Project Plan phase 1,2🎀

🎀 Extracting Data From SHEGLAM

SHEGLAM Website Link

Bonus Link

🎀 Project idea and background 💡

The idea of this project is to collect and analyze product data from **SHEGLAM** cosmetics website using web scraping techniques. By extracting detailed information about various makeup and beauty products, including names, prices, ratings, and subcategories, we aim to clean and structure the data for further analysis. This will help uncover patterns in product categories, pricing strategies, ratings, and popularity.

The beauty and cosmetics industry has seen a significant rise in online shopping, making it crucial to understand consumer trends and product offerings on e-commerce platforms. **SHEGLAM**, a popular and affordable cosmetics brand, provides a wide range of products that make it an ideal website for data extraction. Scraping **SHEGLAM** website allows us to gather valuable product data and perform analysis to discover trends and insights that can benefit both consumers and businesses.

🎀 Project goals 📈

1. Extract product data from **SHEGLAM** website using Selenium.
2. Extract product data from **SHEGLAM** website using Selenium.
3. Identify and flag Best Seller products for comparison and deeper analysis.
4. Clean and structure the scraped data for further processing.
5. Visualize key insights from data using data visualization tools.
6. Save the final processed data in a database (e.g., **MongoDB**) .

🎀 Why SHEGLAM? 👀

SHEGLAM was selected as the data source for this project for several reasons:

- Diverse Product Range: **SHEGLAM** offers an extensive selection of beauty and personal care products across various categories, from makeup essentials to hair care and beauty tools, providing a wealth of data for in-depth analysis.
- User-Friendly Website: The clean, modern, and intuitive structure of the website makes it ideal for web scraping using automation tools, ensuring efficient and accurate data extraction.
- Consistent and Detailed Product Information: The website provides detailed and consistent product information, including prices, and user ratings. This makes it an excellent resource for performing data analysis on consumer preferences, price points, and product quality.
- Emerging Brand with a Global Reach: **SHEGLAM** is an emerging player in the beauty industry, offering affordable, high-quality products that have garnered attention worldwide. Analyzing data from such a platform allows us to gain insights into current beauty trends and consumer behavior on a global scale.
- Real-Time Data for Dynamic Trends: The constantly updated product listings and trends on **SHEGLAM** provide real-time data, ensuring that the analysis reflects the latest market shifts and consumer preferences.

🎀 Tools and Libraries Used in this phase (1) 📚

- **Python**: Main programming language used for scripting and data processing.
- **Selenium**: For web automation and scraping dynamic content from **SHEGLAM** website.
- **Pandas**: For data manipulation, structuring, and saving the dataset to CSV.
- **Regular Expressions (re module)**: Used for pattern matching in product names to classify subcategories.
- **HTML Parser (html.unescape)**: To clean and decode HTML entities in product names.
- **Time module**: To add delays and handle page loading dynamically.
- **WebDriverWait and Expected Conditions (from Selenium)**: To ensure elements load before interaction.

🎀 Data to be Extracted and Utilized 📋

Data	Description
Category	The top-level category (e.g., Makeup, Hair)
Product Name	The full name or title of the product
Subcategory	A more specific classification (e.g., Lipstick, Shampoo)
Price	The listed price of the product
Rating	The average customer rating (out of 5 stars)
Best Seller	A flag indicating whether the product is marked as a bestseller

---

🎀 Data Extraction Task 📤

1. Automation & Scraping of Sheglam Product Data:

The notebook starts by importing necessary **Python libraries**:

- `selenium` : Automates browser actions to scrape **dynamic** content.
  - `webdriver` : Launches and controls the Chrome browser.
  - `By` : Helps locate HTML elements (e.g., by **CSS** selector or **XPath**).
  - `WebDriverWait` : Enables waiting for specific elements to load.
  - `expected_conditions as EC` : Used with `WebDriverWait` to wait for elements (like buttons or content) to become available.
- `time` : Adds delays (`sleep`) to allow dynamic content to load properly.
- `html.unescape` : Converts HTML character codes (like `&amp;`) back into normal characters (`&`), ensuring clean product names.
- `pandas` : Handles data storage and manipulation.
- `matplotlib` and `seaborn` : Used for visualization.
- `re` : Supports regular expressions, useful for advanced pattern matching and text extraction (included for potential text parsing or data cleaning tasks).
- `plotly.express` : you gain access to a powerful yet simple tool for creating interactive and publication-quality visualizations.

WebDriver Configuration

```
driver = webdriver.Chrome()
wait = WebDriverWait(driver, 10)
```

A Selenium WebDriver is initialized to control a Chrome browser. `WebDriverWait` is used to manage delays while waiting for elements to load on the page.

2. NLP-Based Product Categorization Logic:

To enable structured analysis of **SHEGLAM** product data, we employed a simple Natural Language Processing (NLP) technique—**keyword matching**—for automatic product categorization. This method analyzes product names using a predefined set of keywords to determine the most appropriate **subcategory and main category**.

We created a dictionary called **mapping_lists**, which groups keywords under six main categories: Face, Eyes, Lips, Tools & Others, Hair Tools, and Hair Care. Each main category contains **subcategories** (e.g., "Foundation", "Mascara", "Lip Balm"), and each subcategory is associated with a list of relevant **keywords** that are commonly found in product titles or descriptions.

During the scraping process, the product name is first converted to lowercase and then compared to all keywords in the selected category using simple string containment logic. This matching is performed by the **match_nlp_attributes** function. If a match is found, the corresponding subcategory is assigned to the product.

For example:

A product titled "**Liquid Concealer Pen**" contains the keyword "concealer", so it is classified under the "Concealer" subcategory in "Face".

To streamline this matching process, we constructed a flattened **lookup dictionary** called subcategory_keywords. This dictionary maps each keyword to its corresponding subcategory for a given main category, significantly improving matching efficiency.

This keyword-based **NLP** approach allows for consistent, automated categorization of products, making it easier to organize, analyze, and visualize the scraped dataset.

Then, a secondary dictionary called `subcategory_keywords` is constructed to **flatten the original mapping**, making it easier to look up which keyword corresponds to which subcategory under each main category.

3. Function Summary for Product Scraping Logic:

click_main_view_more():

This function repeatedly clicks the "**View More**" button on the product listing page until all products are visible. It uses Selenium's wait conditions and JavaScript execution to ensure the button is clickable before clicking, and exits when no more button is found.

get_product_details(el):

Extracts key information from a single product element such as:

Name (using the product name selector),

Price (from the sale price section), and

Rating/Stars, computed by parsing the filled star percentages from the style attributes.

It uses default values if any element is missing to ensure robustness.

match_nlp_attributes(name, category):

Implements a simple **NLP-based keyword** matching technique. It converts the product name to lowercase and searches for predefined keywords related to subcategories within the name. If a match is found, it returns the subcategory label; otherwise, it returns "N/A".

scrape_category(category_name):

This is the **main function** that orchestrates the scraping of all products under a specific category.

It loads all products using click_main_view_more().

Then iterates over each product to extract **details** via get_product_details().

For "Hair Tools", it handles an **additional step**: opens the product in a new tab and scrapes **plug/voltage** details.

Finally, it uses match_nlp_attributes() to classify the product into a subcategory, and appends the structured data to a global list.

4. The script follows a structured sequence to scrape all product categories from the SHEGLAM website:

1. Navigate to Homepage

- The scraper opens the homepage(**SHEGLAM**) and waits for the content to load.

2. Scrape Main Product Categories

- A dictionary `main_categories` stores XPaths of the major product categories like:
  - Face
  - Eyes
  - Lips
  - Tools & Others
- The script clicks each category, waits for the products to load, and then calls `scrape_category()` to extract the product data.
- After scraping each category, it returns to the homepage to continue.

3. Scrape Hair Section (SHEGLAM HAIR)

- The scraper clicks on the **"SHEGLAM HAIR"** tab.
- It loops through two subcategories:
  - **Hair Tools**
  - **Hair Care**
- For each, it navigates into the section, scrapes the data, and then returns back.

4. Scrape Best Seller Products

- The scraper visits the dedicated **"Best Sellers"** collection page.
- It uses `click_main_view_more()` to ensure all items are loaded.
- It then **extracts** the names of the best sellers and stores them in a set `best_seller_products`, which is used later to mark those items in the main dataset.

Each product in the `data` list is checked. If its name is found in `best_seller_products`, its `Best Seller` flag is set to **1**.

5. Saving data in structured file:

- The **final** data is converted to a pandas DataFrame and saved as `sheglam_products.csv`
- Finally, the Selenium browser session is **closed** using `driver.quit()`

Category	Name	Price	Stars	Subcategory	Best Seller
Face	Melon Melt Niacinamide Serum Primer	$8.49	4.9	Primer	1
Face	Good Grip Hydrating Primer	$9.99	4.9	Primer	1
Face	Pore No More Primer	$7.99	4.8	Primer	1
Face	Lock'd In Setting Spray	$6.49	4.8	Setting Spray	1
Face	Good Grip Hydrating Prime & Set Spray	$6.49	4.8	Setting Spray	1
Face	Glow Bloom Liquid Highlighter-Vanilla Frost	$5.99	4.7	Highlighter	1
Face	Pore Eraser Blurring Stick	$4.99	4.7	Primer	1
Face	Hydro-Touch Refreshing Setting Powder	$7.49	5.0	Powder	1
Face	Buttery Bliss Blush Stick-Guava Juice	$5.99	0.0	Blush	1

A sample of the dataset

The Chrome browser is gracefully closed with `driver.quit()`.

🎀 Data Cleaning, Processing, and Regular Expressions Task🧹

1. cleaning and formatting to improve data quality and consistency:

- During the data **cleaning process**, we started by inspecting the dataset structure and checking for missing values.We then **identified and displayed** any fully **duplicated** rows to ensure data integrity. Additionally, we explored the **unique values** in the *Category* and *Subcategory* columns to understand the diversity within these features.
- To **detect** outliers, we performed an interquartile range (IQR) analysis on the *Price* column. Based on this, we identified and listed records with abnormal price values. Finally, we visualized the price distribution across different categories using a boxplot, which helped in spotting outliers and comparing pricing trends among categories.

> Note:
>
> I didn't remove the price outliers because they could represent valid variations, like premium products or special deals. Removing them might lose important information or distort the analysis.

2. Regular Expression part:

- Price Cleaning & Whitespace Detection. To ensure accurate numerical analysis, we cleaned the `Price` column by removing any currency symbols (**like** `$`) using **regular expressions**, then converted the values to float type for further analysis.
- Additionally, we scanned all object-type columns to detect the presence of **extra whitespaces** (e.g., two or more consecutive spaces) which might affect data consistency. This step helped ensure textual fields were clean and uniform before processing.
- We applied **regular expressions (regex)** to identify and **extract** specific product **collections** from the product names. We compiled a list of known collection patterns (e.g., *Harry Potter*, *Hello Kitty*, *Rick and Morty*, etc.) and combined them into a single regex pattern to match any of them, regardless of formatting inconsistencies or spacing issues.

🎀 Data Analysis Task🧪

1. Computing basic statistics from the extracted data :

The dataset contains **357** entries for both price and star ratings. The average price is **$8.52**, ranging from **$0.79** to **$49.99**. The majority of prices are between **$5** and **$8**. The average star rating is **3.81**, with most items rated between **4.30** and **5.00**, and ratings range from **0.00** to **5.00**.

2. Best Seller Product Proportions

The dataset shows that **31.37%** of the products are marked as "**Best Seller**," while **68.63%** are **not**. This indicates that a significant portion of the products falls into the non-best seller category.

3. Average Price and Stars by Category

The analysis shows the average price and star ratings for different product categories:

- **Eyes**: **$6.93**, **3.75** stars
- **Face**: **$7.46**, **3.77** stars
- **Hair Care**: **$6.69**, **1.96** stars
- **Hair Tools**: **$37.70**, **4.66** stars
- **Lips**: **$7.73**, **3.70** stars
- **Tools & Others**: **$4.73**, **4.21** stars

Hair Tools stands out with the **highest price** and **star rating**, while **Hair Care** has the **lowest average** rating.

4. Average Price and Stars by Subcategory

The average price and star ratings by subcategory show that **Hot-Air Brushes & Straight Hair Combs** have the highest average price at **$49.42**, while **Sponge** has the lowest at **$2.75**. Categories like **Hair Straightener Iron** and **Curling Tongs & Curling Wands** have high star ratings, with **Hair Straightener Iron** reaching a perfect **5** stars. Some subcategories, such as **Eye Primer** and **Lashes**, have no ratings, which could indicate lack of reviews or unreviewed products.

5. Best Seller Distribution by Category

To identify which **categories** contain the **most best-selling products**, the data was filtered to include only products marked as "Best Seller". The number of best sellers was then counted for each category.

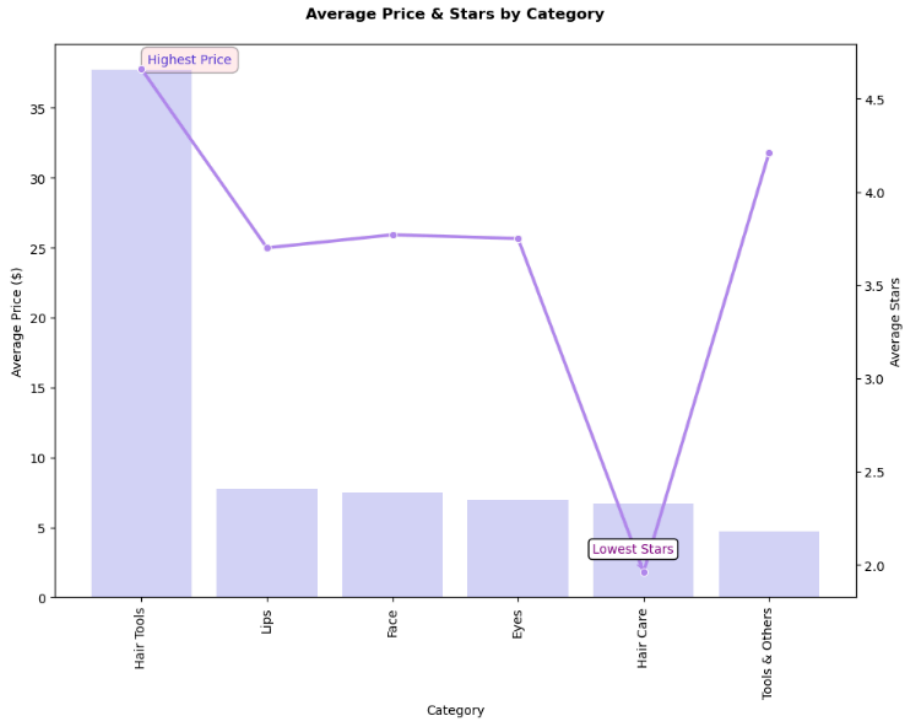
The results show that the **"Face"** category contains the highest number of best seller products (**42**), followed by **"Eyes"** (**27**) and **"Lips"** (**21**). On the other hand, **"Hair Care"** has the fewest best seller products (**2**), indicating a lower popularity or demand in that category.

🎀 Data Visualization Task 📊

1. Average Price and Star Ratings by Category:

It calculates and visualizes the average `Price` and `Stars` for each `Category`. A **bar chart** shows average prices, while a **line** graph overlays average star ratings. Annotations **highlight** the category with the **highest** price and the lowest star rating.

Output:

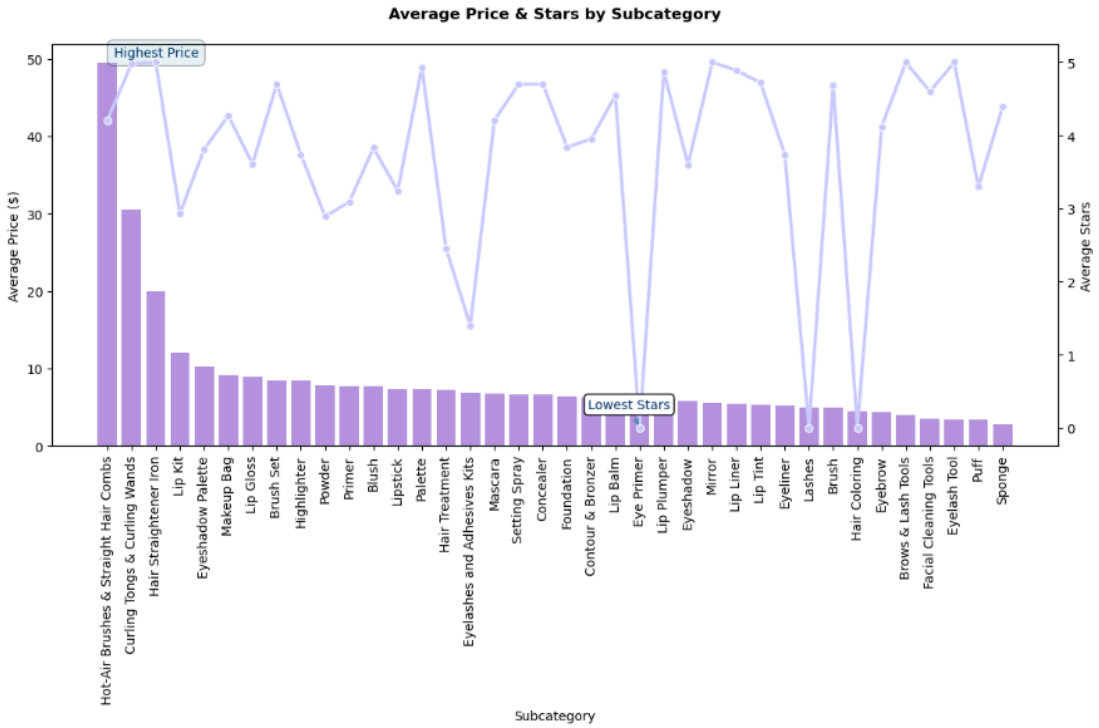


- This analysis showed that **Hair Tools** have the **highest price and stars,** while **Hair Care** has the **lowest Stars**, helping highlight strengths and areas for improvement.

2. Average Price and Star Ratings by Subcategory

It calculates and visualizes the average `Price` and `Stars` for each `Subcategory`. A bar chart shows average prices, while a line graph overlays average star ratings. Annotations highlight the subcategory with the highest price and the lowest star rating.

Output:



The plot **revealed** that there is **no clear relationship** between average price and star ratings. Some subcategories have **high average** prices but relatively **low ratings**, while others have low prices and high ratings. This suggests that **a higher price does not necessarily indicate better customer satisfaction.**

3. Correlation Heatmap Between Product Price and Rating:

It analyzes the **relationship** between '**Price**' and '**Stars**' by calculating their correlation and visualizing it in a heatmap. The **heatmap** helps to easily **identify** how **strongly** these two variables are related.

Output:

```
          Price     Stars
Price  1.000000  0.056445
Stars  0.056445  1.000000
```

- The **correlation** coefficient between **Price and Stars** is approximately **0.056**, indicating a **very weak positive relationship**. This suggests that product **prices** have **little** to **no impact** on customer **ratings**.

4. Comparison of Best Seller and Non-Best Seller Products:

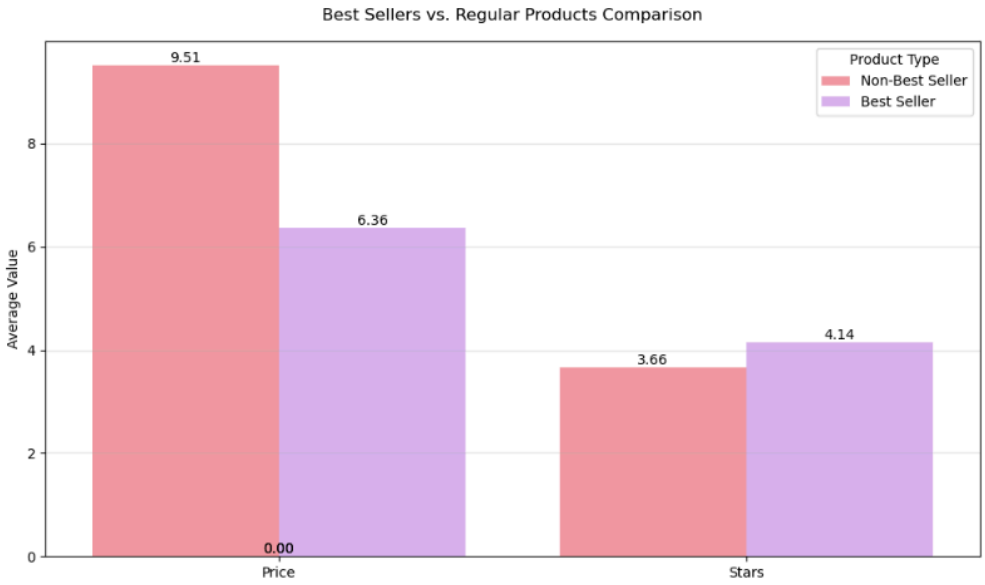
The **goal** of this code is to compare the average '**Price**' and '**Stars**' between '**Best Seller**' and '**Non-Best Seller**' products and visualize the comparison using a **bar** plot. The goal is to understand whether being a best seller is associated with **higher** prices or better ratings. The results are visualized in a

grouped bar chart that clearly shows the difference in **average** values between the **two product types.**

Output:

```
Average Price and Stars in best sellers:

                Price  Stars
Non-Best Seller  9.51   3.66
Best Seller      6.36   4.14
```



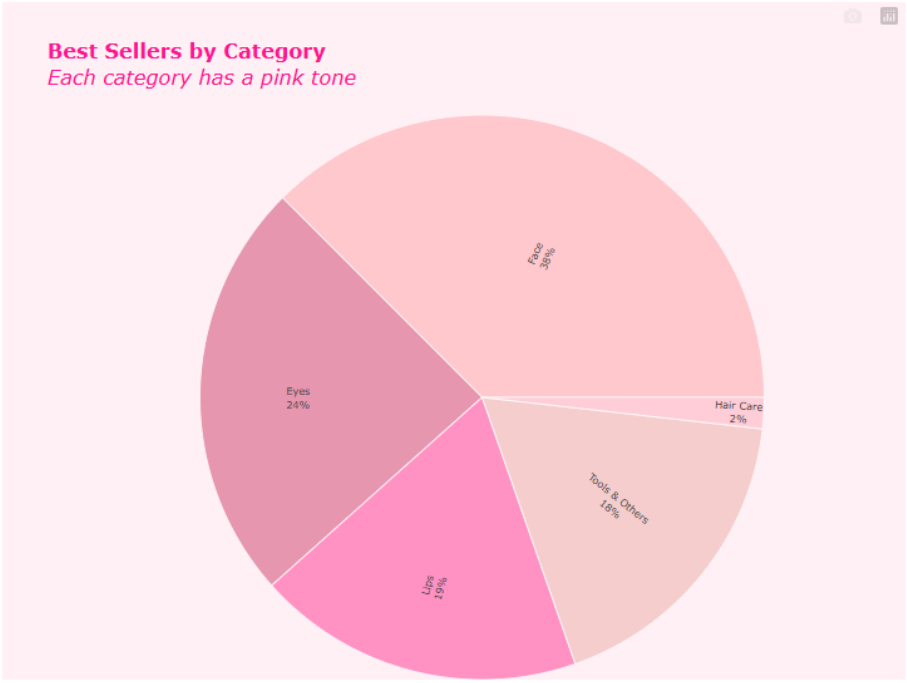
Based on the **results**, we can **conclude** that:

- **Best seller products tend to have lower average prices but higher average ratings compared to non-best seller products.**
  This suggests that **affordability** combined with **higher customer satisfaction** may contribute to a product becoming a **best seller**.

5. Visual Distribution of Best Sellers Across Categories:

We **create** an **interactive** sunburst chart that visualizes the **distribution** of **best-selling** products across different **categories**. It uses different **pink** tones for each category, with the size of each segment representing the number of best-selling products. The chart provides a visually appealing way to **explore how** best sellers are **distributed** across categories.

Output:



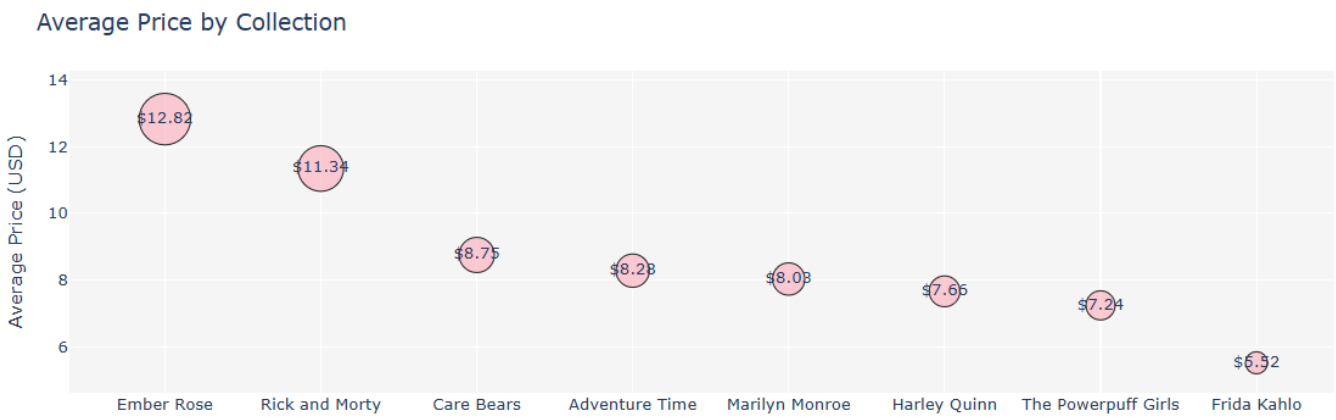
- Among the **best-selling** products, the '**Face**' and '**Eyes**' categories had the **highest proportions**, while '**Hair Care**' had the **lowest** at **2%**."

6. Average Price by Collection: Bubble Chart Visualization:

The **goal** of this code is to **calculate** the **average price** of products within each **collection** and identify which collection has the **highest** and **lowest** average price. **Specifically**

It creates a **bubble chart** that visualizes the average price of products within each **collection**. The size of each bubble is proportional to the average price, with the price displayed inside each bubble. The chart provides an easy-to-understand visual representation of the average prices across different collections.

Output:

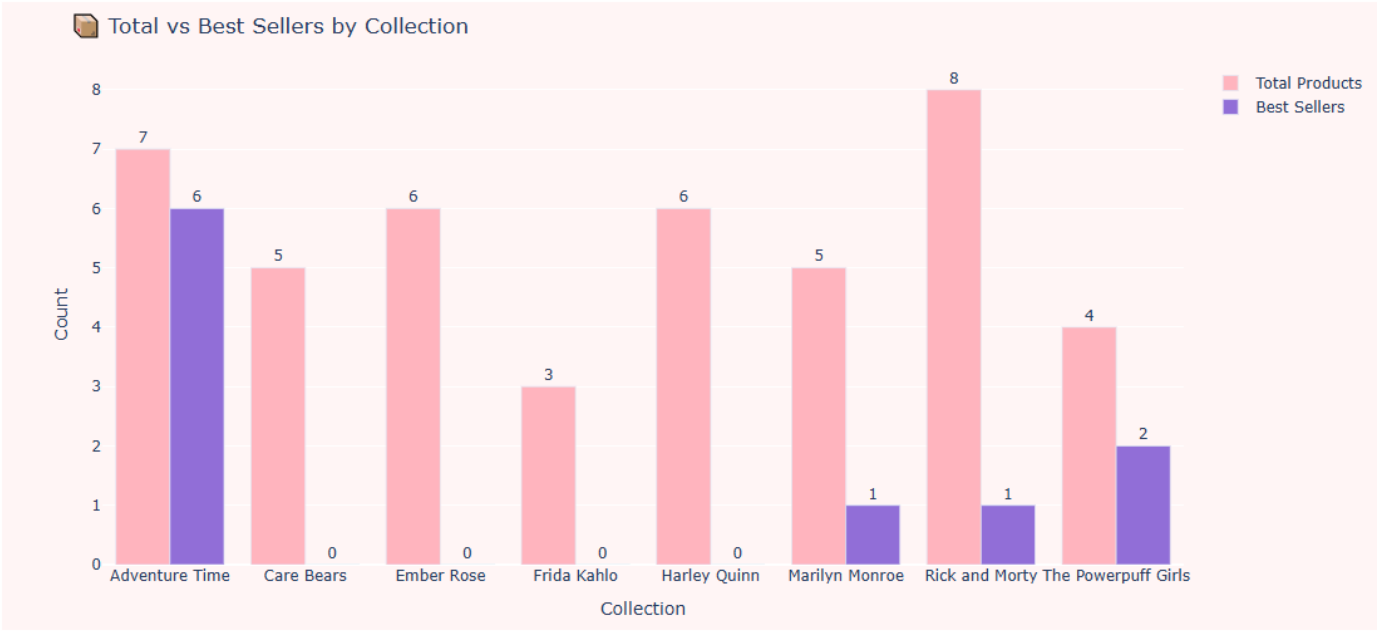


- It **shows** that the "**Ember Rose**" collection has the **highest** average price, while the "**Frida Kahlo**" collection has the **lowest**.

7. Comparison of Total Products and Best Sellers by Collection

A **grouped bar** chart was created to compare the **total** number of **products** and the number of **best sellers** within each **collection**. This visualization provides a clear view of **how popular each collection is**, not only by size but also in terms of product performance. By analyzing this chart, it's easy to identify which collections have a higher concentration of best-selling products relative to their total offerings.

Output:



The **"Total Products vs Best Sellers by Collection"** chart reveals differences in product performance across collections. ***Rick and Morty*** collection has the **highest total product** counts (**8**) but has one of the **lowest** number of **best sellers** (**1**)**, while** Adventure Time has the **second highest total product** counts (**7**) and (**6**) of them are **best sellers , suggesting** popularity or variety, while **other collections** show **low** total product counts, and **low** best seller counts indicating niche appeal.

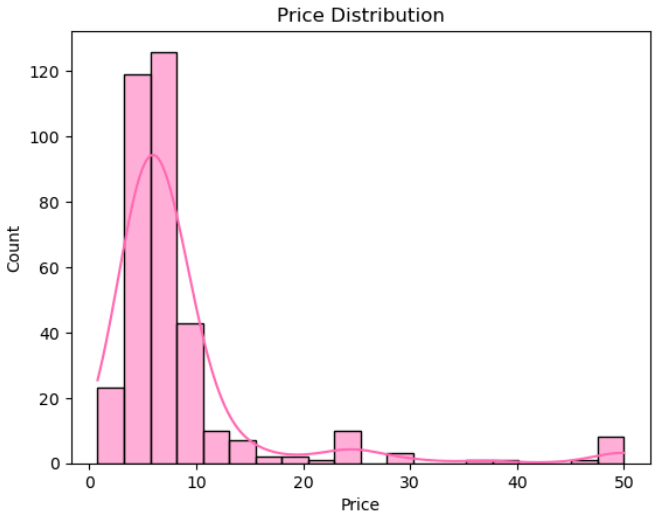
8. Price and Rating Distribution of Products:

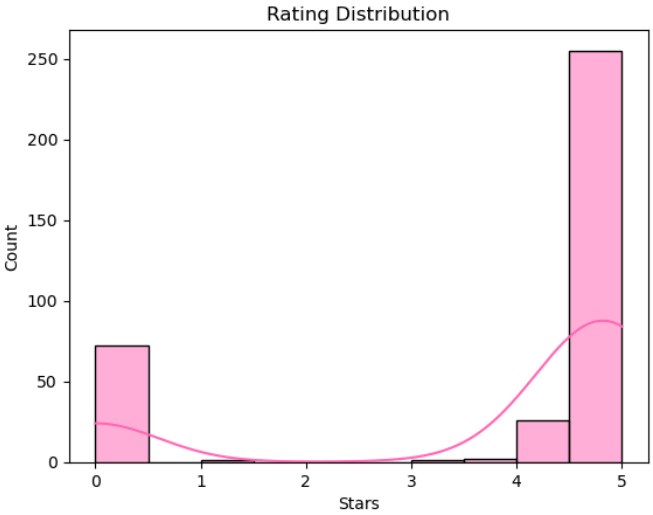
It **visualizes** the **distribution** of **prices** and **ratings** across products:

- **Price Distribution**: A **histogram** with a kernel density estimate (**KDE**) is used to show that **most products** on the site are **budget-friendly**, indicating a concentration of **lower-priced items**.
- **Rating Distribution**: A **histogram** with **KDE** is used to show that **most** products **have high ratings**, **suggesting** that customers generally **rate** the products **highly**.

These visualizations help to understand the **overall pricing and rating trends** on **SHEGLAM**.

Output:





Price Distribution:

Most products are priced between **$0** and **$20**, with fewer high-priced items above **$20**. Products over **$40** are rare.

Rating Distribution:

Most products have high ratings (**4-5 stars**), with few low ratings (**1-2 stars**). 3-star ratings show room for improvement.

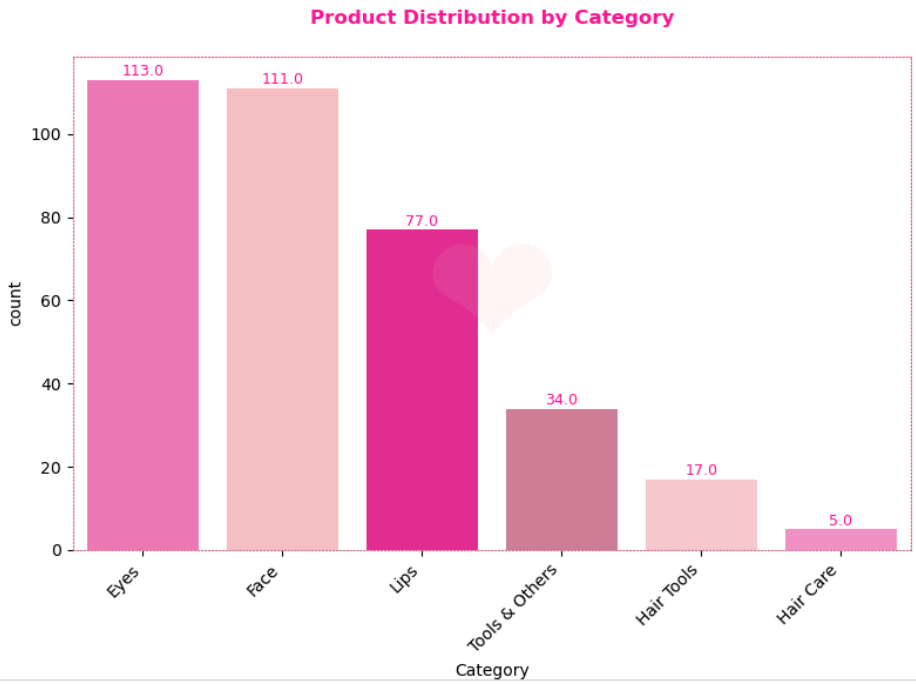
9. Number of Products in Each Category:

The goal of this code is to visualize the **distribution** of products across categories with a **countplot**. The key features of the visualization are:

- **Product Count by Category**: The **bar chart** shows the **number of products** in each **category**, with each bar's **height representing** the count of products.

This visualization helps in **understanding** the product distribution by category, while also incorporating a visually appealing design.

Output:



- We will **notice** that the '**Eyes**' Category has the **highest** number of products (**113**), the '**Hair Care**' Category has the **lowest**(**5**).

10. Word Cloud of Product Names

It generates a **word cloud** visualization that displays the most common words in product names. Key points:

- **Word Cloud**: It visualizes frequently occurring words from the '**Name**' column of the DataFrame `df`. The larger the word, the more often it appears in product names.

This visualization helps to quickly identify frequently used words in product names, offering insights into common themes or trends.

Output:

This visualization highlights the most frequently occurring words in product names. The most common terms include **"SHEGLAM," "X," "Blush," "Palette," "Lipstick,"** and others.

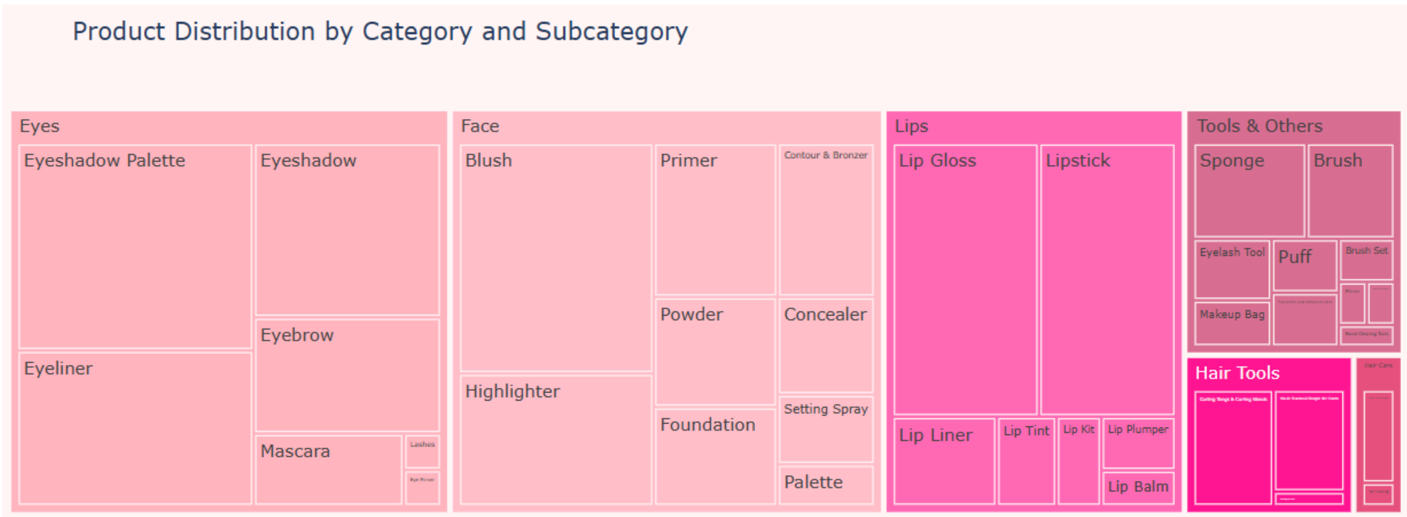
11. **Treemap of Product Distribution by Category and Subcategory:**

It creates an interactive **treemap** that visualizes the distribution of products by both **Category** and **Subcategory**. Key features:

- **Treemap Structure**: The treemap uses the 'Category' and 'Subcategory' columns to organize the data, with each **category** and its **subcategories** represented as colored blocks.

This visualization helps to understand the **hierarchy** and distribution of products across categories and subcategories.

Output:



- The **Face** and **Lips** categories **dominate** the product assortment, with a **wide variety** of subcategories under each.
  In contrast, categories like **Hair Care** and **Hair Tools** have **fewer** products and limited subcategory diversity.
  This suggests a stronger **brand** focus on facial and lip products, while other categories may represent more niche or supplementary offerings.

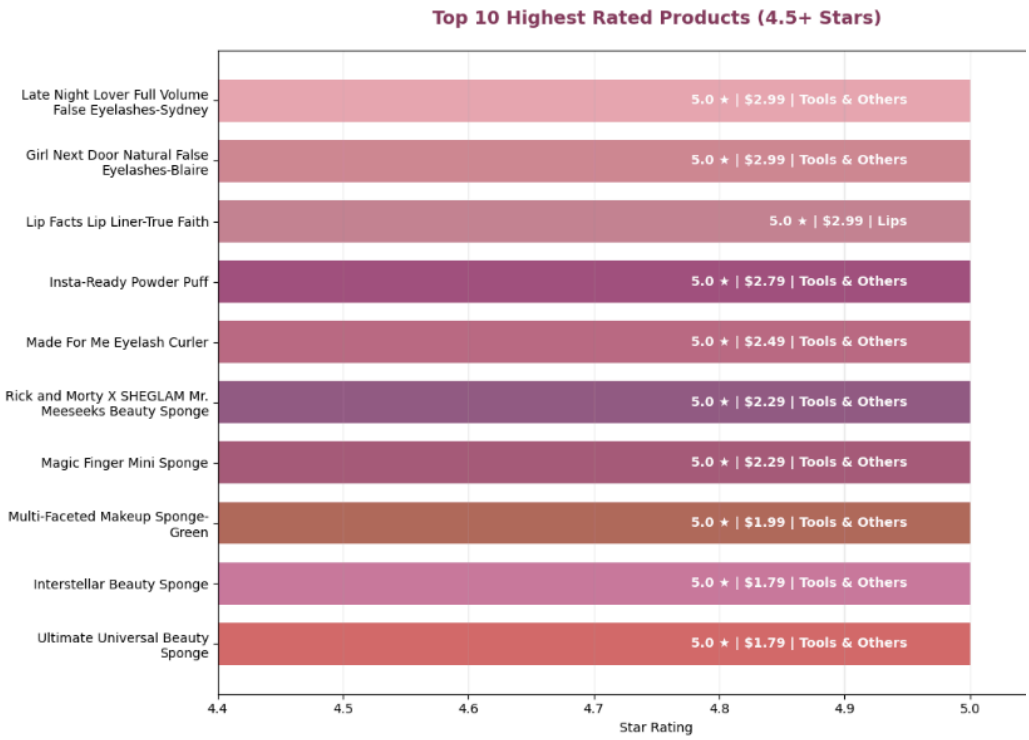
12. **Top 10 Most Loved Products (4.5 and Above):**

The goal of this code is to create a **horizontal bar chart** that displays the **top 10** highest-rated products with a rating of **4.5 stars** or **higher**. Key features:

- **Top 10 Products**: It filters products with a rating of 4.5 or higher and sorts them by **rating** (descending) and **price** (ascending), displaying the top 10.

This visualization helps to highlight the top-rated products along with their price and category, making it easy to identify popular and premium products.

Output:



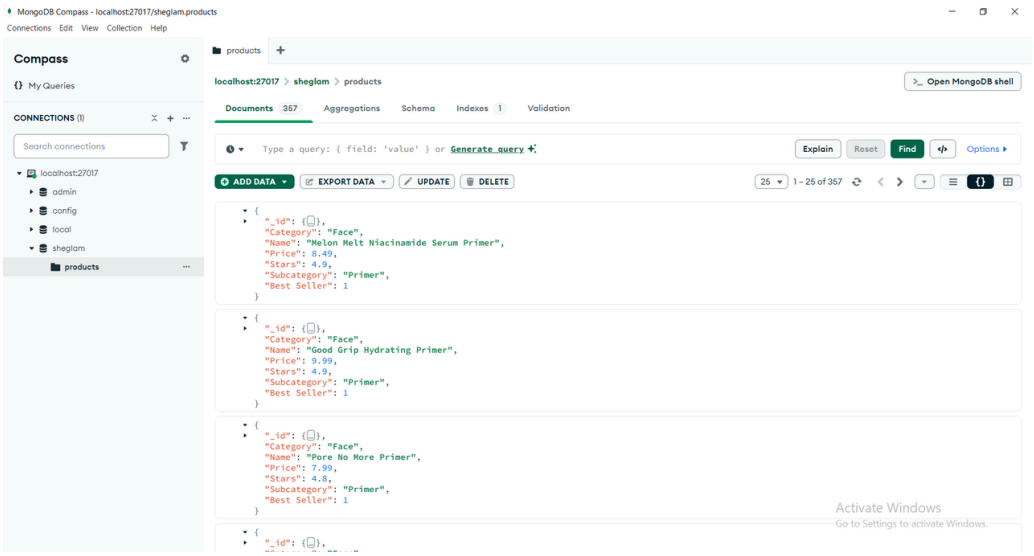
- We will notice that '**Late Night Lover**' and '**Girl Next Door**' from **Tools&Others** Category , '**Lip Facts Lip-Liner**' from **Lips** Category, have the **highest rating**.

🎀 Data Storage(MongoDB) Task 📦

13. Saving the final processed data in MongoDB:

- Code in our notebook connects to a local **MongoDB** server and accesses a database named **"sheglam"**.
  It then selects the **"products"** collection within the database.
- After that, it converts a DataFrame (`df`) into a list of dictionaries (records), and inserts all these records into the **"products"** collection at once using `insert_many()` .
- **Finally**, it prints a message confirming that the data has been successfully inserted into **MongoDB**.

Output:



Our Database

🎀 Bonus Task 🎁

📌 Streamlit Application Report: SHEGLAM Product Dashboard

🔷 Project Purpose

This dashboard was developed to provide interactive analysis and visual summaries of **SHEGLAM** products, focusing on pricing, ratings, best-seller trends, and special collections.

🔷 Tools Used

- **Python** (Pandas, Streamlit, Plotly, Seaborn, Matplotlib, WordCloud)
- **Streamlit** for interactive dashboard development

🔷 Dashboard Structure

The dashboard consists of **three tabs**:

- **Main Dashboard**
  - Summary statistics (Price & Stars)
  - Category/Subcategory averages
  - Best Seller counts
  - Top Rated Products (≥ 4.5 stars)
  - Filter by Category via sidebar
- **Visualizations**
  - Bar and Line plots for average Price & Stars by Category/Subcategory
  - Comparison between Best Sellers and Non-Best Sellers
  - Sunburst chart showing Best Sellers by Category
  - Category distribution (Bar chart)
  - Rating & Price distributions (Histograms)
  - Word cloud for common terms in product names
  - Top 10 Rated Products
- **Collections Analysis**
  - Bubble chart showing average price per collection
  - Bar chart comparing total products and best sellers across collections
  - Filtered to predefined themed collections (e.g., "Adventure Time", "Frida Kahlo")

🔷 Deployment

The app is deployed via Streamlit and the link was added.