Welcome to my EDA

This time we're gonna explore the data from amazone website

Firstly, lets import the module that we need

```
In [22]:  import matplotlib.pyplot as plt
          import pandas as pd
          import seaborn as sns
```

Here's the csv that i got from laptop category on amazon

```
In [23]:  testing = pd.read_csv("amazon_laptop_products.csv")
```

The data not perfect yet, so let's fix it!

Start from generalize the format

```
In [24]:  testing["price"] = (
              testing["price"]
              .astype(str)
              .str.replace("$", "", regex=False)
              .str.replace(",", "", regex=False)
              .astype(float)
          )

          testing["rating"] = pd.to_numeric(testing["rating"], errors="coerce")
          testing["total_buy"] = (
              testing["total_buy"]
              .astype(str)
              .str.replace(",", "", regex=False)
              .astype(float)
          )
```

Were noticing that the data we're not complete

```
In [25]:  testing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   title      312 non-null    object
 1   price      284 non-null    float64
 2   rating     296 non-null    float64
 3   total_buy  296 non-null    float64
dtypes: float64(3), object(1)
memory usage: 9.9+ KB
```

Drop it off

```
In [26]:  testing.fillna(0, inplace=True)
          testing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   title      312 non-null    object
 1   price      312 non-null    float64
 2   rating     312 non-null    float64
 3   total_buy  312 non-null    float64
dtypes: float64(3), object(1)
memory usage: 9.9+ KB
```

Split it by space and get the first 6 words

```python
In [27]: temp_column = testing["title"]
         temp_column = temp_column.str.replace(",", "")
         new_title = temp_column.astype(str).str.split(" ",expand=True).iloc[:,0:6
         new_title
```

Out[27]:

	0	1	2	3	4	5
0	HP	14	Laptop	Intel	Celeron	N4020
1	Laptop	Laptop	Computer	with	Intel	N97
2	HP	Laptop	Computer	for	Student	&
3	ASUS	ROG	Strix	G16	(2025)	Gaming
4	Laptop	16-inch	8GB	DDR	256GB	SSD
...	...	...	...	...	...	...
307	Kipling	Women's	Seoul	Extra	Large	17"
308	2025	Gaming	Laptop	15.6	Inch	Laptop
309	Lenovo	IdeaPad	3	Laptop	15.6"	FHD
310	15.6"	Laptop	Screen	Extender	1080P	FHD
311	Lenovo	Ideapad	3i	15.6	FHD	Touchscreen

312 rows × 6 columns

Merge it in one column

```python
In [28]: new_title["merged"] = new_title[0] + " " + new_title[1] + " " + new_title
         new_title["merged"] = new_title["merged"].str.title()
         new_title
```

`Out[28]:`

	0	1	2	3	4	5	merged
0	HP	14	Laptop	Intel	Celeron	N4020	Hp 14 Laptop Intel Celeron N4020
1	Laptop	Laptop	Computer	with	Intel	N97	Laptop Laptop Computer With Intel N97
2	HP	Laptop	Computer	for	Student	&	Hp Laptop Computer For Student &
3	ASUS	ROG	Strix	G16	(2025)	Gaming	Asus Rog Strix G16 (2025) Gaming
4	Laptop	16-inch	8GB	DDR	256GB	SSD	Laptop 16-Inch 8Gb Ddr 256Gb Ssd
...	...	...	...	...	...	...	...
307	Kipling	Women's	Seoul	Extra	Large	17"	Kipling Women'S Seoul Extra Large 17"
308	2025	Gaming	Laptop	15.6	Inch	Laptop	2025 Gaming Laptop 15.6 Inch Laptop
309	Lenovo	IdeaPad	3	Laptop	15.6"	FHD	Lenovo Ideapad 3 Laptop 15.6" Fhd
310	15.6"	Laptop	Screen	Extender	1080P	FHD	15.6" Laptop Screen Extender 1080P Fhd
311	Lenovo	Ideapad	3i	15.6	FHD	Touchscreen	Lenovo Ideapad 3I 15.6 Fhd Touchscreen

312 rows × 7 columns

Change the title before

`In [29]:`
```
testing["title"] = new_title["merged"]
testing
```

|  | title | price | rating | total_buy |
|---|---|---|---|---|
| 0 | Hp 14 Laptop Intel Celeron N4020 | 172.16 | 4.1 | 3877.0 |
| 1 | Laptop Laptop Computer With Intel N97 | 369.00 | 5.0 | 16.0 |
| 2 | Hp Laptop Computer For Student & | 0.00 | 4.6 | 343.0 |
| 3 | Asus Rog Strix G16 (2025) Gaming | 1274.99 | 4.6 | 80.0 |
| 4 | Laptop 16-Inch 8Gb Ddr 256Gb Ssd | 199.99 | 5.0 | 16.0 |
| ... | ... | ... | ... | ... |
| 307 | Kipling Women'S Seoul Extra Large 17" | 72.00 | 4.7 | 806.0 |
| 308 | 2025 Gaming Laptop 15.6 Inch Laptop | 549.99 | 3.8 | 4.0 |
| 309 | Lenovo Ideapad 3 Laptop 15.6" Fhd | 499.00 | 4.2 | 273.0 |
| 310 | 15.6" Laptop Screen Extender 1080P Fhd | 279.99 | 4.6 | 323.0 |
| 311 | Lenovo Ideapad 3I 15.6 Fhd Touchscreen | 0.00 | 4.8 | 41.0 |

312 rows × 4 columns

**Is there any correlation?**

```
In [30]: corr = testing[["price", "rating", "total_buy"]].corr()

sns.heatmap(corr, annot=True, cmap="coolwarm", center=0)
plt.title("Correlation Heatmap")
plt.show()
```


Correlation Heatmap
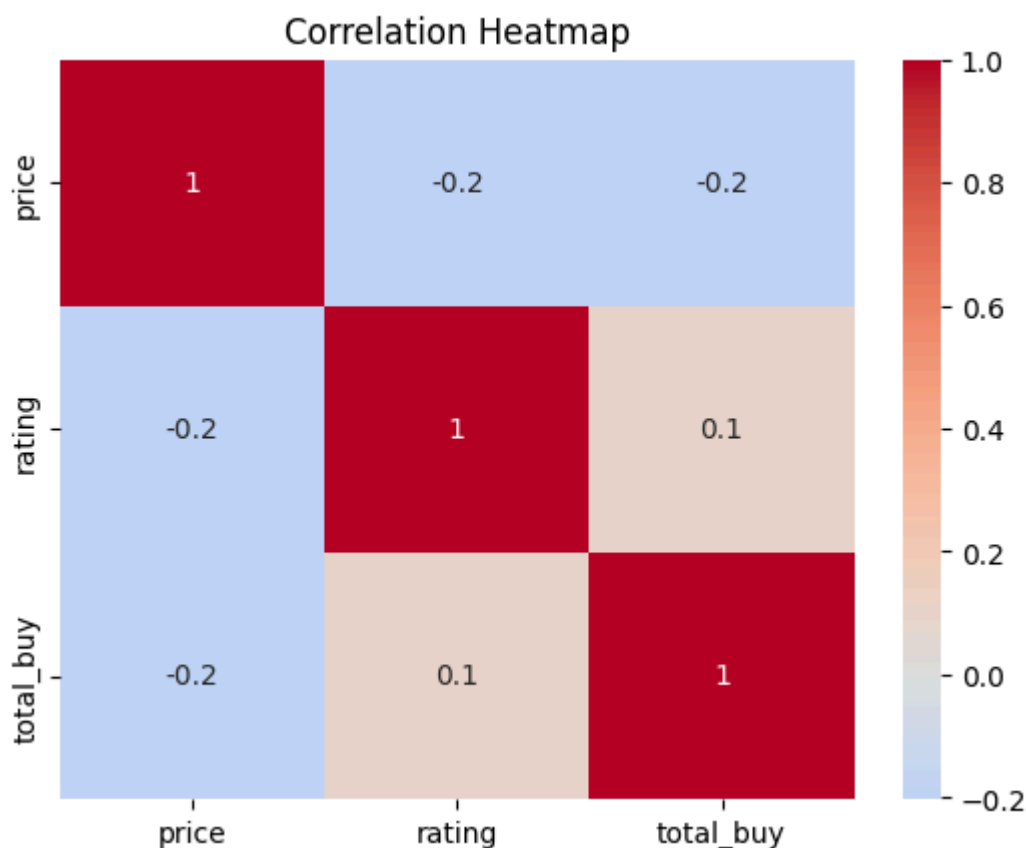
More closer the index to 1, more correlation the columns are

Visualize the most expensive laptop by rating
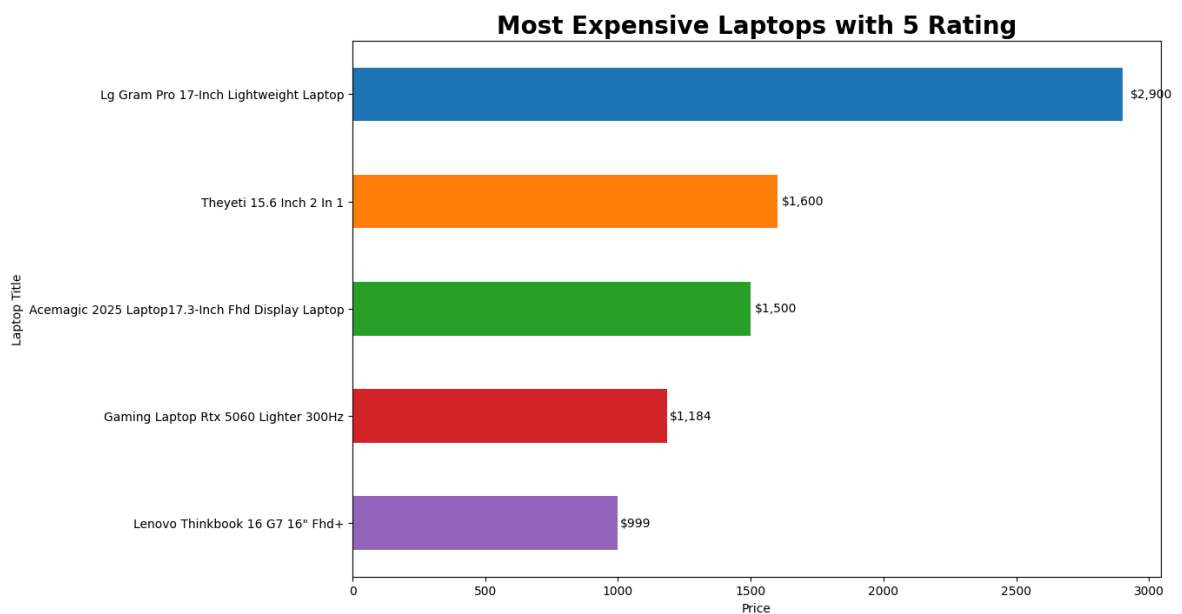
```
In [31]: good_laptops = (
             testing[testing["rating"] == 5]
             .sort_values(by="price", ascending=False)
             .head(5)
             .reset_index(drop=True)
         )

         ax = good_laptops.plot(
             kind="barh",
             x="title",
             y="price",
             legend=False,
             figsize=(12, 8),
             color=plt.cm.tab10.colors[:len(good_laptops)]

         )

         plt.title("Most Expensive Laptops with 5 Rating", fontsize=20, weight="bo
         plt.xlabel("Price")
         plt.ylabel("Laptop Title")

         for p in ax.patches:
             ax.text(
                 p.get_width() + (p.get_width()*0.01),
                 p.get_y() + p.get_height()/2,
                 f"${p.get_width():,.0f}",
                 va="center"
             )
         ax.invert_yaxis()

         plt.show()
```

Most Expensive Laptops with 5 Rating

Laptop	Price
Lg Gram Pro 17-Inch Lightweight Laptop	$2,900
Theyeti 15.6 Inch 2 In 1	$1,600
Acemagic 2025 Laptop17.3-Inch Fhd Display Laptop	$1,500
Gaming Laptop Rtx 5060 Lighter 300Hz	$1,184
Lenovo Thinkbook 16 G7 16" Fhd+	$999

The most bought item

```
In [32]: most_bought = testing[["title","total_buy"]].sort_values(by="total_buy",
         ax = most_bought.plot(
             kind="barh",
             x="title",
```

```python
        y="total_buy",
        figsize=[12,8],
        color=plt.cm.tab10.colors[:len(most_bought)]
        )
for p in ax.patches:
    ax.text(
    p.get_width() + (p.get_width() * 0.01),
    p.get_y() + p.get_height() / 2,
    (lambda x: f"{x/1_000_000:.1f}M".replace(".", ",") if x >= 1_000_000
     else f"{x/1_000:.1f}K".replace(".", ",") if x >= 1_000
     else str(int(x)))(p.get_width()),
    va="center"
)

plt.title("Most Bought Item", fontsize=20, weight="bold")
plt.xlabel("Total Buy")
plt.ylabel("Laptop Title")
ax.invert_yaxis()
plt.show()
```
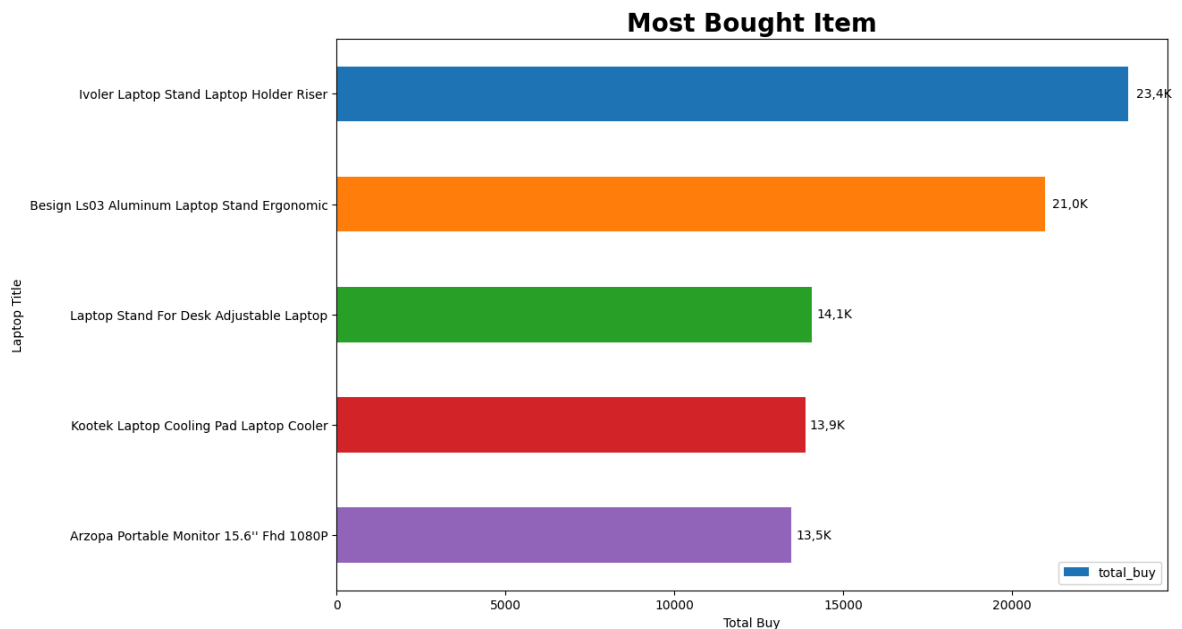

We're noticing here, that not all item is specific laptop

So the most worth it item by highest rating, highest bought, and lowest price is

```python
In [33]:  worth_it_item = testing.sort_values(by=["rating", "total_buy", "price"],
          ax = worth_it_item.plot(
              kind="barh",
              x="title",
              y=["price","total_buy", "rating"],
              figsize=(12,8),
              color=plt.cm.tab10.colors[:len(worth_it_item)]
          )
          for p in ax.patches:
              ax.text(
              p.get_width() + (p.get_width() * 0.01),
              p.get_y() + p.get_height() / 2,
              str(int(p.get_width())),
              va="center"
          )
```

```
plt.title("Most Worth It Item", fontsize=20, weight="bold")
plt.xlabel("Value")
plt.ylabel("Laptop Title")
ax.invert_yaxis()
plt.show()
```


Most Worth It Item