Welcome to our project overview, where I explore the advanced capabilities of artificial intelligence in web scraping, specifically focusing on extracting data from Amazon's web pages. My AI-driven solution is designed to streamline the process of data collection by directly processing HTML files and converting them into structured CSV formats. This method not only enhances efficiency but also significantly reduces the time and effort required for data cleaning and analysis. By automating the extraction and transformation processes, my tool empowers users to quickly gather and analyze vast amounts of product data, customer reviews, and pricing information, thereby enabling more informed decision-making and insightful market analysis. Importantly, no coding knowledge is required to utilize this tool, although having some understanding of code can be beneficial. For those interested in the technical details, the code generated by the AI is available for review and copying at the end of this document. This project not only demonstrates the practical applications of AI in web scraping but also sets the stage for future advancements in automated data processing technologies.

Webscrapping Amazon web page

We'll be web scraping Amazon's web page to gather detailed information about headphones. By extracting data on product specifications, prices, customer reviews, and ratings, we aim to build a comprehensive dataset. This dataset will provide valuable insights into market trends, consumer preferences, and competitive pricing. Our ultimate goal is to leverage this information to make informed business decisions, optimize product offerings, and enhance our marketing strategies in the headphones segment.

1. Uploading HTML files and merge them.

- We have decided to build this solution as creating a bot to iterate through Amazon web page is more complex and there are legal considerations such as ToS, intellectual property rights or data privacy laws that have to be handle in order no to have issues with the company. For this project is enough as we are going to analize 10 different brands and around 150 items.

Prompt n1: 'Merge this HTML files'

We uploaded six different html files from amazon web page for headphones search.

The brands that we filter for where: 'Xiaomi', 'Sony', 'JBL', 'HUAWEI', 'Philips', 'JVC', 'Apple', 'Beats by Dr. Dre', 'Energy Sistem', 'LG'

Answer n1: 'The HTML files have been successfully merged. You can download the merged file from the link below: Download the merged HTML file'

Prompt n2: (screenshot)

I used the screenshot of the prompt as there markdown recognises html language and shows it as what we are able to read as humans.

Basically what we did is to copy and paste what parts of the html code we wanted ChatGPT to web scrape.

In our case we are going to work with:

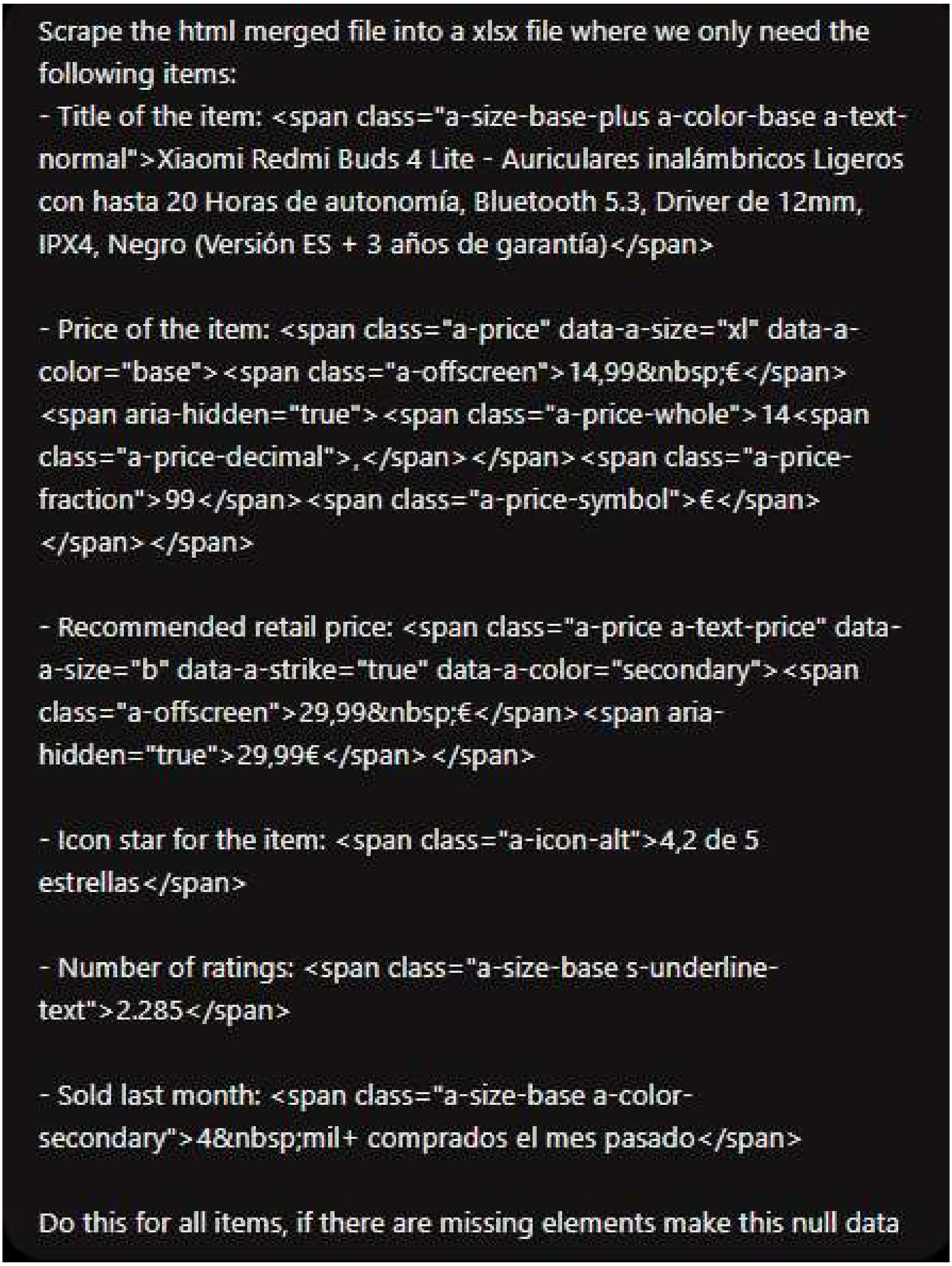
- Product description

- Price
- Recommended Retail price
- Star rating
- Total number of ratings
- Total units sold last month

Answer n2: 'The required data has been scraped and saved into an Excel file. You can download it from the link below: Download the scraped Excel file '

```
# This code for every screenshot is not from AI. It is created in
order to be able to see the screenshots generated from ChatGPT-4 and
be able to download the file as pdf.

html_prompt = Image.open('HTML_promtp.png')
display(html_prompt)
```



Scrape the html merged file into a xlsx file where we only need the following items:
- Title of the item: <span class="a-size-base-plus a-color-base a-text-normal">Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos Ligeros con hasta 20 Horas de autonomía, Bluetooth 5.3, Driver de 12mm, IPX4, Negro (Versión ES + 3 años de garantía)</span>

- Price of the item: <span class="a-price" data-a-size="xl" data-a-color="base"><span class="a-offscreen">14,99 €</span><span aria-hidden="true"><span class="a-price-whole">14<span class="a-price-decimal">,</span></span><span class="a-price-fraction">99</span><span class="a-price-symbol">€</span></span></span>

- Recommended retail price: <span class="a-price a-text-price" data-a-size="b" data-a-strike="true" data-a-color="secondary"><span class="a-offscreen">29,99 €</span><span aria-hidden="true">29,99€</span></span>

- Icon star for the item: <span class="a-icon-alt">4,2 de 5 estrellas</span>

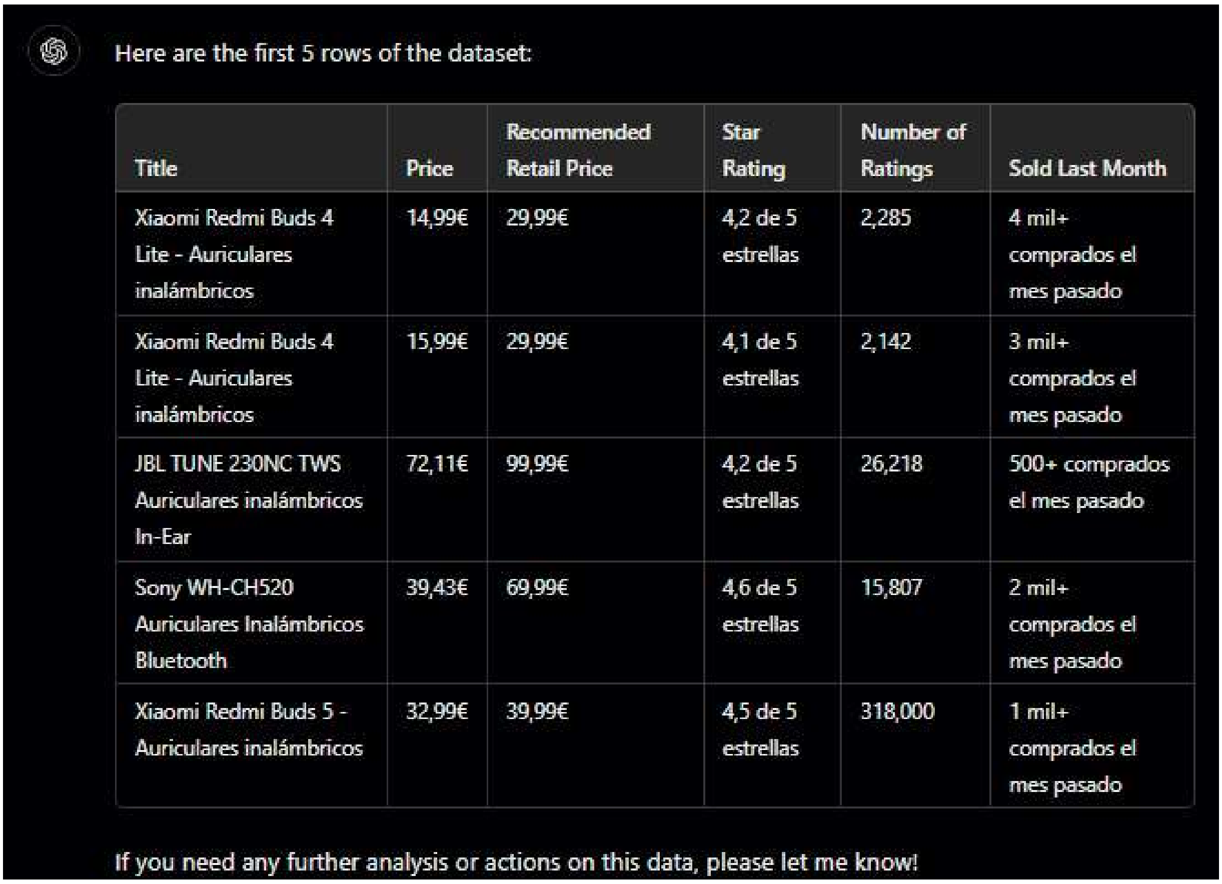
- Number of ratings: <span class="a-size-base s-underline-text">2.285</span>

- Sold last month: <span class="a-size-base a-color-secondary">4 mil+ comprados el mes pasado</span>

Do this for all items, if there are missing elements make this null data

An Exce file was generated with the information required as a table.

```
excel = Image.open('Excel.png')
display(excel)
```



Here are the first 5 rows of the dataset:

Title	Price	Recommended Retail Price	Star Rating	Number of Ratings	Sold Last Month
Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos	14,99€	29,99€	4,2 de 5 estrellas	2,285	4 mil+ comprados el mes pasado
Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos	15,99€	29,99€	4,1 de 5 estrellas	2,142	3 mil+ comprados el mes pasado
JBL TUNE 230NC TWS Auriculares inalámbricos In-Ear	72,11€	99,99€	4,2 de 5 estrellas	26,218	500+ comprados el mes pasado
Sony WH-CH520 Auriculares Inalámbricos Bluetooth	39,43€	69,99€	4,6 de 5 estrellas	15,807	2 mil+ comprados el mes pasado
Xiaomi Redmi Buds 5 - Auriculares inalámbricos	32,99€	39,99€	4,5 de 5 estrellas	318,000	1 mil+ comprados el mes pasado

If you need any further analysis or actions on this data, please let me know!

Cleaning and preparing the data

Prompt n3:

We are going to clean and prepare the data. Do the following tasks:

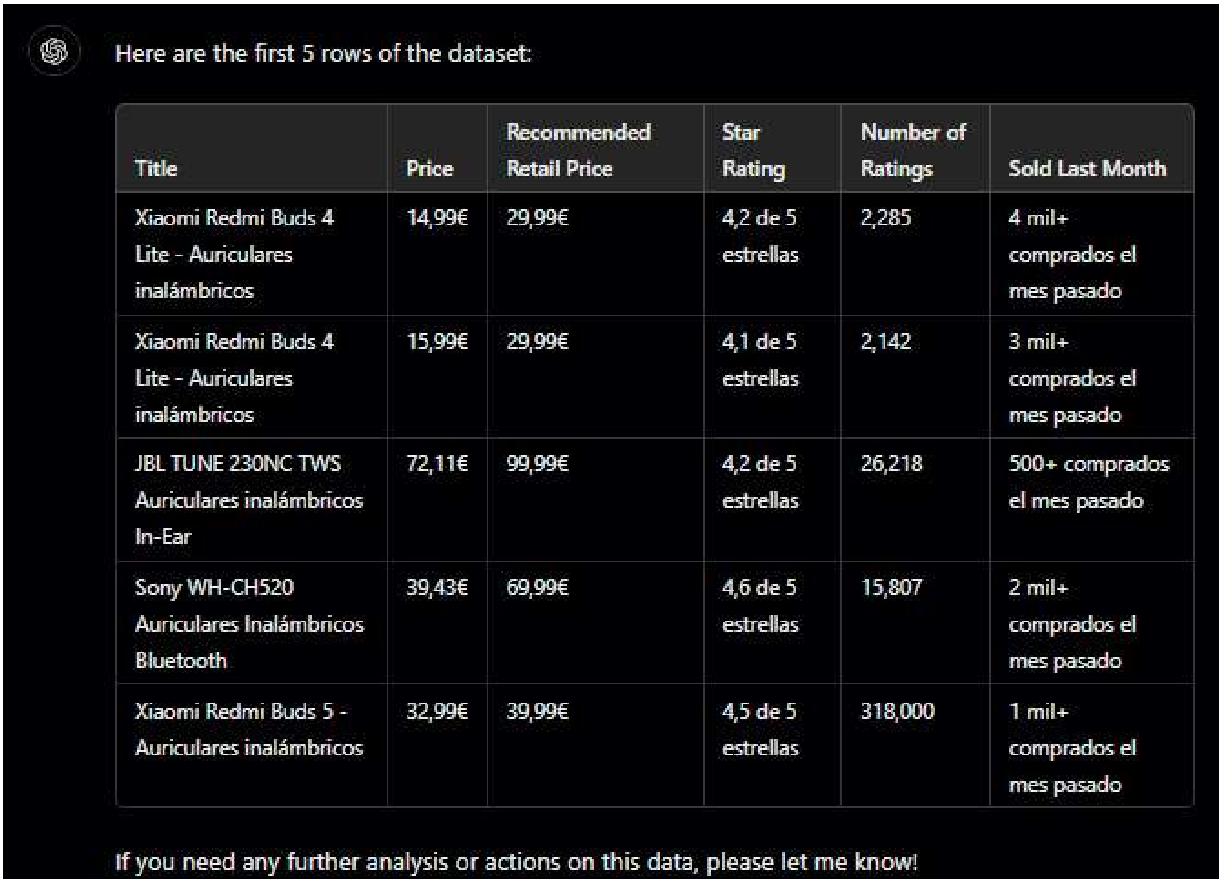
- Replace column name 'Title' with 'product_description'
- Replace column name 'Price' with 'price'
- Replace column name 'Recommended Retail Price' with 'recommended_retail_price'
- Replace column name 'Star Rating' with 'star_rating'
- Replace column name 'Number of Ratings' with 'number_of_ratings'
- Replace column name 'Sold Last Month' with 'units_sold_last_moth'
- From column 'price', delete currency symbol €.
- Convert column 'price' to float.
- From column 'recommended_retail_price', delete currency symbol €.
- Convert column ''recommended_retail_price' to float.
- From column 'star_rating' delete the string part 'de 5 estrellas'. As an example for row number 1 the result should be 4,2

- Convert column 'star_rating' to float.
- From column 'units_sold_last_moth' delete the string 'comprados el mes pasado'.

Finally show a the first rows of the cleaned data set

Answer n3: 'The data has been cleaned and prepared as requested. Here is the first few rows of the cleaned dataset. If you need any further modifications or analysis, please let me know! '

```
Cleaned_data = Image.open('Cleaned_data.png')
display(excel)
```



Here are the first 5 rows of the dataset:

Title	Price	Recommended Retail Price	Star Rating	Number of Ratings	Sold Last Month
Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos	14,99€	29,99€	4,2 de 5 estrellas	2,285	4 mil+ comprados el mes pasado
Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos	15,99€	29,99€	4,1 de 5 estrellas	2,142	3 mil+ comprados el mes pasado
JBL TUNE 230NC TWS Auriculares inalámbricos In-Ear	72,11€	99,99€	4,2 de 5 estrellas	26,218	500+ comprados el mes pasado
Sony WH-CH520 Auriculares Inalámbricos Bluetooth	39,43€	69,99€	4,6 de 5 estrellas	15,807	2 mil+ comprados el mes pasado
Xiaomi Redmi Buds 5 - Auriculares inalámbricos	32,99€	39,99€	4,5 de 5 estrellas	318,000	1 mil+ comprados el mes pasado

If you need any further analysis or actions on this data, please let me know!

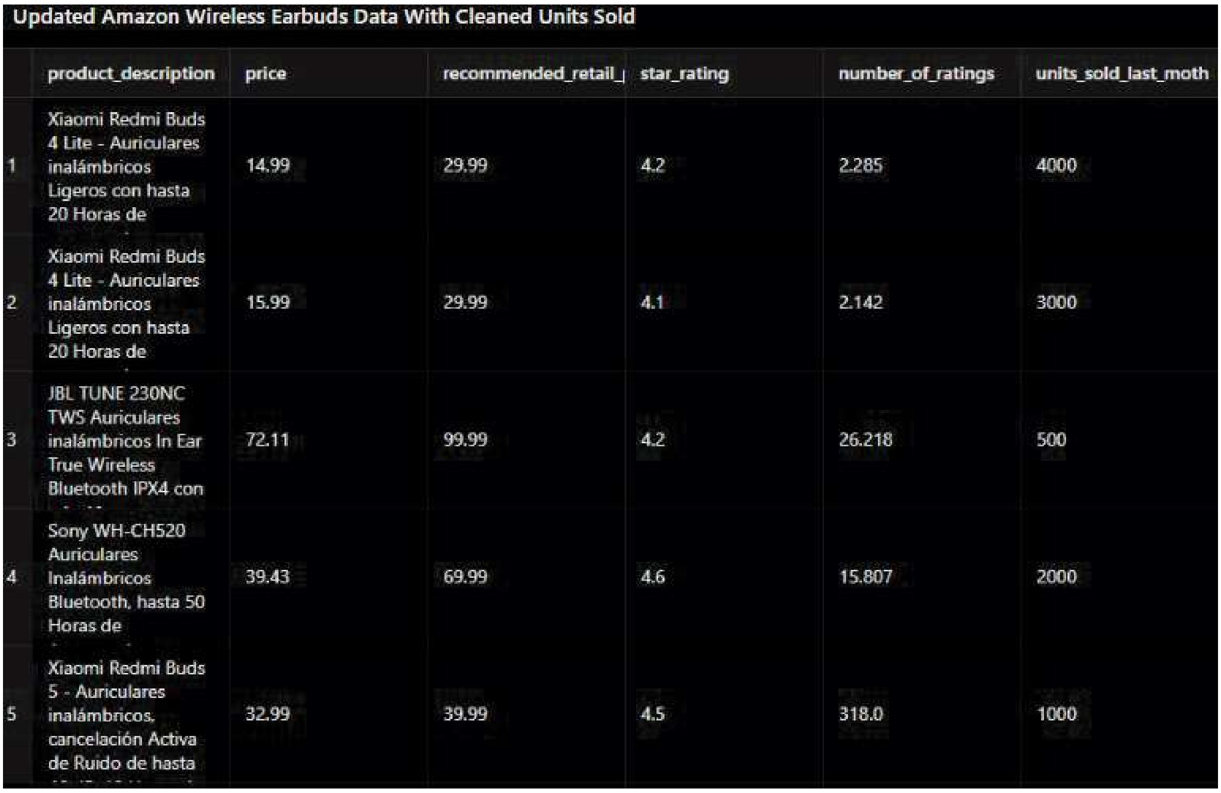
Prompt n4:

There are different conditions in column 'units_sold_last_month'. To solve the issue i want you to build an if condition that iterates through the column values and do the following:

- For those strings that have the following structure '4 mil +' convert it to 4000
- For those strings that have the following structure '500+' convert it to 500
- For those strings that have the following structure '50+' convert it to 50
- For the rest, keep it blank

Answer n4: 'The units sold data has been cleaned and the updated dataset is ready. You can download the updated Excel file using the link below: Download the updated dataset '

```
Units_sold_cleaned = Image.open('Units_sold_cleaned.png')
display(Units_sold_cleaned)
```



Updated Amazon Wireless Earbuds Data With Cleaned Units Sold

	product_description	price	recommended_retail_	star_rating	number_of_ratings	units_sold_last_moth
1	Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos Ligeros con hasta 20 Horas de	14.99	29.99	4.2	2.285	4000
2	Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos Ligeros con hasta 20 Horas de	15.99	29.99	4.1	2.142	3000
3	JBL TUNE 230NC TWS Auriculares inalámbricos In Ear True Wireless Bluetooth IPX4 con	72.11	99.99	4.2	26.218	500
4	Sony WH-CH520 Auriculares Inalámbricos Bluetooth, hasta 50 Horas de	39.43	69.99	4.6	15.807	2000
5	Xiaomi Redmi Buds 5 - Auriculares inalámbricos, cancelación Activa de Ruido de hasta	32.99	39.99	4.5	318.0	1000

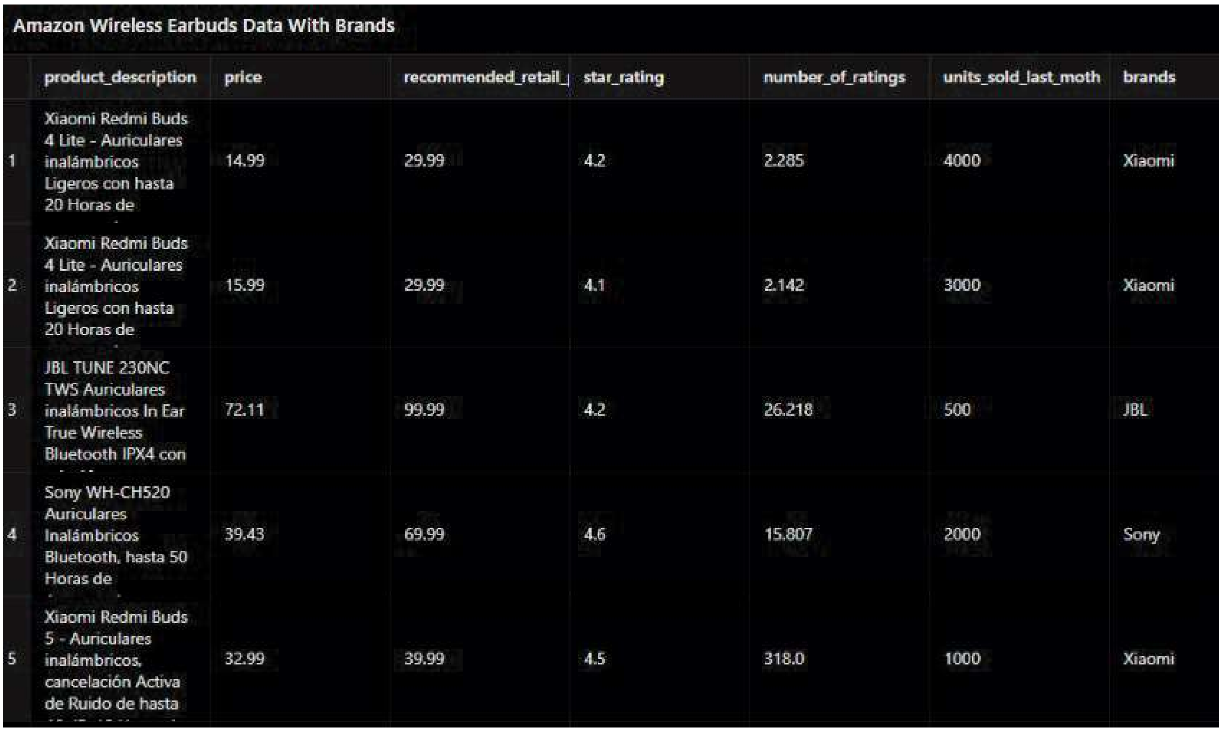
Prompt n5:

There were 10 brands selected: 'Xiaomi', 'Sony', 'JBL', 'HUAWEI', 'Philips', 'JVC', 'Apple', 'Beats by Dr. Dre', 'Energy Sistem', 'LG'

From column product_description scan the strings for those brands and create a new column named 'brands' with the brand found.

Answer n5: 'The dataset has been updated with a new column 'brands' that identifies the brand found in the product descriptions. You can download the updated Excel file using the link below: Download the updated dataset with brands'

```
Brands = Image.open('Brands.png')
display(Brands)
```



Amazon Wireless Earbuds Data With Brands

	product_description	price	recommended_retail_	star_rating	number_of_ratings	units_sold_last_moth	brands
1	Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos Ligeros con hasta 20 Horas de	14.99	29.99	4.2	2.285	4000	Xiaomi
2	Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos Ligeros con hasta 20 Horas de	15.99	29.99	4.1	2.142	3000	Xiaomi
3	JBL TUNE 230NC TWS Auriculares inalámbricos In Ear True Wireless Bluetooth IPX4 con	72.11	99.99	4.2	26.218	500	JBL
4	Sony WH-CH520 Auriculares Inalámbricos Bluetooth, hasta 50 Horas de	39.43	69.99	4.6	15.807	2000	Sony
5	Xiaomi Redmi Buds 5 - Auriculares inalámbricos, cancelación Activa de Ruido de hasta	32.99	39.99	4.5	318.0	1000	Xiaomi

Prompt n6:

Column product_description contains the description built by the seller of the headphones. I want to create a classification of how the descriptions can make a customer be interested in the product. In order to do this i want to create five new columns. Four of them are based on characteristics from the product description and the fifth is a sum of the results of the other four. The first four columns would have a yes/no anwser based on the result of the rules they should follow.

Do the following tasks:

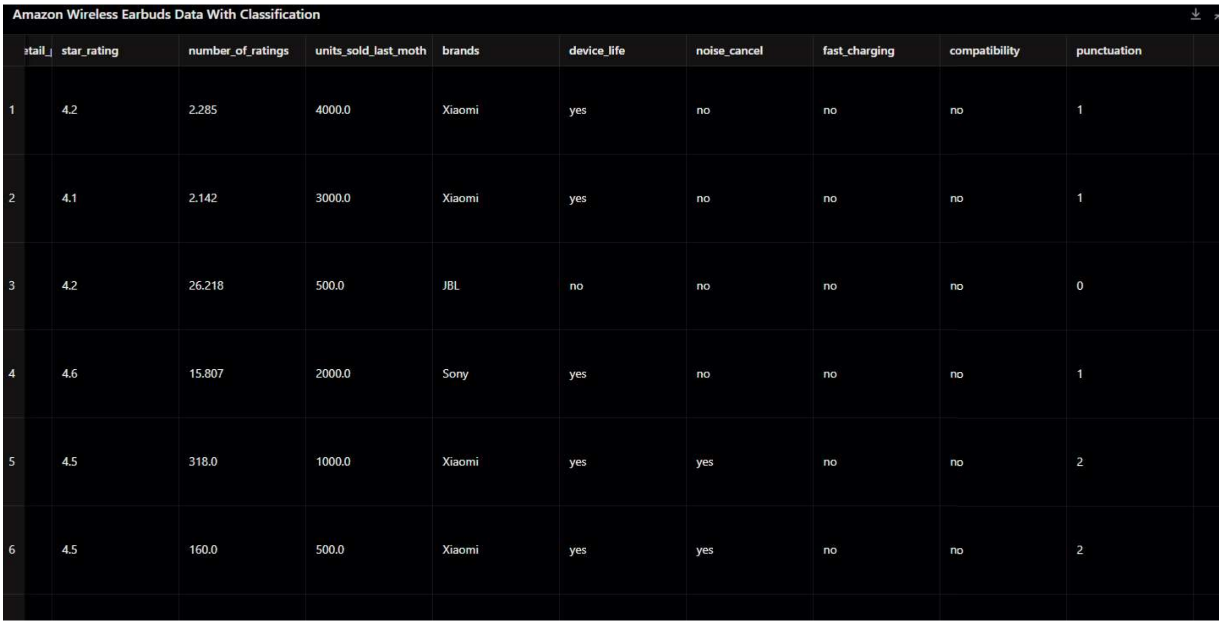
- Create four new columns named 'device_life', 'noise_cancel', 'fast_charging' and 'compatibility'
- For column 'device_life' insert yes if you find words like 'battery', 'autonomy', 'bateria', 'autonomia', 'horas', 'hours'. If not such words are found, insert no.
- For column 'noise_cancel' insert yes if you find words like 'cancelacion', 'ruido', 'noise', 'cancellation', 'supresion', 'supression'. If not such words are found insert no.
- For column 'fast_charging' insert yes if you find words like 'carga rapida' , 'fast charging'. If not such words are found insert no.
- For column 'compatilibilty' insert yes if you find words like 'compatibles', 'compatible'. If not such words are found insert no.
- Finally create a new column named 'punctuation'. Here sum all the yes from each row and insert the results.

Answer n6:

I have added the following columns to the dataset:

device_life: Indicates 'yes' if the product description mentions terms related to battery life, otherwise 'no'. noise_cancel: Indicates 'yes' if the product description mentions terms related to noise cancellation, otherwise 'no'. fast_charging: Indicates 'yes' if the product description mentions terms related to fast charging, otherwise 'no'. compatibility: Indicates 'yes' if the product description mentions terms related to compatibility, otherwise 'no'. punctuation: Sum of 'yes' values from the four new columns. The updated dataset with these new columns is now available for further analysis. If you need any specific insights or visualizations, please let me know! '

```
Punctuation = Image.open('Punctuation.png')
display(Punctuation)
```



Amazon Wireless Earbuds Data With Classification

	etail_	star_rating	number_of_ratings	units_sold_last_moth	brands	device_life	noise_cancel	fast_charging	compatibility	punctuation
1		4.2	2.285	4000.0	Xiaomi	yes	no	no	no	1
2		4.1	2.142	3000.0	Xiaomi	yes	no	no	no	1
3		4.2	26.218	500.0	JBL	no	no	no	no	0
4		4.6	15.807	2000.0	Sony	yes	no	no	no	1
5		4.5	318.0	1000.0	Xiaomi	yes	yes	no	no	2
6		4.5	160.0	500.0	Xiaomi	yes	yes	no	no	2

Prompt n7:

Create a new column called 'product_id'. This task has two objectives:

- Give a name to each row so we can identify each product without having such a long string as the column product_description.
- Make sure that there are no duplicates.

Based on the objectives to cover, scan each string from product_description and give them a unique name based on the columns brand and product_description. For example: product_description = 'Xiaomi Redmi Buds 4 Lite - Auriculares inalámbricos Ligeros con hasta 20 Horas de autonomía, Bluetooth 5.3, Driver de 12mm, IPX4, Negro (Versión ES + 3 años de garantía)' brand = 'Xiaomi' product_id = 'Xiami_1'

Scan the product_description for exact duplicates. If there are, just keep one. Show the number of duplicates even if there are zero. Show the total number of rows before and after the duplicate scan.

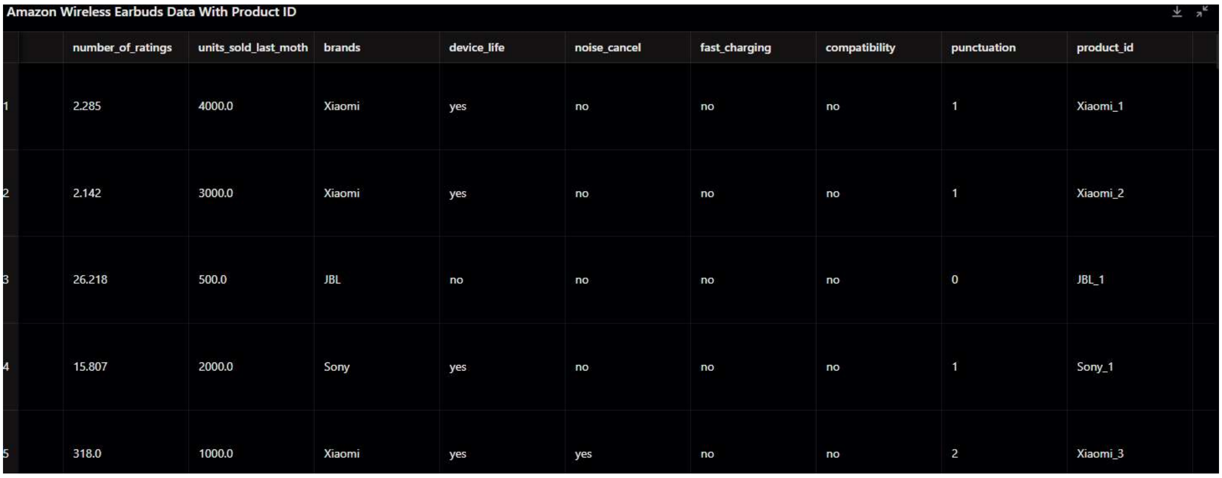
Answer n7:

The dataset has been processed as follows:

- Total number of rows before removing duplicates: 143
- Total number of rows after removing duplicates: 143
- Number of duplicates found and removed: 0

A new column, product_id, has been created to uniquely identify each product based on the brand and a sequential number. If you need any further modifications or analyses, please let me know!

```
Product_id = Image.open('Product_id.png')
display(Product_id)
```



Amazon Wireless Earbuds Data With Product ID

		number_of_ratings	units_sold_last_moth	brands	device_life	noise_cancel	fast_charging	compatibility	punctuation	product_id
1		2.285	4000.0	Xiaomi	yes	no	no	no	1	Xiaomi_1
2		2.142	3000.0	Xiaomi	yes	no	no	no	1	Xiaomi_2
3		26.218	500.0	JBL	no	no	no	no	0	JBL_1
4		15.807	2000.0	Sony	yes	no	no	no	1	Sony_1
5		318.0	1000.0	Xiaomi	yes	yes	no	no	2	Xiaomi_3

We saw some errors with brands. We are going to solve them

Prompt n8:

For the following product_id, rename the brand to:

- Other_1 = Xiaomi
- Other_2 = Philips
- Other_3 = Philips
- Other_4 = Beats by Dr. Dre
- Other_5 = Philips
- Other_6 = Philips
- Other_7 = Sony

For the product_description that begin with 'Beats' assign the brand to Beats by Dr. Dre and change the product_id to its correspondent id

Rename the brand to its associated product_ID

Answer n8:

The updates have been made according to your specifications. Here are the changes applied:

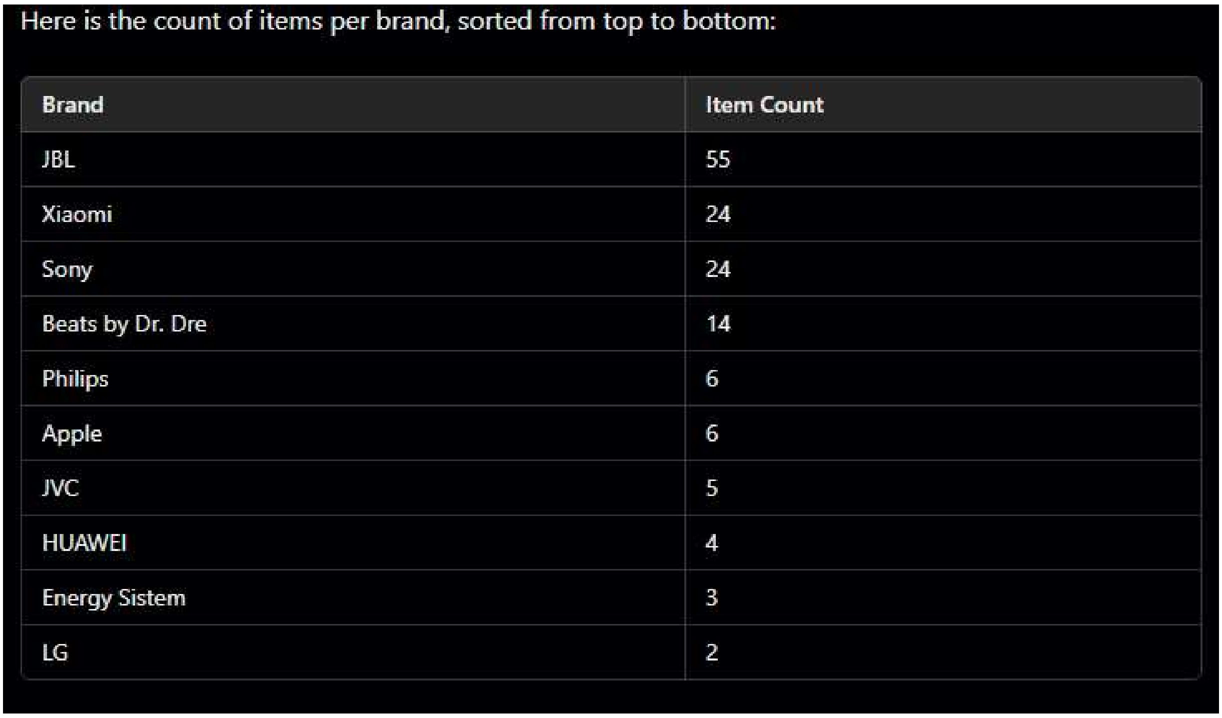
For specific product_ids, the brand names have been updated. For product descriptions starting with 'Beats', the brand has been set to 'Beats by Dr. Dre' and the product_id has been updated accordingly. The product_id has been reassigned based on the updated brand names to ensure uniqueness.

Prompt n9:

Count the number of items per brand and show them from top to bottom

Answer n9:

```
Brands_count = Image.open('Brands_count.png')
display(Brands_count)
```



Here is the count of items per brand, sorted from top to bottom:

Brand	Item Count
JBL	55
Xiaomi	24
Sony	24
Beats by Dr. Dre	14
Philips	6
Apple	6
JVC	5
HUAWEI	4
Energy Sistem	3
LG	2

As the aim of the project is to predict the sales per brand and there are brand that have low number of items. We have decided to used the brands with more than 1o items.

Prompt n10:

Create a new data frame where only brand with more that 10 items are considered to be analyze

Answer n10:

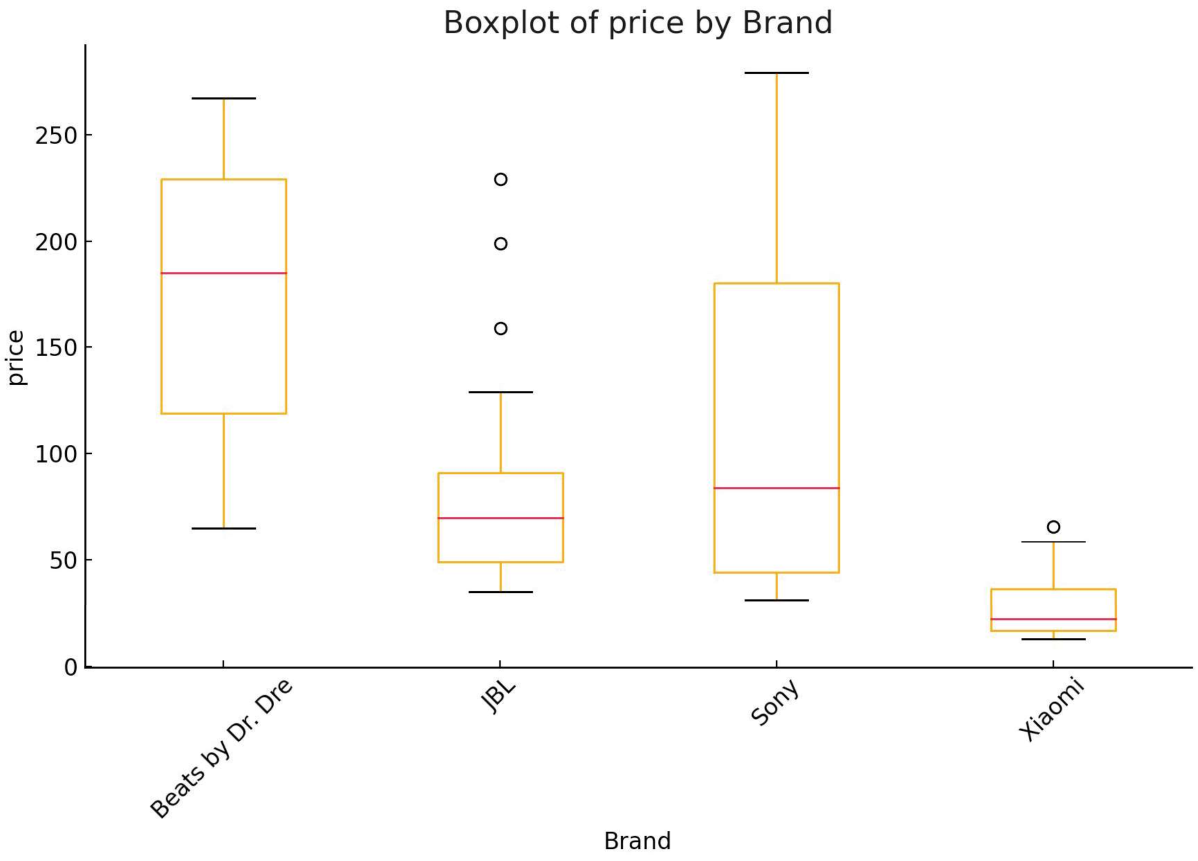
The filtered dataset has been saved. You can download it using the link below: Download the filtered file

Prompt n11:

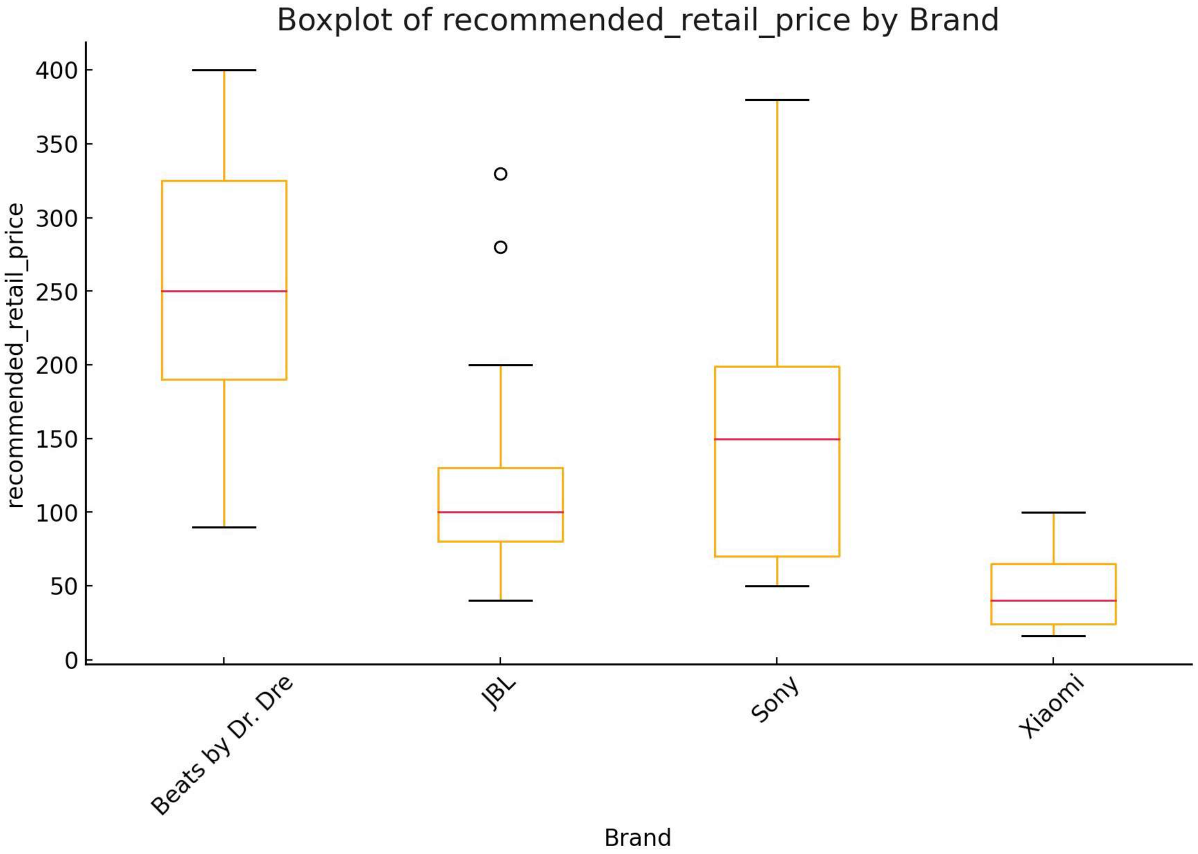
I want to fill in the missing values from columns price, recommended_retail price and units_sold_last_moth. In order to do this i have to decide wether to fill the missing values with the mean or the median. To decide i want you to create a boxplot for every brand with missing values in columns those columns. Just take into account the brand from the new data frame 'filtered_data'

Answer n10:

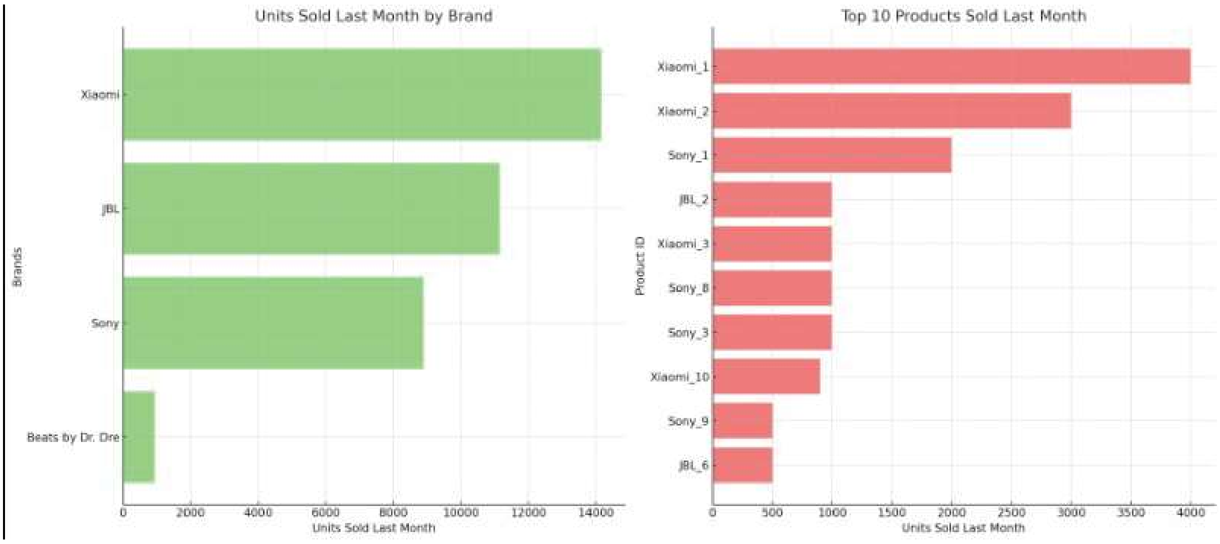
```
price = Image.open('price.png')
display(price)
```



```
recommneded_price = Image.open('recommneded_price.png')
display(recommneded_price)
```



```python
units_sold = Image.open('units_sold.png')
display(units_sold)
```



Prompt n11:

Do the following tasks:

- In column price replace the missing values with the mean in Beats by Dr. Dre and Sony. For JBL and Xiaomi replace the missing values with the median.
- In column recommended_retail_price replace the missing values with the mean in Beats by Dr. Dre, Sony and Xiaomi. For JBL replace the missing values with the median.
- In column units_sold_last_month replace the missing values with the mean in Beats by Dr. Dre. For JBL, Sony and Xiaomi replace the missing values with the median.

Answer n11:

The missing values have been filled as specified. Here are the mean and median values used for each iteration:

Price:

- Beats by Dr. Dre: Mean = 176.36
- Sony: Mean = 118.09
- JBL: Median = 70.00
- Xiaomi: Median = 22.49
- Recommended Retail Price:
- Beats by Dr. Dre: Mean = 259.12
- Sony: Mean = 175.95
- Xiaomi: Mean = 44.46
- JBL: Median = 99.99
- Units Sold Last Month:
- Beats by Dr. Dre: Mean = 66.67
- Sony: Median = 200.00
- Xiaomi: Median = 300.00
- JBL: Median = 200.00

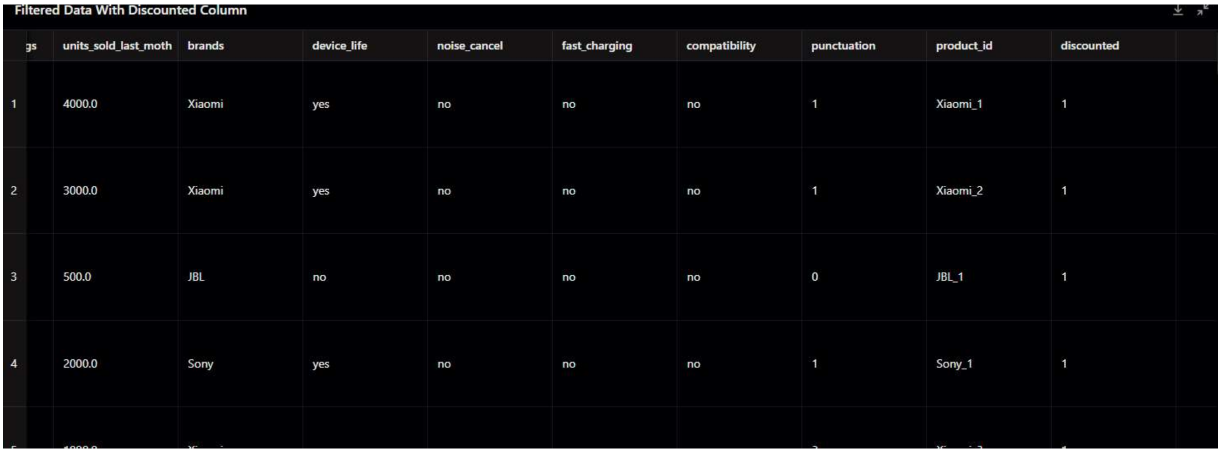
The updated data is now ready.

Prompt n11:

Create a new column called 'discounted' when values in column price are lower than values in column 'recommended_retail_price' Insert a 1 when is it true and insert 0 when false

Answer n11:

```
discounted = Image.open('discounted.png')
display(discounted)
```



Filtered Data With Discounted Column

	gs	units_sold_last_moth	brands	device_life	noise_cancel	fast_charging	compatibility	punctuation	product_id	discounted
1		4000.0	Xiaomi	yes	no	no	no	1	Xiaomi_1	1
2		3000.0	Xiaomi	yes	no	no	no	1	Xiaomi_2	1
3		500.0	JBL	no	no	no	no	0	JBL_1	1
4		2000.0	Sony	yes	no	no	no	1	Sony_1	1

We have concluded the data cleaning and preparation tasks. Now we have our data frame ready to be analyze and used for our predictive challenge.

Data Visualization

Prompt n12:

Create two pararel graphs.

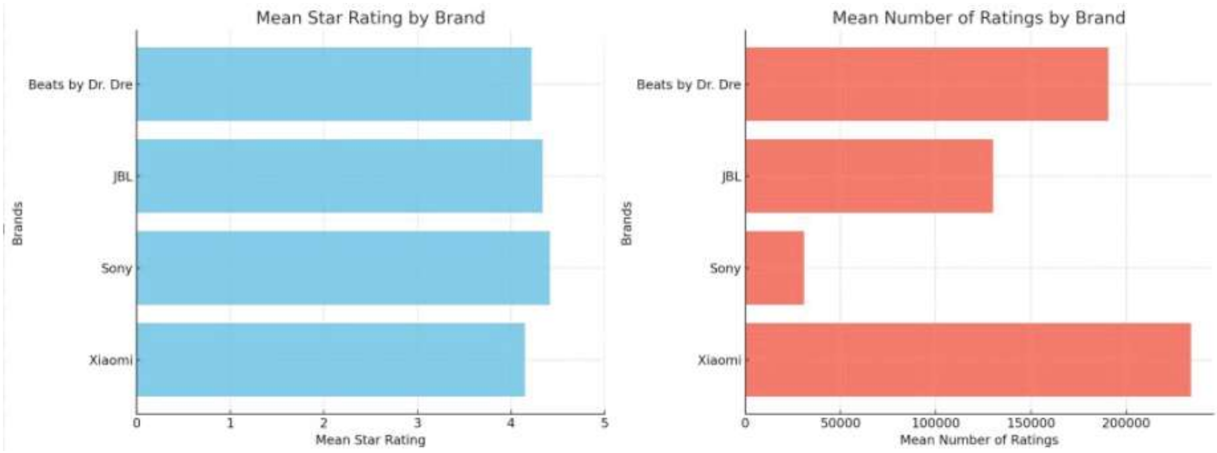
- Bar graph for each brand by the mean of the star_rating
- Bar graph for each brand by the mean of the number_of_ratings

Answer n12:

Here are the two parallel bar graphs:

The left graph shows the mean star rating for each brand. The right graph shows the mean number of ratings for each brand. Both graphs have the brands on the y-axis to facilitate comparison, and the axes are inverted to align the order of the brands across both graphs. This setup allows for a clearer comparison of the brand's performance in terms of rating and popularity.

```
bar_graphs = Image.open('bar_graphs.png')
display(bar_graphs)
```



Prompt n13:

Create two paralel graphs:

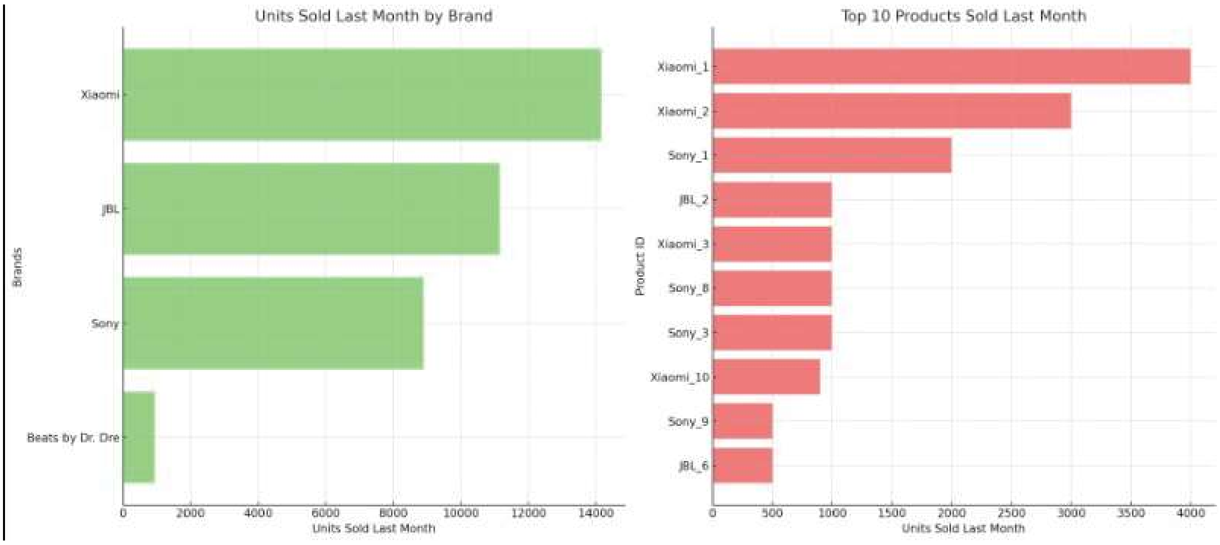
- Bar graph of units_sold_last_month by brand from top to bottom.
- Top 10 produts sold last month. Use product_id to classify the products

Answer n13:

Here are the two parallel bar graphs:

The left graph displays the total units sold last month by each brand, ranked from top to bottom. The right graph shows the top 10 products sold last month, identified by their product IDs, ranked from highest to lowest sales. These visualizations provide insights into the sales distribution among brands and individual products over the last month.

```
Units_sold = Image.open('Units_sold.png')
display(Units_sold)
```



Prompt n14:

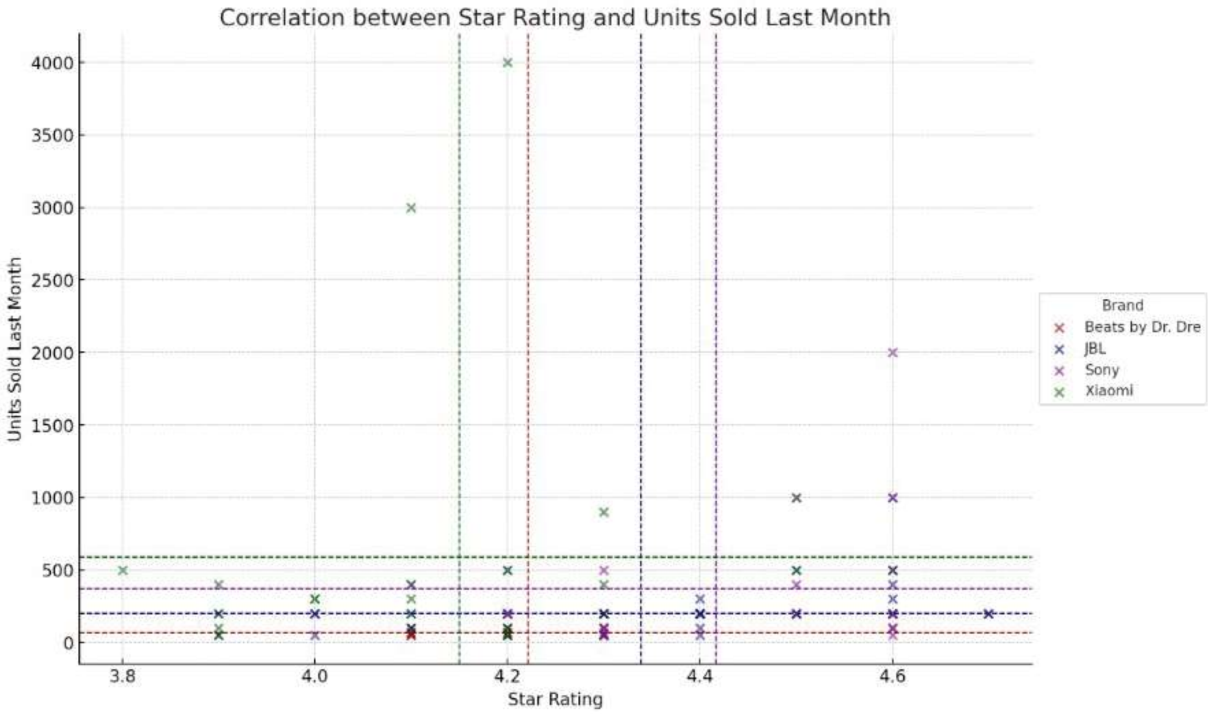
Create a visualization that represents the correlation between the star_rating and the units_sold_last_month. Build a dashed line with the mean units_sold_last_month by brand. Use the same colors for as the ones used for each brand Build a dashed line with the mean star_rating by brand. Use the same colors for as the ones used for each brand Use colors that do help the data to be seen easily. Put the legend aside from the graph

Asnwer n14:

Here's a visualization that represents the correlation between star_rating and units_sold_last_month for each brand.

The scatter plot shows individual product performance within each brand. Dashed lines represent the mean units sold last month (horizontal) and the mean star rating (vertical) for each brand, using distinct colors to differentiate between the brands. The legend is positioned to the side of the graph for clarity, and I've selected colors that make the data points and lines stand out clearly.

```
Correlation_Star_Units = Image.open('Correlation_Star_Units.png')
display(Units_sold)
```



Prompt n15:

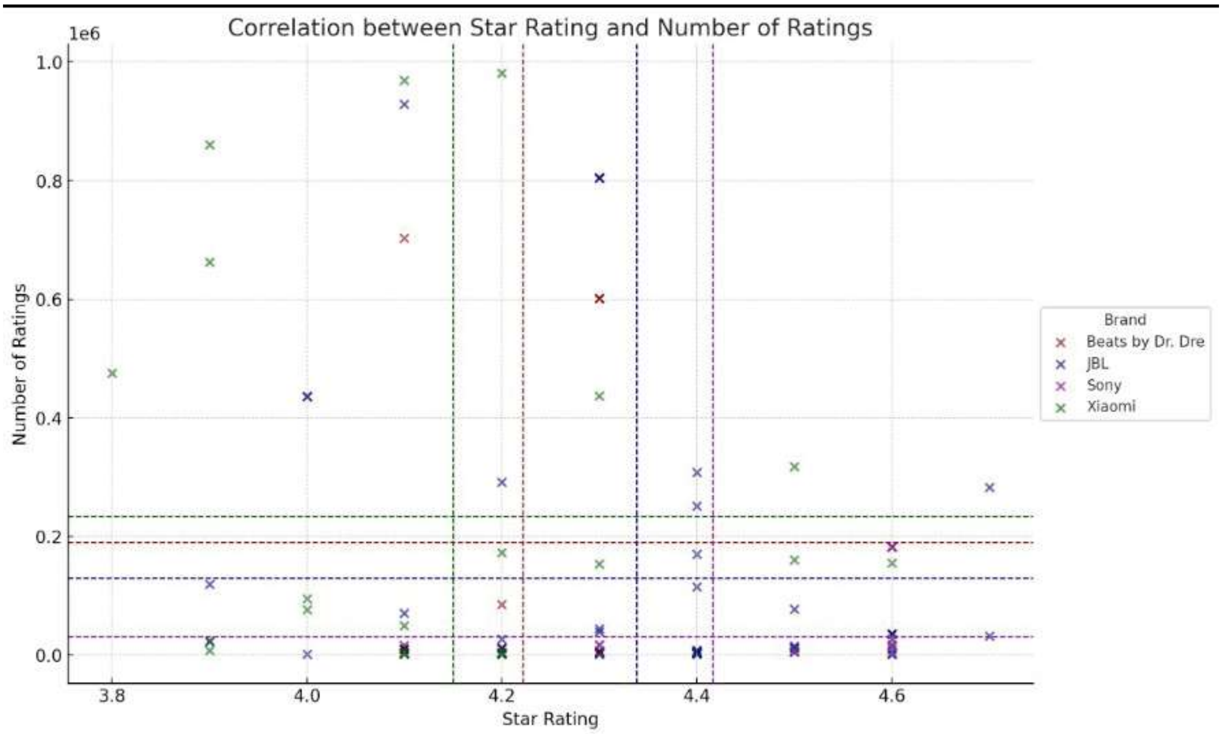
Create a visualization that represents the correlation between the star_rating and the number_of_ratings. Build a dashed line with the mean number_of_ratings by brand. Use the same colors for as the ones used for each brand Build a dashed line with the mean star_rating by brand. Use the same colors for as the ones used for each brand Use colors that do help the data to be seen easily. Put the legend aside from the graph

Answer n15:

Here's the visualization showing the correlation between star_rating and number_of_ratings:

The scatter plot details the relationship for each brand, illustrating how star ratings correlate with the volume of ratings received. Horizontal dashed lines indicate the mean number of ratings for each brand, while vertical dashed lines show the mean star ratings. The colors used are distinct for each brand to enhance visibility, and the legend is placed to the side to keep the plot area clear.

```
Star_rating_numbers = Image.open('Star_rating_numbers.png')
display(Star_rating_numbers)
```



Prompt n16:

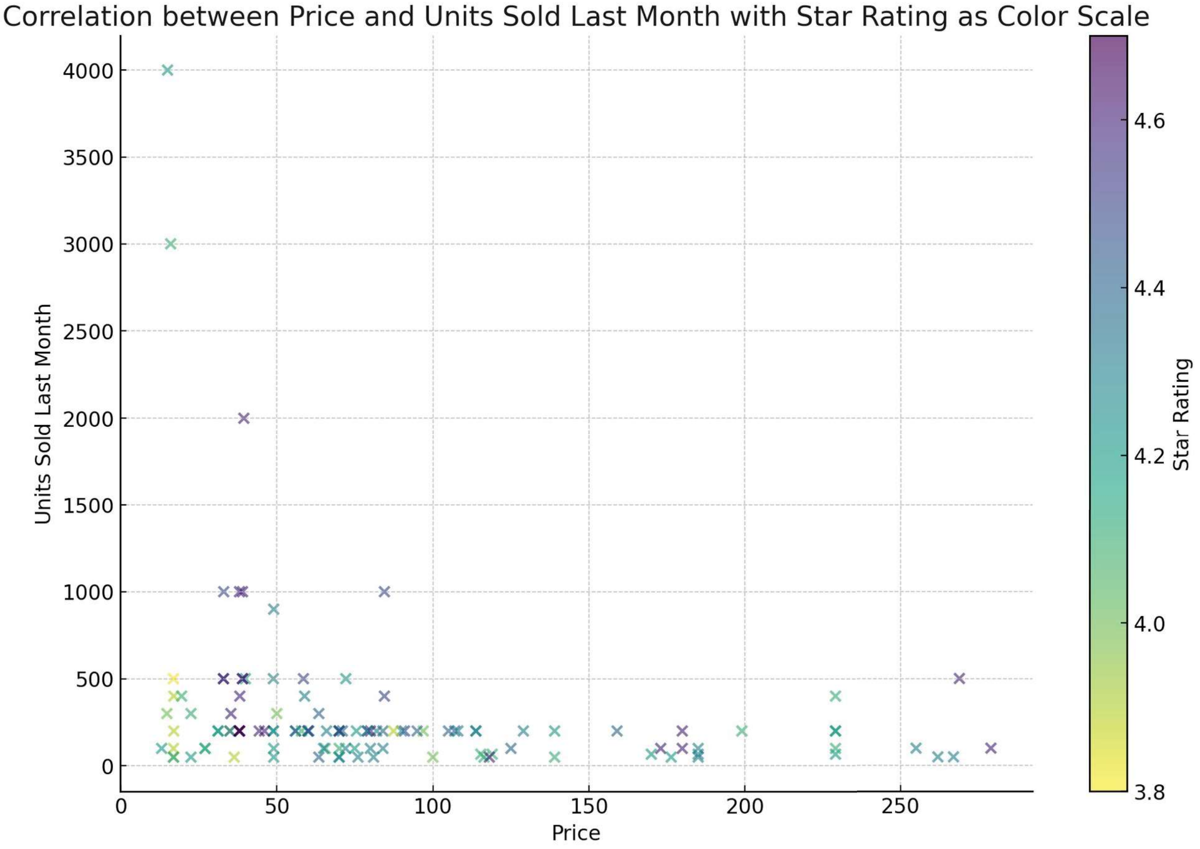
Create a visualization that represents the correlation between the price and the units_sold_last_month considering the star_rating as an input for the correlation with a color scale. The color from the color scale should be darker for the better rated items and ligther for the worse rated.

Answer n16:

Here's the visualization representing the correlation between price and units_sold_last_month, with star_rating influencing the color scale:

Scatter Plot: Each point represents a product, with its position determined by its price and units sold last month. Color Scale: Darker colors indicate higher star ratings, while lighter colors indicate lower star ratings, using a viridis colormap for clear differentiation. This visualization helps to see how price and units sold relate while also considering the impact of star ratings.

```
sales_star_rating = Image.open('sales_star_rating.png')
display(sales_star_rating)
```



Prompt n17:

Execute the same visualization but separate it by brand. Build a graph for each brand and show it separated Create a grey background so data points can be seen easily

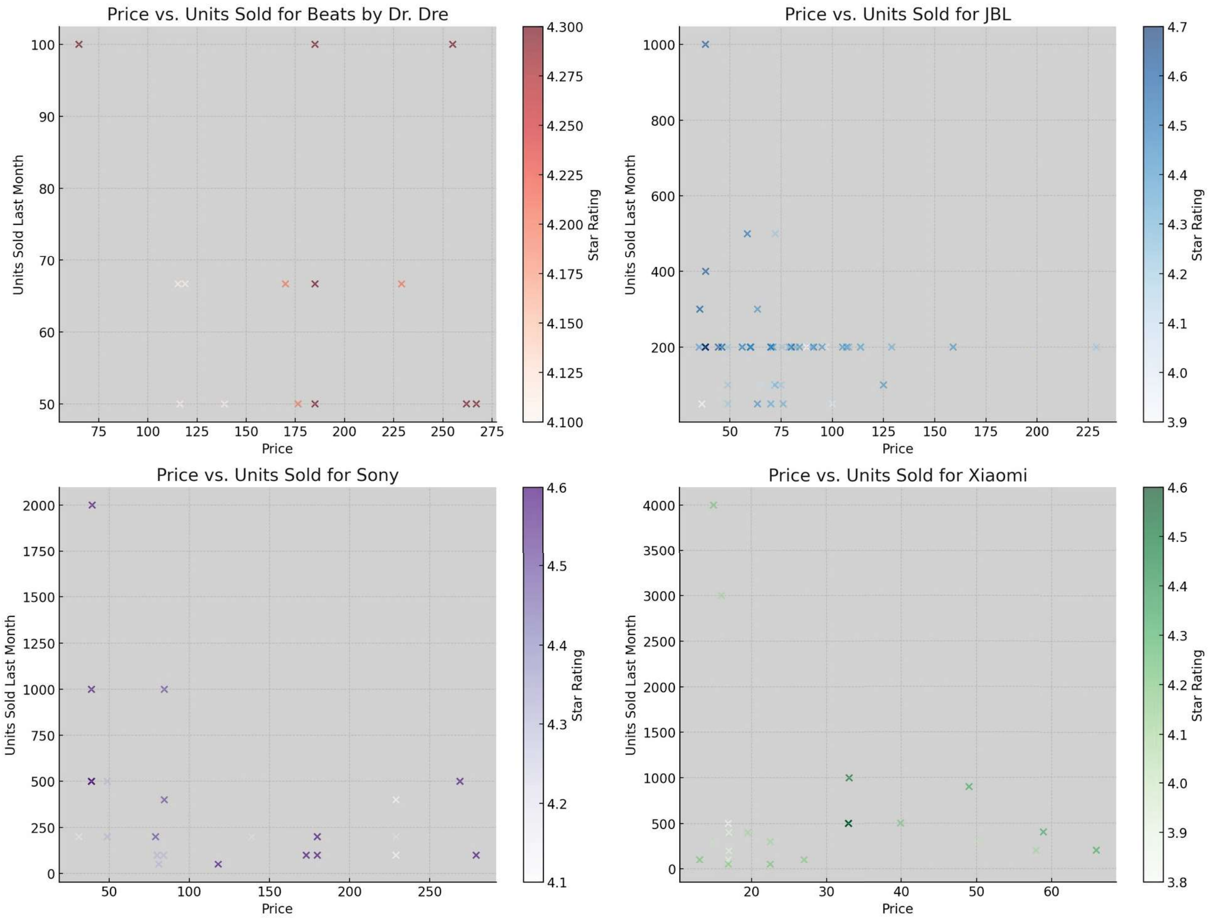
Answer n17:

Here are the separate visualizations for each brand, showing the correlation between price and units_sold_last_month, with star_rating as the color scale:

- JBL (Blues colormap)
- Beats by Dr. Dre (Reds colormap)
- Xiaomi (Greens colormap)
- Sony (Purples colormap)

Each subplot provides a detailed view of how the products within each brand perform in terms of sales and ratings.

```python
Sold_rating_brands = Image.open('Sold_rating_brands.png')
display(Sold_rating_brands)
```



After building some visuals to get a better knowledge of the dataset we can even ask for a summary of the insights. As you can see in the next prompt, all the insigths given by the IA are really accurate and make sense.

Prompt n18

Create a summary of the insigths you find from all the graphs

Answer n18

Summary of Insights:

- Units Sold Last Month:
  - Xiaomi dominates in terms of units sold, indicating a strong market presence and popularity.

  - Sony and JBL also show substantial sales figures, but not as high as Xiaomi.
  - Beats by Dr. Dre shows lower sales compared to the other brands, suggesting it might have a niche but dedicated customer base.
- Top 10 Products Sold Last Month:
  - Xiaomi products feature prominently in the top 10, further underscoring their popularity and market reach.
  - The sales figures highlight the dominance of a few key products in driving overall brand performance.
- Correlation between Star Rating and Units Sold Last Month:
  - A positive correlation is visible; products with higher star ratings tend to sell more units.
  - Dashed lines representing brand averages help identify which brands exceed or fall below their average performance.
- Correlation between Star Rating and Number of Ratings:
  - Products with higher star ratings generally have more reviews, suggesting that well-reviewed products attract more customer feedback.
  - Dashed lines again provide a benchmark, showing the average performance for each brand.
- Correlation between Price and Units Sold Last Month (Considering Star Rating):
  - Higher-priced items tend to have varying units sold, indicating that while price is a factor, other aspects like brand reputation and product quality (star rating) also play a significant role.
  - Products with higher star ratings (darker colors) show better sales performance across different price points, emphasizing the importance of quality and customer satisfaction.
- Brand-Specific Insights from Correlation Graphs:
  - JBL: Higher-rated JBL products show a strong correlation with better sales, indicating customer trust in quality.
  - Beats by Dr. Dre: High ratings correlate well with units sold, but there are fewer products, hinting at a premium niche market.
  - Xiaomi: A broad range of products across various price points, with higher-rated items performing exceptionally well in sales.

- Sony: Consistent performance across products, with higher-rated items showing better sales figures.

Conclusion:

- Xiaomi emerges as a dominant player in terms of sales volume, supported by a broad product range and competitive pricing.
- JBL and Beats by Dr. Dre exhibit strong brand loyalty and quality perception, translating to good sales performance.
- Sony maintains a balanced performance, suggesting a reliable but not market-leading position.

These insights provide a comprehensive understanding of the market dynamics, highlighting how star ratings, number of ratings, price, and brand reputation interplay to influence sales performance.

```
#Code answer for prompt n1


from bs4 import BeautifulSoup

# Load the HTML content of each file
file_paths = [
    "/mnt/data/Amazon.es_ Auriculares Inalambricos - Marca_ 10 Seleccionados_ Electrónica_1.html",
    "/mnt/data/Amazon.es_ Auriculares Inalambricos - Marca_ 10 Seleccionados_ Electrónica_2.html",
    "/mnt/data/Amazon.es_ Auriculares Inalambricos - Marca_ 10 Seleccionados_ Electrónica_3.html",
    "/mnt/data/Amazon.es_ Auriculares Inalambricos - Marca_ 10 Seleccionados_ Electrónica_4.html",
    "/mnt/data/Amazon.es_ Auriculares Inalambricos - Marca_ 10 Seleccionados_ Electrónica_5.html",
    "/mnt/data/Amazon.es_ Auriculares Inalambricos - Marca_ 10 Seleccionados_ Electrónica_6.html"
]

soup_objects = []

for path in file_paths:
    with open(path, 'r', encoding='utf-8') as file:
        content = file.read()
        soup = BeautifulSoup(content, 'html.parser')
        soup_objects.append(soup)

# Create a new BeautifulSoup object for the merged HTML
merged_soup = BeautifulSoup('<!DOCTYPE html><html lang="es"></html>', 'html.parser')
```

```python
merged_html = merged_soup.html

# Merge the contents of all soup objects
for soup in soup_objects:
    body_content = soup.find('body')
    if body_content:
        merged_html.append(body_content)

# Output the merged HTML content
merged_html_content = merged_soup.prettify()

# Save the merged content to a new file
merged_file_path = "/mnt/data/Amazon.es_Auriculares_Inalambricos_Marca_10_Seleccionados_Electronica_Merged.html"
with open(merged_file_path, 'w', encoding='utf-8') as merged_file:
    merged_file.write(merged_html_content)

merged_file_path

#Code answer for prompt n2

import pandas as pd
from bs4 import BeautifulSoup

# Load the merged HTML file
merged_file_path = "/mnt/data/Amazon.es_Auriculares_Inalambricos_Marca_10_Seleccionados_Electronica_Merged.html"
with open(merged_file_path, 'r', encoding='utf-8') as file:
    merged_content = file.read()
    soup = BeautifulSoup(merged_content, 'html.parser')

# Initialize lists to store scraped data
titles = []
prices = []
retail_prices = []
star_ratings = []
num_ratings = []
sold_last_month = []

# Extract the required information
for item in soup.find_all('div', class_='s-main-slot s-result-list s-search-results sg-row'):
    title = item.find('span', class_='a-size-base-plus a-color-base a-text-normal')
    price = item.find('span', class_='a-price')
    retail_price = item.find('span', class_='a-price a-text-price')
    star_rating = item.find('span', class_='a-icon-alt')
    num_rating = item.find('span', class_='a-size-base s-underline-
text')
    sold_month = item.find('span', class_='a-size-base a-color-
secondary')

    titles.append(title.text if title else None)
    prices.append(price.find('span', class_='a-offscreen').text if
price else None)
    retail_prices.append(retail_price.find('span', class_='a-
offscreen').text if retail_price else None)
    star_ratings.append(star_rating.text if star_rating else None)
    num_ratings.append(num_rating.text if num_rating else None)
    sold_last_month.append(sold_month.text if sold_month else None)

# Create a DataFrame
data = {
    'Title': titles,
    'Price': prices,
    'Recommended Retail Price': retail_prices,
    'Star Rating': star_ratings,
    'Number of Ratings': num_ratings,
    'Sold Last Month': sold_last_month
}
df = pd.DataFrame(data)

# Save the DataFrame to an Excel file
output_file_path = "/mnt/data/Amazon_Items_Scraped.xlsx"
df.to_excel(output_file_path, index=False)

import ace_tools as tools;
tools.display_dataframe_to_user(name="Amazon Items Data",
dataframe=df)

output_file_path

#Code answer for prompt n3

# Renaming columns
data.rename(columns={
    'Title': 'product_description',
    'Price': 'price',
    'Recommended Retail Price': 'recommended_retail_price',
    'Star Rating': 'star_rating',
    'Number of Ratings': 'number_of_ratings',
    'Sold Last Month': 'units_sold_last_moth'
}, inplace=True)

# Cleaning 'price' column
data['price'] = data['price'].str.replace('€', '').str.replace(',',
'.').astype(float)
```

```
# Cleaning 'recommended_retail_price' column
data['recommended_retail_price'] =
data['recommended_retail_price'].str.replace('€', '').str.replace(',',
'.').astype(float)

# Cleaning 'star_rating' column
data['star_rating'] = data['star_rating'].str.replace(' de 5
estrellas', '').str.replace(',', '.').astype(float)

# Cleaning 'units_sold_last_moth' column
data['units_sold_last_moth'] =
data['units_sold_last_moth'].str.replace(' comprados el mes pasado',
'')

# Display the first few rows of the cleaned data
import ace_tools as tools;
tools.display_dataframe_to_user(name="Cleaned Amazon Wireless Earbuds
Data", dataframe=data.head())

data.head()

#Code answer for prompt n4

# Convert column to string before applying the function
data['units_sold_last_moth'] =
data['units_sold_last_moth'].astype(str)

# Function to clean 'units_sold_last_moth' column
def clean_units_sold(value):
    value = value.replace('\xa0', ' ')
    if ' mil+' in value:
        return int(value.replace(' mil+', '')) * 1000
    elif '+' in value:
        return int(value.replace('+', ''))
    else:
        return ''

# Apply the cleaning function again
data['units_sold_last_moth'] =
data['units_sold_last_moth'].apply(clean_units_sold)

# Display the cleaned dataframe to the user
tools.display_dataframe_to_user(name="Updated Amazon Wireless Earbuds
Data with Cleaned Units Sold", dataframe=data.head())

# Save the updated dataframe to an Excel file
updated_file_path =
'/mnt/data/Updated_Amazon_Wireless_Earbuds_Data.xlsx'
data.to_excel(updated_file_path, index=False)
```

```
updated_file_path

# Code answer for prompt n5

# List of brands
brands_list = ['Xiaomi', 'Sony', 'JBL', 'HUAWEI', 'Philips', 'JVC',
'Apple', 'Beats by Dr. Dre', 'Energy Sistem', 'LG']

# Function to find the brand in the product description
def find_brand(description):
    for brand in brands_list:
        if brand in description:
            return brand
    return 'Other'

# Apply the function to create a new column 'brands'
data['brands'] = data['product_description'].apply(find_brand)

# Display the updated dataframe to the user
tools.display_dataframe_to_user(name="Amazon Wireless Earbuds Data
with Brands", dataframe=data.head())

# Save the updated dataframe to an Excel file
final_file_path =
'/mnt/data/Amazon_Wireless_Earbuds_Data_with_Brands.xlsx'
data.to_excel(final_file_path, index=False)

final_file_path

# Code answer for prompt n6

# Define the keywords for each category
device_life_keywords = ['battery', 'autonomy', 'bateria', 'autonomia',
'horas', 'hours']
noise_cancel_keywords = ['cancelacion', 'ruido', 'noise',
'cancellation', 'supresion', 'supression']
fast_charging_keywords = ['carga rapida', 'fast charging']
compatibility_keywords = ['compatibles', 'compatible']

# Function to check for keywords in product_description and return
'yes' or 'no'
def check_keywords(description, keywords):
    return 'yes' if any(keyword in description.lower() for keyword in
keywords) else 'no'

# Apply the function to create the new columns
data['device_life'] = data['product_description'].apply(lambda x:
check_keywords(x, device_life_keywords))
data['noise_cancel'] = data['product_description'].apply(lambda x:
check_keywords(x, noise_cancel_keywords))
```

```
data['fast_charging'] = data['product_description'].apply(lambda x:
check_keywords(x, fast_charging_keywords))
data['compatibility'] = data['product_description'].apply(lambda x:
check_keywords(x, compatibility_keywords))

# Create the 'punctuation' column by summing the 'yes' values in the
four new columns
data['punctuation'] = data[['device_life', 'noise_cancel',
'fast_charging', 'compatibility']].apply(lambda x:
x.str.count('yes')).sum(axis=1)

import ace_tools as tools;
tools.display_dataframe_to_user(name="Amazon Wireless Earbuds Data
with Classification", dataframe=data)

# Display the updated dataframe
data.head()

# Code answer for prompt n7

# First, remove exact duplicates based on product_description
total_rows_before = len(data)
data.drop_duplicates(subset='product_description', inplace=True)
total_rows_after = len(data)
number_of_duplicates = total_rows_before - total_rows_after

# Create a new column 'product_id'
data['product_id'] = data.groupby('brands').cumcount() + 1
data['product_id'] = data['brands'] + '_' +
data['product_id'].astype(str)

import ace_tools as tools;
tools.display_dataframe_to_user(name="Amazon Wireless Earbuds Data
with Product ID", dataframe=data)

# Display the number of duplicates and the total number of rows before
and after the scan
total_rows_before, total_rows_after, number_of_duplicates

# Code answer for prompt n8

# Updating the brand names based on product_id
data.loc[data['product_id'] == 'Other_1', 'brands'] = 'Xiaomi'
data.loc[data['product_id'] == 'Other_2', 'brands'] = 'Philips'
data.loc[data['product_id'] == 'Other_3', 'brands'] = 'Philips'
data.loc[data['product_id'] == 'Other_4', 'brands'] = 'Beats by Dr.
Dre'
data.loc[data['product_id'] == 'Other_5', 'brands'] = 'Philips'
data.loc[data['product_id'] == 'Other_6', 'brands'] = 'Philips'
data.loc[data['product_id'] == 'Other_7', 'brands'] = 'Sony'
```

```
# Updating brand names for product descriptions beginning with 'Beats'
data.loc[data['product_description'].str.startswith('Beats'),
'brands'] = 'Beats by Dr. Dre'

# Updating the product_id for changed brand entries
data['product_id'] = data.groupby('brands').cumcount() + 1
data['product_id'] = data['brands'] + '_' +
data['product_id'].astype(str)

import ace_tools as tools;
tools.display_dataframe_to_user(name="Updated Brand Data",
dataframe=data)

# Display the updated data
data[data['brands'].isin(['Xiaomi', 'Philips', 'Beats by Dr. Dre',
'Sony'])]

# Code answer for prompt n9

# Counting the number of items per brand and sorting them from top to
bottom
items_per_brand_sorted = data['brands'].value_counts().reset_index()
items_per_brand_sorted.columns = ['brands', 'item_count']

# Displaying the results
import ace_tools as tools; tools.display_dataframe_to_user(name="Items
Per Brand Sorted", dataframe=items_per_brand_sorted)

items_per_brand_sorted

# Code answer for prompt n10

# Filtering the dataframe to include only brands with more than 10
items
brands_to_analyze =
items_per_brand_sorted[items_per_brand_sorted['item_count'] > 10]
['brands']

filtered_data = data[data['brands'].isin(brands_to_analyze)]

import ace_tools as tools;
tools.display_dataframe_to_user(name="Filtered Data for Brands with
More Than 10 Items", dataframe=filtered_data)

filtered_data.head()

# Code answer for prompt n11

# Calculate the mean and median values for each column and brand
means = filtered_data.groupby('brands').mean()
medians = filtered_data.groupby('brands').median()
```

```
# Dictionary to store the mean and median values used for each
iteration
imputation_values = {
    'price': {},
    'recommended_retail_price': {},
    'units_sold_last_moth': {}
}

# Function to replace missing values based on mean or median
def replace_missing_values(df, column, brand, method):
    if method == 'mean':
        value = means.loc[brand, column]
    elif method == 'median':
        value = medians.loc[brand, column]
    else:
        raise ValueError("Method must be either 'mean' or 'median'")

    df.loc[df['brands'] == brand, column] = df.loc[df['brands'] ==
brand, column].fillna(value)
    return value

# Perform the replacements as specified
brands = ['Beats by Dr. Dre', 'Sony', 'JBL', 'Xiaomi']

# Price replacements
imputation_values['price']['Beats by Dr. Dre'] =
replace_missing_values(filtered_data, 'price', 'Beats by Dr. Dre',
'mean')
imputation_values['price']['Sony'] =
replace_missing_values(filtered_data, 'price', 'Sony', 'mean')
imputation_values['price']['JBL'] =
replace_missing_values(filtered_data, 'price', 'JBL', 'median')
imputation_values['price']['Xiaomi'] =
replace_missing_values(filtered_data, 'price', 'Xiaomi', 'median')

# Recommended Retail Price replacements
imputation_values['recommended_retail_price']['Beats by Dr. Dre'] =
replace_missing_values(filtered_data, 'recommended_retail_price',
'Beats by Dr. Dre', 'mean')
imputation_values['recommended_retail_price']['Sony'] =
replace_missing_values(filtered_data, 'recommended_retail_price',
'Sony', 'mean')
imputation_values['recommended_retail_price']['Xiaomi'] =
replace_missing_values(filtered_data, 'recommended_retail_price',
'Xiaomi', 'mean')
imputation_values['recommended_retail_price']['JBL'] =
replace_missing_values(filtered_data, 'recommended_retail_price',
'JBL', 'median')
```

```
# Units Sold Last Month replacements
imputation_values['units_sold_last_moth']['Beats by Dr. Dre'] =
replace_missing_values(filtered_data, 'units_sold_last_moth', 'Beats
by Dr. Dre', 'mean')
imputation_values['units_sold_last_moth']['Sony'] =
replace_missing_values(filtered_data, 'units_sold_last_moth', 'Sony',
'median')
imputation_values['units_sold_last_moth']['Xiaomi'] =
replace_missing_values(filtered_data, 'units_sold_last_moth',
'Xiaomi', 'median')
imputation_values['units_sold_last_moth']['JBL'] =
replace_missing_values(filtered_data, 'units_sold_last_moth', 'JBL',
'median')

import ace_tools as tools;
tools.display_dataframe_to_user(name="Imputation Values Used",
dataframe=pd.DataFrame(imputation_values))

# Display the imputation values used
imputation_values

# Code answer for prompt n12

# Calculate the mean of star_rating and number_of_ratings by brand
brand_star_rating_mean = data.groupby('brands')
['star_rating'].mean().reset_index()
brand_number_of_ratings_mean = data.groupby('brands')
['number_of_ratings'].mean().reset_index()

# Create a figure with two subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(16, 6))

# Bar graph for mean star_rating by brand
axes[0].barh(brand_star_rating_mean['brands'],
brand_star_rating_mean['star_rating'], color='skyblue')
axes[0].set_title('Mean Star Rating by Brand')
axes[0].set_xlabel('Mean Star Rating')
axes[0].set_ylabel('Brands')
axes[0].invert_yaxis()  # Invert y axis to match the order in the
second graph

# Bar graph for mean number of ratings by brand
axes[1].barh(brand_number_of_ratings_mean['brands'],
brand_number_of_ratings_mean['number_of_ratings'], color='salmon')
axes[1].set_title('Mean Number of Ratings by Brand')
axes[1].set_xlabel('Mean Number of Ratings')
axes[1].set_ylabel('Brands')
axes[1].invert_yaxis()  # Invert y axis to match the order in the
first graph
```

```
plt.tight_layout()
plt.show()

# Code answer to promtp n13:

# Calculate the total units sold last month by brand and sort from
highest to lowest
brand_units_sold = data.groupby('brands')
['units_sold_last_moth'].sum().sort_values(ascending=False).reset_inde
x()

# Get the top 10 products sold last month and sort by units sold
top_products_sold = data[['product_id',
'units_sold_last_moth']].sort_values(by='units_sold_last_moth',
ascending=False).head(10)

# Create a figure with two subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(18, 8))

# Bar graph of units_sold_last_month by brand
axes[0].barh(brand_units_sold['brands'],
brand_units_sold['units_sold_last_moth'], color='lightgreen')
axes[0].set_title('Units Sold Last Month by Brand')
axes[0].set_xlabel('Units Sold Last Month')
axes[0].set_ylabel('Brands')
axes[0].invert_yaxis()  # Invert y axis to show top brands at the top

# Bar graph of top 10 products sold last month
axes[1].barh(top_products_sold['product_id'],
top_products_sold['units_sold_last_moth'], color='lightcoral')
axes[1].set_title('Top 10 Products Sold Last Month')
axes[1].set_xlabel('Units Sold Last Month')
axes[1].set_ylabel('Product ID')
axes[1].invert_yaxis()  # Invert y axis to show top products at the
top

plt.tight_layout()
plt.show()

# Code answer for prompt n14

import numpy as np

# Calculate mean units sold last month by brand
mean_units_by_brand = data.groupby('brands')
['units_sold_last_moth'].mean()
# Calculate mean star rating by brand
mean_star_by_brand = data.groupby('brands')['star_rating'].mean()

# Set distinct and clear colors for each brand
brand_colors = {'JBL': 'darkblue', 'Beats by Dr. Dre': 'darkred',
'Xiaomi': 'darkgreen', 'Sony': 'darkviolet'}

# Scatter plot for correlation between star_rating and
units_sold_last_month
plt.figure(figsize=(12, 8))
for brand, group_data in data.groupby('brands'):
    plt.scatter(group_data['star_rating'],
group_data['units_sold_last_moth'],
                color=brand_colors[brand], label=brand, alpha=0.6,
edgecolors='black')

# Add dashed lines for mean values by brand
for brand in mean_units_by_brand.index:
    plt.axhline(y=mean_units_by_brand[brand],
color=brand_colors[brand], linestyle='--', linewidth=1)
    plt.axvline(x=mean_star_by_brand[brand],
color=brand_colors[brand], linestyle='--', linewidth=1)

plt.title('Correlation between Star Rating and Units Sold Last Month')
plt.xlabel('Star Rating')
plt.ylabel('Units Sold Last Month')
plt.legend(title='Brand', loc='center left', bbox_to_anchor=(1, 0.5))
plt.grid(True)
plt.show()

# Code answer to prompt n15

# Calculate mean number of ratings by brand
mean_ratings_by_brand = data.groupby('brands')
['number_of_ratings'].mean()

# Scatter plot for correlation between star_rating and
number_of_ratings
plt.figure(figsize=(12, 8))
for brand, group_data in data.groupby('brands'):
    plt.scatter(group_data['star_rating'],
group_data['number_of_ratings'],
                color=brand_colors[brand], label=brand, alpha=0.6,
edgecolors='black')

# Add dashed lines for mean values by brand
for brand in mean_ratings_by_brand.index:
    plt.axhline(y=mean_ratings_by_brand[brand],
color=brand_colors[brand], linestyle='--', linewidth=1)
    plt.axvline(x=mean_star_by_brand[brand],
color=brand_colors[brand], linestyle='--', linewidth=1)

plt.title('Correlation between Star Rating and Number of Ratings')
plt.xlabel('Star Rating')
plt.ylabel('Number of Ratings')
plt.legend(title='Brand', loc='center left', bbox_to_anchor=(1, 0.5))
plt.grid(True)
plt.show()

# Code answer to prompt n16:

# Create a scatter plot for the correlation between price and
units_sold_last_month with reversed color scale
plt.figure(figsize=(12, 8))
sc = plt.scatter(data['price'], data['units_sold_last_moth'],
                 c=data['star_rating'], cmap='viridis_r', alpha=0.6,
edgecolors='black')

# Add color bar to represent the star rating scale
cbar = plt.colorbar(sc)
cbar.set_label('Star Rating')

plt.title('Correlation between Price and Units Sold Last Month with
Star Rating as Color Scale')
plt.xlabel('Price')
plt.ylabel('Units Sold Last Month')
plt.grid(True)
plt.show()

# Code answer for prompt n17

# Create a figure with subplots for each brand with a grey background
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

# Flatten the axes array for easier iteration
axes = axes.flatten()

# Iterate over each brand and create a scatter plot for each
for idx, (brand, group_data) in enumerate(data.groupby('brands')):
    ax = axes[idx]
    ax.set_facecolor('lightgrey')
    sc = ax.scatter(group_data['price'],
group_data['units_sold_last_moth'],
                    c=group_data['star_rating'],
cmap=brand_color_maps[brand], alpha=0.6, edgecolors='black')
    ax.set_title(f'Price vs. Units Sold for {brand}')
    ax.set_xlabel('Price')
    ax.set_ylabel('Units Sold Last Month')
    cbar = plt.colorbar(sc, ax=ax)
    cbar.set_label('Star Rating')
    ax.grid(True)

plt.tight_layout()
plt.show()
```

```python
# Code to be able to see screenshots and download the file to pdf (not from ChatGPT)

from IPython.display import display
from PIL import Image
```