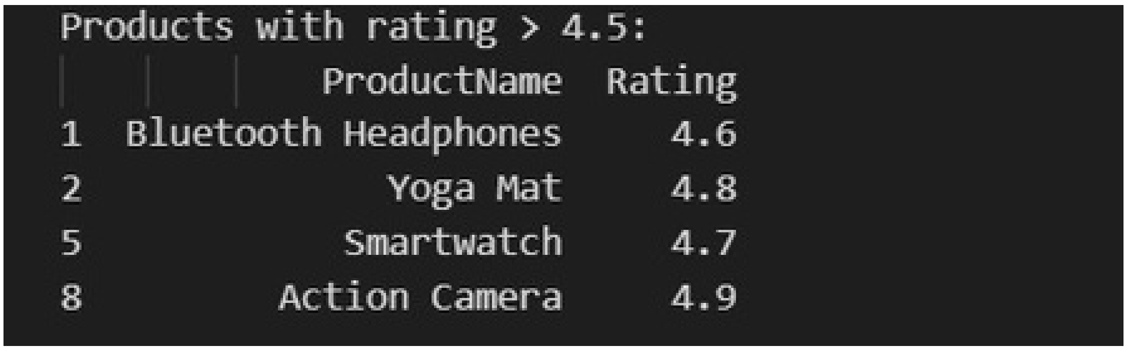
1. Find products with rating > 4.5:

Solution:

high_rated_products = df[df['Rating'] > 4.5] print("Products with rating > 4.5:")

print(high_rated_products[['ProductName', 'Rating']])

Output:



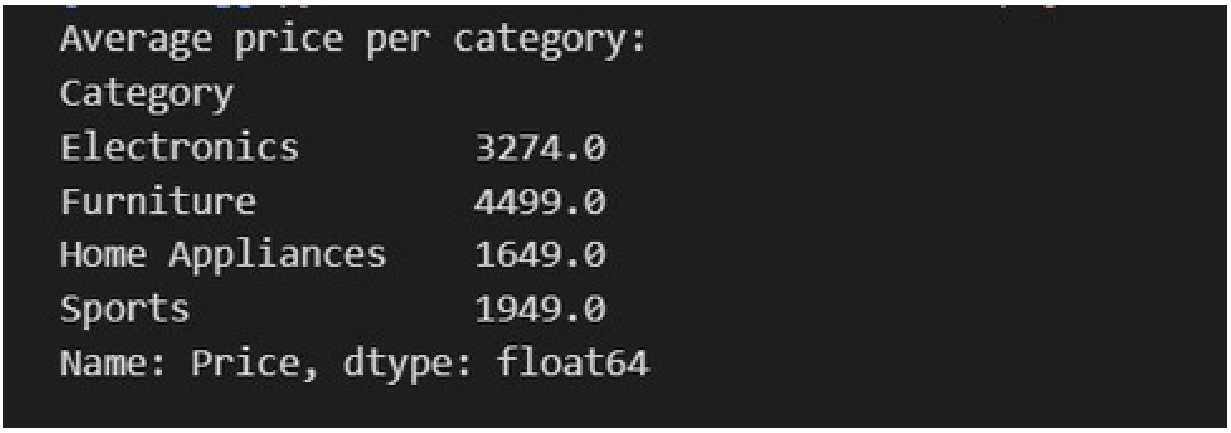
```
Products with rating > 4.5:
            ProductName  Rating
1  Bluetooth Headphones     4.6
2              Yoga Mat     4.8
5            Smartwatch     4.7
8         Action Camera     4.9
```

1. Find the average price of products in each category:

Solution:

avg_price_per_category = df.groupby('Category')['Price'].mean() print("Average price per category:") print(avg_price_per_category)

Output:

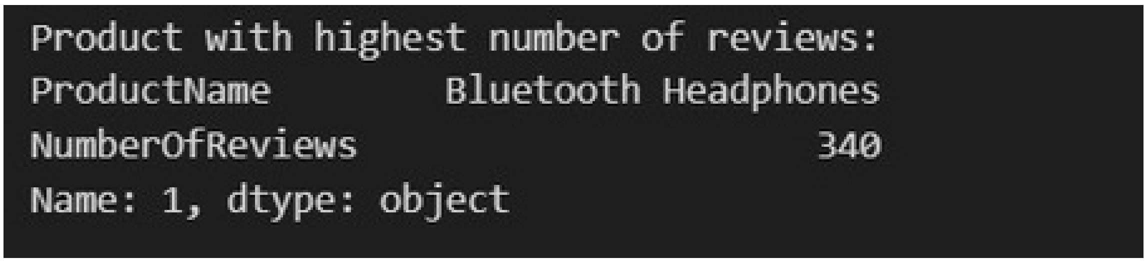

```
Average price per category:
Category
Electronics         3274.0
Furniture           4499.0
Home Appliances     1649.0
Sports              1949.0
Name: Price, dtype: float64
```

1. Find the product with the highest number of reviews:

Solution:

most_reviewed_product = df.loc[df['NumberOfReviews'].idxmax()] print("Product with highest number of reviews:") print(most_reviewed_product[['ProductName', 'NumberOfReviews']])

Output:

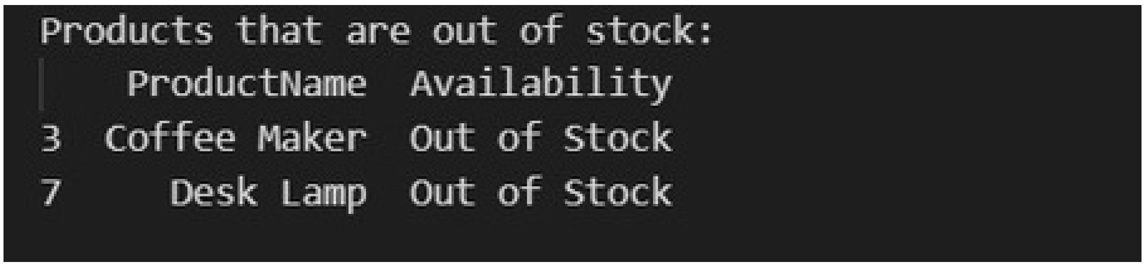

```
Product with highest number of reviews:
ProductName        Bluetooth Headphones
NumberOfReviews                     340
Name: 1, dtype: object
```

1. Find products that are currently out of stock:

Solution:

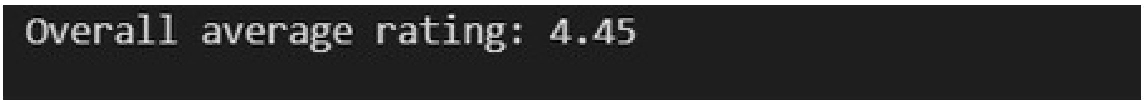
out_of_stock_products = df[df['Availability'] == 'Out of Stock'] print("Products that are out of stock:")

print(out_of_stock_products[['ProductName', 'Availability']]) Output:


```
Products that are out of stock:
    ProductName  Availability
3  Coffee Maker  Out of Stock
7     Desk Lamp  Out of Stock
```

1. Calculate the overall average rating:

Solution: overall_average_rating = df['Rating'].mean() print("Overall average rating:", overall_average_rating) Output:



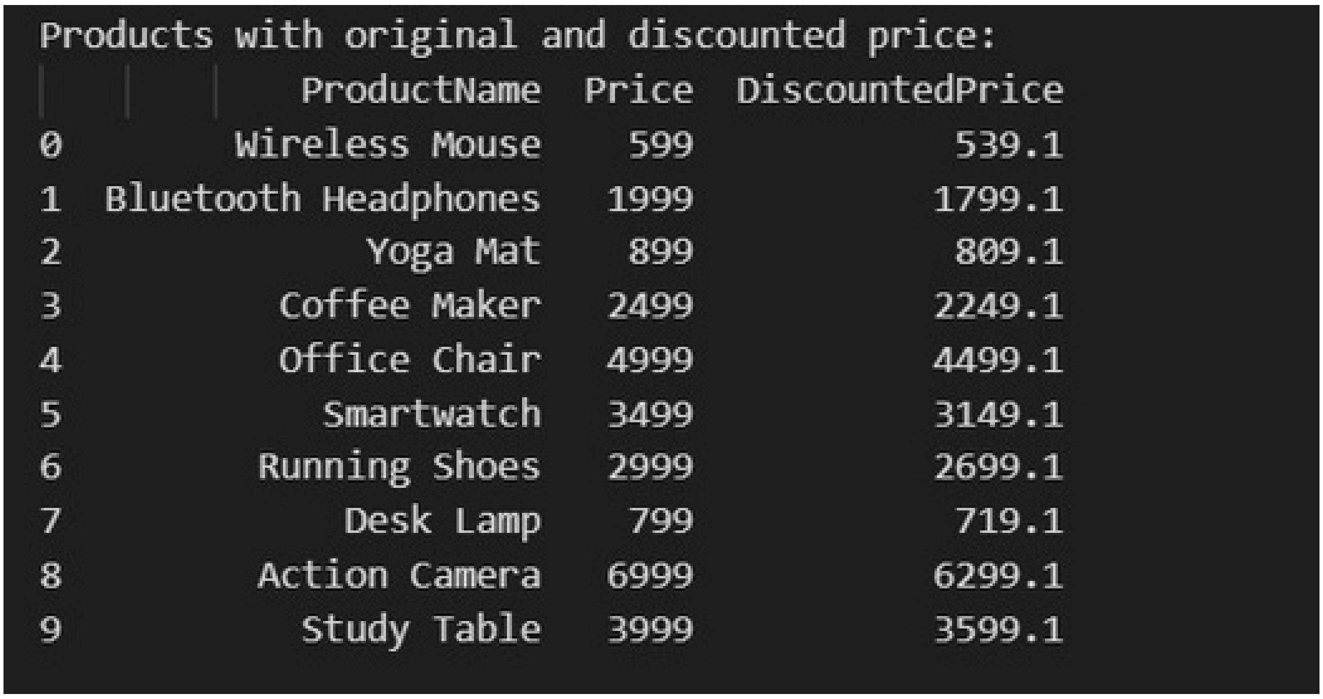
```
Overall average rating: 4.45
```

1. Create a new column for 10% discounted price:

Solution:

df['DiscountedPrice'] = df['Price'] * 0.9

print("Products with original and discounted price:") print(df[['ProductName', 'Price', 'DiscountedPrice']])
Output:



```
Products with original and discounted price:
             ProductName  Price  DiscountedPrice
0         Wireless Mouse    599            539.1
1  Bluetooth Headphones    1999           1799.1
2               Yoga Mat    899            809.1
3           Coffee Maker   2499           2249.1
4           Office Chair   4999           4499.1
5             Smartwatch   3499           3149.1
6         Running Shoes    2999           2699.1
7              Desk Lamp    799            719.1
8          Action Camera   6999           6299.1
9            Study Table   3999           3599.1
```

1. Find underrated products (high rating >4.5 but reviews <20):

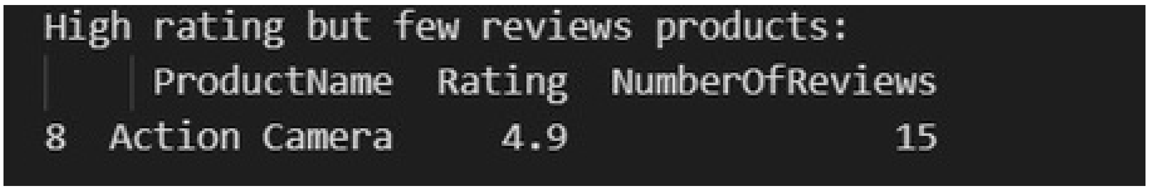
Solution:

underrated_products = df[(df['Rating'] > 4.5) &

(df['NumberOfReviews'] < 20)]

print("High rating but few reviews products:") print(underrated_products[['ProductName', 'Rating', 'NumberOfReviews']])

Output:



```
High rating but few reviews products:
      ProductName  Rating  NumberOfReviews
8  Action Camera     4.9               15
```

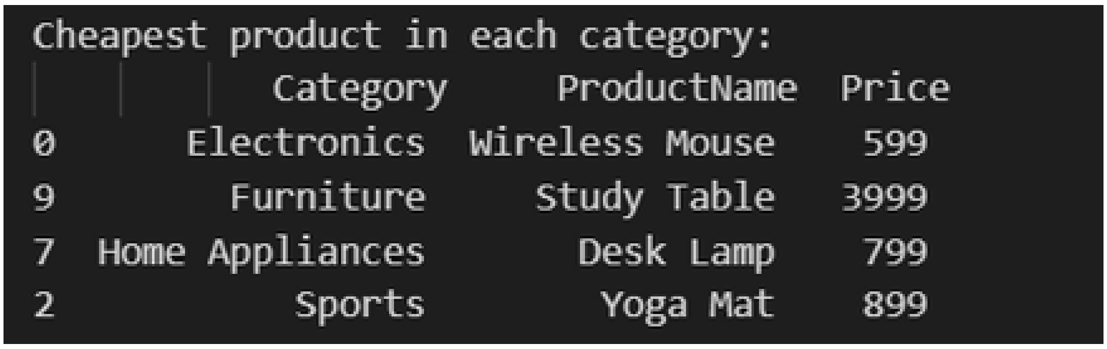
11. Find the cheapest product in each category:

Solution:

cheapest_in_category = df.loc[df.groupby('Category')['Price'].idxmin()]

print("\nCheapest product in each category:\n", cheapest_in_category[['Category', 'ProductName', 'Price']])

Output:



```
Cheapest product in each category:
            Category     ProductName  Price
0       Electronics  Wireless Mouse    599
9         Furniture     Study Table   3999
7   Home Appliances       Desk Lamp    799
2            Sports        Yoga Mat    899
```

12. List all products priced above the average price:

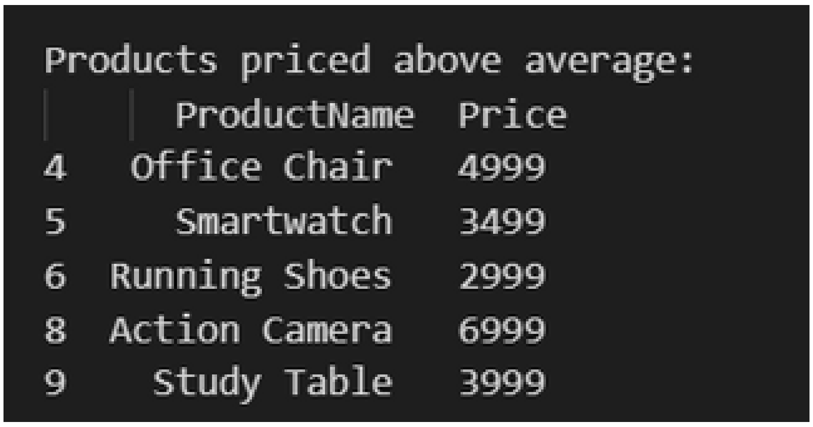
Solution:

average_price = df['Price'].mean()

above_avg_products = df[df['Price'] > average_price]

print("\nProducts priced above average:\n", above_avg_products[['ProductName', 'Price']])

Output:



```
Products priced above average:
      ProductName  Price
4    Office Chair   4999
5      Smartwatch   3499
6   Running Shoes   2999
8   Action Camera   6999
9     Study Table   3999
```

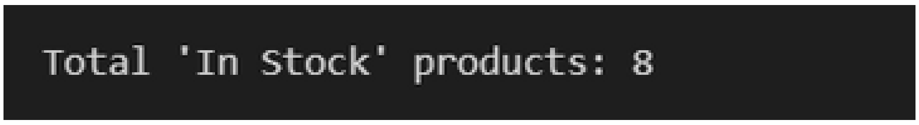
13. Total number of products "In Stock":

Solution:

in_stock_count = df[df['Availability'] == 'In Stock'].shape[0]

print("\nTotal 'In Stock' products:", in_stock_count)

Output:



```
Total 'In Stock' products: 8
```

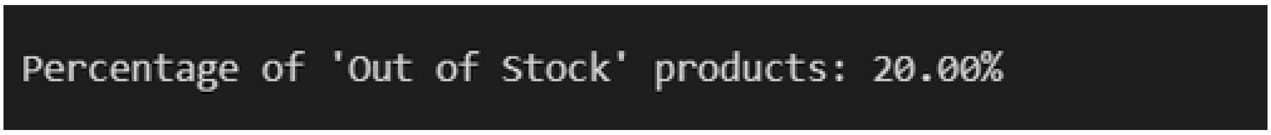
14. Percentage of products "Out of Stock":

Solution:

out_of_stock_percentage = (df[df['Availability'] == 'Out of Stock'].shape[0] / len(df)) * 100

print("\nPercentage of 'Out of Stock' products: {:.2f}%".format(out_of_stock_percentage))

Output:



```
Percentage of 'Out of Stock' products: 20.00%
```

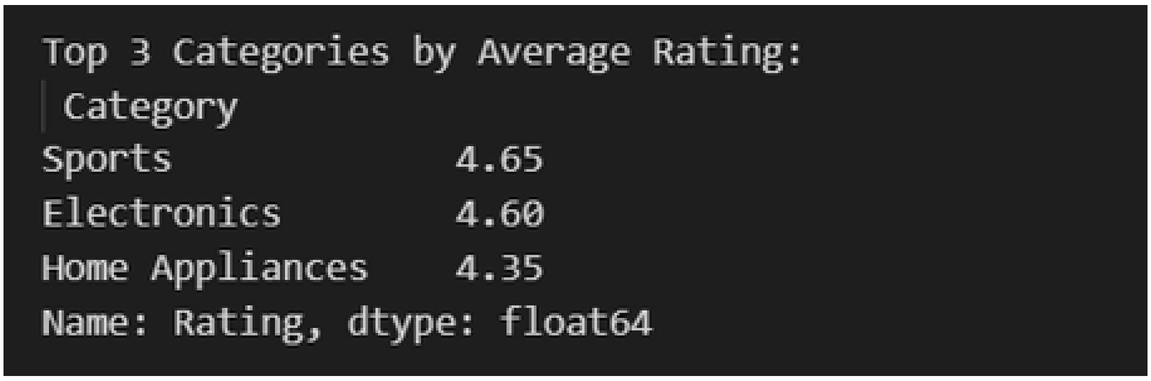
15. Top 3 categories with highest average ratings:

Solution:

top3_categories_rating = df.groupby('Category')['Rating'].mean().sort_values(ascending=False).head(3)

print("\nTop 3 Categories by Average Rating:\n", top3_categories_rating)

Output:



```
Top 3 Categories by Average Rating:
 Category
Sports              4.65
Electronics         4.60
Home Appliances     4.35
Name: Rating, dtype: float64
```

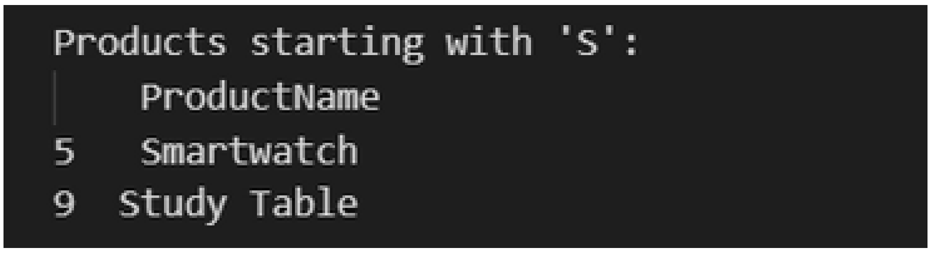
16. Products whose name starts with 'S':

Solution:

products_starting_S = df[df['ProductName'].str.startswith('S')]

print("\nProducts starting with 'S':\n", products_starting_S[['ProductName']])

Output:



```
 Products starting with 'S':
     ProductName
 5    Smartwatch
 9  Study Table
```

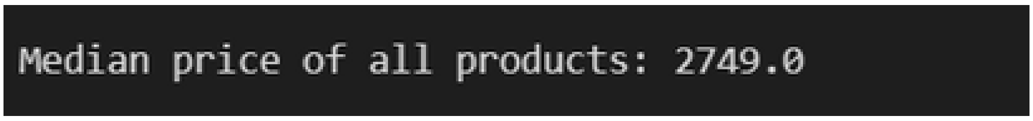
17. Median price of all products:

Solution:

median_price = df['Price'].median()

print("\nMedian price of all products:", median_price)

Output:



```
Median price of all products: 2749.0
```

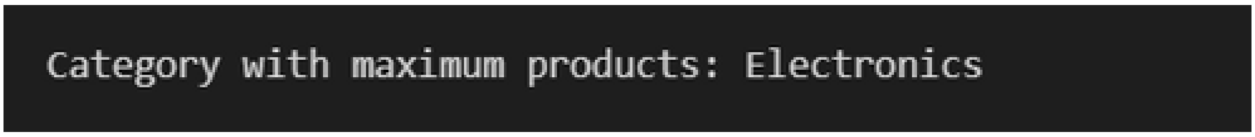
18. Category with maximum number of products:

Solution:

max_products_category = df['Category'].value_counts().idxmax()

print("\nCategory with maximum products:", max_products_category)

Output:



```
Category with maximum products: Electronics
```

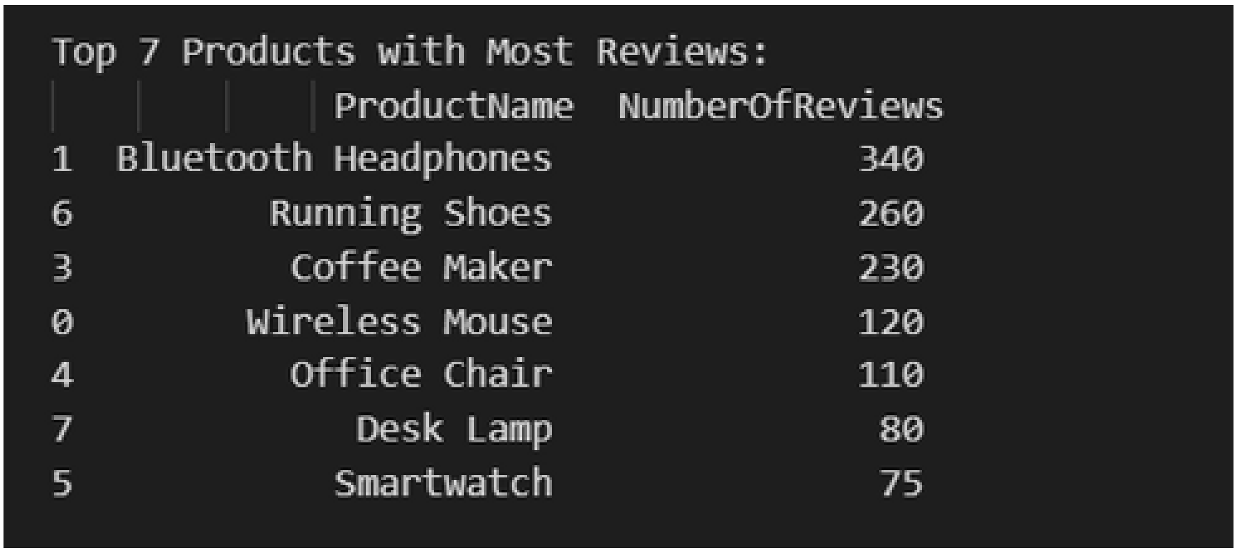
19. Top 7 products with most reviews:

Solution:

top7_most_reviews = df.sort_values(by='NumberOfReviews', ascending=False).head(7)

print("\nTop 7 Products with Most Reviews:\n", top7_most_reviews[['ProductName', 'NumberOfReviews']])

Output:



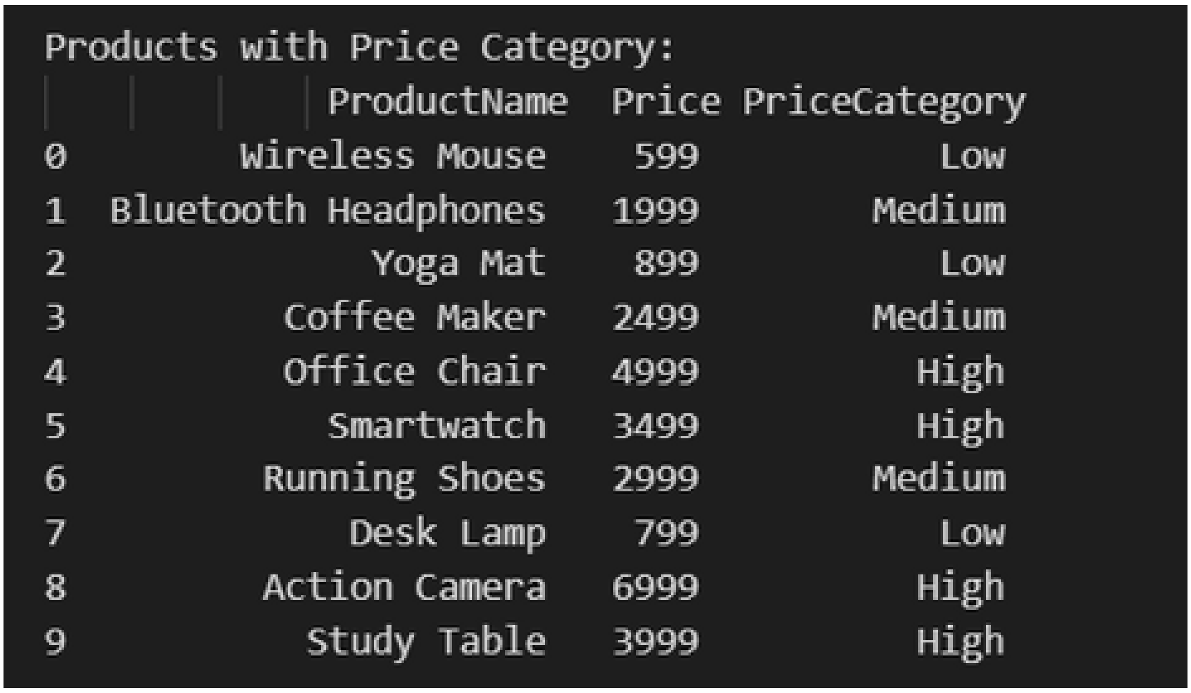
```
Top 7 Products with Most Reviews:
             ProductName  NumberOfReviews
1  Bluetooth Headphones              340
6         Running Shoes              260
3          Coffee Maker              230
0        Wireless Mouse              120
4          Office Chair              110
7             Desk Lamp               80
5            Smartwatch               75
```

20. Create PriceCategory column (High/Medium/Low):

Solution:

```
def price_category(price):
    if price > 3000:
        return 'High'
    elif price >= 1000:
        return 'Medium'
    else:
        return 'Low'
df['PriceCategory'] = df['Price'].apply(price_category)
print("\nProducts with Price Category:\n", df[['ProductName', 'Price', 'PriceCategory']])
```

Output:



```
Products with Price Category:
             ProductName  Price PriceCategory
0         Wireless Mouse    599           Low
1  Bluetooth Headphones   1999        Medium
2               Yoga Mat    899           Low
3           Coffee Maker   2499        Medium
4           Office Chair   4999          High
5             Smartwatch   3499          High
6         Running Shoes   2999        Medium
7              Desk Lamp    799           Low
8          Action Camera   6999          High
9            Study Table   3999          High
```