Jilliann Divozzo and Cassidy McKenna
Professor Colleen van Lent
SI 206

**Makeup Mechanics Final Report: How Makeup Brand Rankings Relate to Product Prices**

Link to GitHub repository: https://github.com/jdivozzo/Final-project

A. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)

We originally wanted to find the average price of each category of makeup items for the best and worst ranked brands to see if there was a correlation between average price and brand ranking. We planned to use a makeup API: http://makeup-api.herokuapp.com/ and a website that ranked the top 100 makeup brands and gave details about sales revenue: https://wwd.com/lists/top-cosmetic-companies-2023-1236299225/grupo-boticario/.

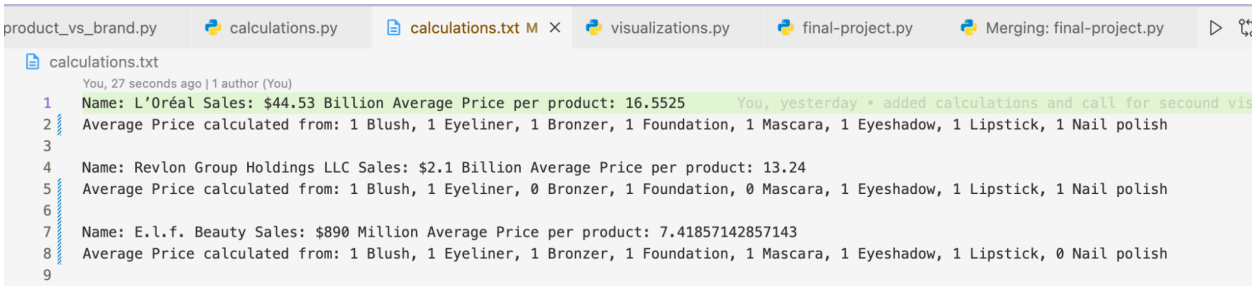
B. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)

After realizing we didn't have enough overlapping brands from the original API and website, we instead continued to grab makeup brands, their ranking, and product revenue from the website, while grabbing product data from the API that included product type and price for every brand that was compatible with the API, but we did pricing by product for only 3 brands rather than upwards of 50. This way we were able to use the data we had already collected to find prices of products per brand and select the overlapping brands to use in calculations and visualizations. Since we were only looking at 3 brands, we decided to widen the scope of our project goal to also include a comparison of brand revenue and average product price for each of the 3 overlapping brands.

C. The problems that you faced (10 points)

Towards the end of our project, we faced issues with inputting only 25 items to our database at a time. It was tricky to figure out how to confirm that 25 items were added so that our function wouldn't add more than 25, end up in a loop, or attempt to add the first 25 every time instead of moving onto the next set the next time the program would run. We ended up consulting with Professor van Lent and used a for loop with a counter to solve this issue in both makeup_api.py and makeup_brands_scrape.py.

D. The calculations from the data in the database (i.e. a screenshot) (10 points)




E. The visualization that you created (i.e. screenshot or image file) (10 points)

link to visualizations

F. Instructions for running your code (10 points)

If not already downloaded, Selenium will need to be available to run our code due to web scraping technicalities. All of our code is run in our final-project.py python file. In order to gather all of the data and have our visualizations accurately represented, our code should be run 4 times using the same brands.db database. To reset the database, delete "brands.db" that is created upon running. Everything should be run in the Final-project folder. You should expect to see one visualization at a time when the data has been entered to the database. It does take a while, probably 2-3 minutes per run.

To download Selenium:

- Open VS Code
- Go to Extensions (Ctrl+Shift+X)
- Search for "Selenium-Cucumber"
- Click Install

Screenshots of data from website and API being added into our database

G. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)

Table for each file (7 files): Name of function, input, output

| **final-project.py** | | |
|---|---|---|
| **Name of function** | **Input** | **Output** |

| | | |
|---|---|---|
| main | None | None |

| **makeup_brands_scrape.py** | | |
|---|---|---|
| **Name of function** | **Input** | **Output** |
| scrape_data | None | Data_list: (list of tuples with title: string, rank: string, sales: string) |

| **makeup_brands_database.py** | | |
|---|---|---|
| **Name of function** | **Input** | **Output** |
| create_table | None | None (creates scraped_data table and adds data to the table) |
| insert_data | data_list (list) | None (adds data to the table scraped_data title, rank, and sales) |

| **makeup_api.py** | | |
|---|---|---|
| **Name of function** | **Input** | **Output** |
| create_table_kinds_makeup | db_file, lst (database, list of makeup types) | None (creates and adds data to Makeup_types table) |
| create_brand_list | Fname (filename: string) | data (List of the brands that can be used with the API) |
| replace_vow | String, let, brand (string of characters to replace, character to use as replacement, brand name: | brand (string) |

| | | |
|---|---|---|
| | string) | |
| get_api_data | db_file, response, base_url, brand, data_list, brand_id (filename: string, response from initial api request, base url for API call: string, list of tuples with data to be inserted into the database so far, brand id that correlates with the brand name: int) | data_list (list of tuples where each tuple has a type id: int, brand id: int, brand name: string, and price: float) |
| get_makeup_data | db_file,names (database, list of brand names that work with API) | data_list (list of tuples where each tuple has a type id: int, brand id: int, brand name: string, and price: float) |
| create_table_makeup | db_file, data_list (database, list of tuples where each tuple has a type id: int, brand id: int, brand name: string, and price: float) | None (creates and adds data to Item_prices table) |

| **visualizations.py** | | |
|---|---|---|
| **Name of function** | **Input** | **Output** |
| get_data_from_db | None | data (scraped from database in list of tuples format) |
| parse_sale | sales (list) | sales (If sales, returns sales as a list in billions, else None) |
| clean_data | data (list of tuples) | cleaned_data (list of tuples: title(string), rank(int), sales(float)) |
| plot_data | data_list (list) | Fig graph: If data, creates bar graph visualization of the top 10 brands in the table, else prints “No data to |

| | | |
|---|---|---|
| | | plot” |
| plot_bottom_data | data_list (list) | Fig graph: If data, creates bar graph visualization of the bottom 10 brands in the table, else prints “No data to plot” |

| **visual_product_price_vs_brand.py** | | |
|---|---|---|
| **Name of function** | **Input** | **Output** |
| get_plot_data | db_file (database file: string) | table_data (dictionary with keys as product types and values as list of tuples with brand name: string, brand rank: int, and item price: float) |
| create_visual | Table_data (dictionary with keys as product types and values as list of tuples with brand name: string, brand rank: int, and item price: float) | None (creates visual, shows it and saves it as price_for_each_brand.png) |

| **calculations.py** | | |
|---|---|---|
| **Name of function** | **Input** | **Output** |
| get_data | db_file (database) | db_data (list of tuples with data from joining all three tables in the database) |
| calculate | db_data (list of tuples with data from joining all three tables in the database) | di_out (dictionary with keys as names of brands: string and values as tuples with sales: string, average price: float, and a dictionary with keys as makeup types and values as the amount of that |

| | | |
|---|---|---|
| | | product in the brand) |
| write_txt | di (dictionary with keys as names of brands: string and values as tuples with sales: string, average price: float, and a dictionary with keys as makeup types and values as the amount of that product in the brand) | None (writes data into calculations.txt) |

H. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

| Date | Issue Description | Location of Resource | Result (did it solve the issue?) |
|---|---|---|---|
| 4/15/2025 | We had trouble figuring out how to input data to our database 25 items at a time. | SI 206 lecture, Colleen | We were able to have Colleen assist us in this issue, she looked at our code and proposed a solution that involved creating a for loop and nested if statement. This did resolve this issue. |
| 4/6/2025 | Trouble scraping data from the website on Chrome | Stack overflow, Python.com | I researched how to use Selenium and WebDriver to scrape data from the website and integrate the data towards the bottom of the page requiring scrolling. This did resolve the issue, but I needed to download an extension. |
| 4/9/2025 | Issues with special characters | Stack overflow | We used VARCHAR 255 to integrate special characters into the database and allow them to be correctly read. Additionally, we swapped the special characters out for |

| | | | |
|---|---|---|---|
| | | | regular characters in the API due to inconsistencies with the website. |
| 4/14/25 | Issues with words in visualization being too close together | Stack overflow | I learned that I could use plt.tight_layout(pad=1.5) to add spacing to the bar chart. This solved the readability issue. |
| 4/9/25 | Wanted __pycache__ to not be tracked by git | GitHub Docs, previous homework starter code | I copied the .gitignore file from a previous homework assignment and also added files of my own that I didn't want git to track. This worked out great. |
| 4/14/25 | We wanted to work on the same file at the same time. | git-scm.com | I learned how to create a branch using git so we both weren't working in main at the same time. I also later researched how to merge the branch back into main. This made it easier to make changes at the same time. |