A logo of a university

Description automatically generated

**Signature Assignment - Final Project Research Paperwork (Group Assignment)**

**Decoding Market Dynamics: A Data-Driven Exploration of Shopping Patterns on Amazon in** **2023**​

**ALY6110 – Data Management and Big Data**

**CRN Number: 20486**

**Group Delta**

**Gayathri Komanduri  (**[komanduri.g@northeasten.edu](mailto:Komanduri.g@northeasten.edu" \t "_blank)**)**

**Keerthana Kandasamy (**[kandasamy.k@northeastern.edu](mailto:kandasamy.k@northeastern.edu) **)**

**Kiran Kumar Athimoolam (**[athimoolam.k@northeastern.edu](mailto:athimoolam.k@northeastern.edu)**)**

**Nagendra Hegde (**[hedge.na@northeastern.edu](mailto:Hedge.na@northeastern.edu)**)**

**Thiripura Ravi (**[ravi.th@northeastern.edu](mailto:Ravi.th@northeastern.edu)**)**

**Introduction:**

Amazon, the giant in online shopping, provides a treasure trove of data that holds key insights into consumer behavior and market trends. In this project, we're diving into this data using powerful tools like SQL and Python, alongside platforms like Databricks, to unravel important insights. Our focus is on understanding market trends, pricing fluctuations, and customer behavior by analyzing Amazon's product and category data. However, working with this vast amount of data is no walk in the park. We encounter challenges such as integrating diverse data sources and cleaning raw data to make it usable.

Our primary objective is to develop an efficient process, leveraging SQL and Python codes, to extract, transform, and load (ETL) the data. With this process in place, we can delve into the data to uncover valuable patterns and trends. For instance, we may discover seasonal variations in pricing or identify bestselling products in certain categories. This project holds significance for businesses aiming to make informed decisions. By harnessing the insights gleaned from our analysis, businesses can adapt their strategies to meet customer demands and stay ahead in the competitive market landscape. In essence, our goal is to translate raw Amazon data into actionable insights that drive business success.

**About the dataset:**

Amazon, a prominent figure in the U.S. online retail landscape, boasts a comprehensive dataset encompassing over 12 million products. This dataset presents an opportunity to glean meaningful insights, such as understanding top-selling products, determining SEO title effectiveness for driving sales, establishing optimal price ranges in specific categories, and uncovering other valuable information. The analysis aims to identify current popular product categories, evaluate their sales performance, pinpoint highly-rated products based on customer ratings, and create a product title generator using patterns from successful sales titles. The dataset is instrumental in discovering lucrative niches for potential sales, understanding overall spending patterns of online shoppers, and serves as a foundation for honing database management skills and optimizing performance.

Columns in the data source and their data types:

asin: Product ID from Amazon. (type:str)

title: Title of the product. (type:str)

imgUrl: Url of the product image. (type:str)

productURL: Url of the product. (type:str)

stars: Product rating. If 0, no ratings were found. (type:float)

reviews: Number of reviews. If 0, no reviews were found. (type:int)

price: Buy now price of the product. If 0, price was unavailable. (type:float, currency: USD)

listPrice: Original price of the product before discount. If 0, no list price was found AKA, no discounts. (type:float, currency: USD)

category\_id: Use the amazon\_categories.csv to find the actual category name. (type:int)

isBestSeller: Whether the product had the Amazon BestSeller status or not. (type:bool)

**Uploading the dataset to DBFS:**

* Amazon Products Dataset 2023 was obtained from the Kaggle which contained 2 files namely amazon\_produt and amazon\_categories.
* To upload the file into the system we select the File option from the top left corner of Data Bricks and then choose Upload data to DBFS option.

**A screenshot of a computer

Description automatically generated**

* Next Upload Data popup will come and there we can upload the files from the file system by selecting next and it will move to the Data Bricks notebook

**A screenshot of a computer

Description automatically generated**

**Code work with Data Bricks:**

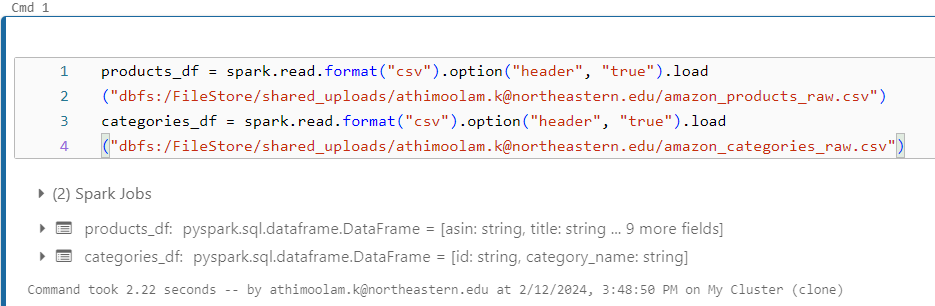
* Once the files are uploaded to work with the data or to create a table we need to first load the CSV files into a data frame.
* Here we created 2 data frames namely products\_df and categories\_df and loaded both files respectively.

products\_df = spark.read.format("csv").option("header", "true").load

("dbfs:/FileStore/shared\_uploads/athimoolam.k@northeastern.edu/amazon\_products\_raw.csv")

categories\_df = spark.read.format("csv").option("header", "true").load

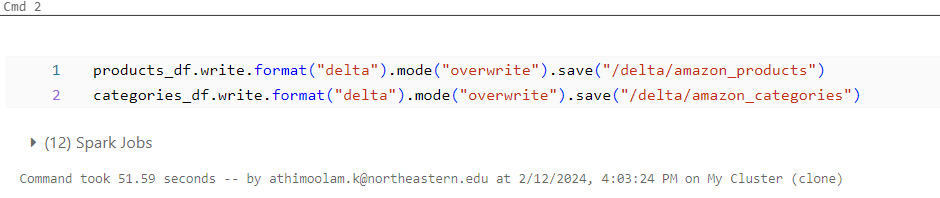
("dbfs:/FileStore/shared\_uploads/athimoolam.k@northeastern.edu/amazon\_categories\_raw.csv")

****

* Here we write the 2 data frames into Delta Lake format, with an overwrite mode, meaning that if any data exists in those locations, it will be replaced by the new DataFrames.
* The .save() method is used to specify the location where the DataFrames will be saved.

**products\_df.write.format("delta").mode("overwrite").save("/delta/amazon\_products")**

**categories\_df.write.format("delta").mode("overwrite").save("/delta/amazon\_categories")**

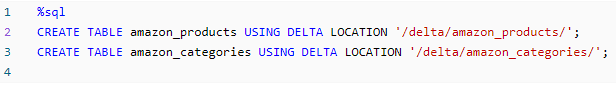
****

* Now we are creating Delta tables in Databricks using SQL commands.
* The CREATE TABLE statement with the USING DELTA syntax is used to create a Delta table, and the LOCATION parameter specifies the location where the Delta files are stored.

**%sql**

**CREATE TABLE amazon\_products USING DELTA LOCATION '/delta/amazon\_products/';**

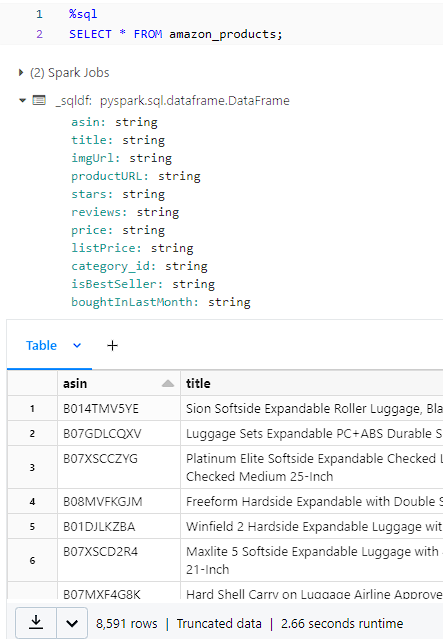
**CREATE TABLE amazon\_categories USING DELTA LOCATION '/delta/amazon\_categories/';**

****

* Once the table is loaded we need to make sure that all data is available hence we run the select statement on both the tables.
* We do the select query for both amazon\_products and amazon\_categories to see if the data is loaded correctly or not.
* Here we can see that all the data has been loaded correctly and the data type for each column is a string.
* As all the data type is string we need to make sure that they are converted in to proper data types before we make a final table.

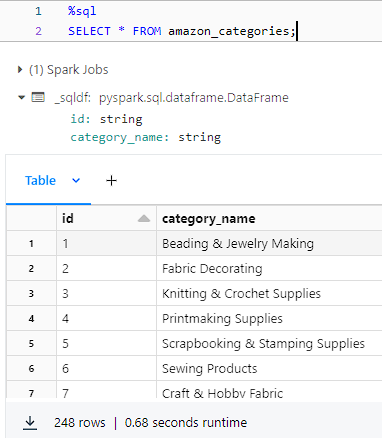
**%sql**

**SELECT \* FROM amazon\_products;**

****

**%sql**

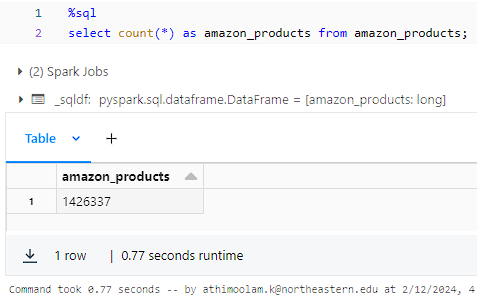
**SELECT \* FROM amazon\_categories;**

****

* To check the size or the number of rows present in the data we run a count query on the products table and categories table.
* In our case, amazon\_products has 1426337 number of records, and amazon\_categories has 248 records.

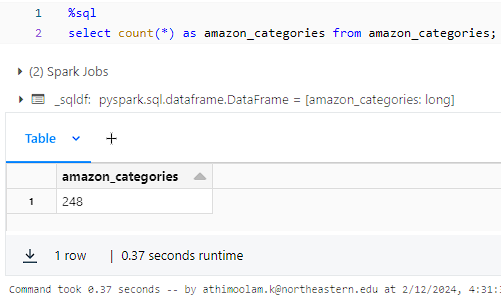
%sql

select count(\*) as amazon\_products from amazon\_products;

****

**%sql**

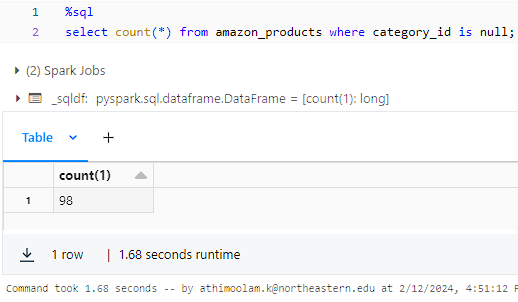
**select count(\*) as amazon\_categories from amazon\_categories;**

****

* When we inspect the data we find that the id column in the amazon\_cateogires table and category\_id in the amazon\_products table are the keys for joining the table.
* In Amazon categories id is the primary key and amazon\_products table category\_id is the foreign key for joining these 2 tables.
* Since these are the joining keys we need to check for junk values mainly for null values before joining.
* As we run the query to check the null values we find that there are 98 null values for the column category\_id hence these need to be cleaned up before joining both tables.

%sql

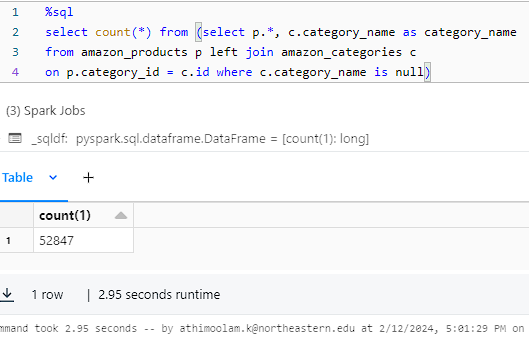
select count(\*) from amazon\_products where category\_id is null;

****

* Also, we need to check in the final table where we join the data from both tables will have any issues among the columns.
* The subquery performs a left join between the amazon\_products table and the amazon\_categories table on the category\_id column from amazon\_products and the id column from amazon\_categories. It selects all columns from amazon\_products and the category\_name column from amazon\_categories.
* The join condition p.category\_id = c.id links the amazon\_products table to the amazon\_categories table based on their respective id and category\_id columns.
* The `left join` ensures that all records from the amazon\_products table are included in the result set, even if there's no matching record in the amazon\_categories table.
* The where clause c.category\_name is null filters the result set to include only records where the category\_name from the amazon\_categories table is null, meaning there was no matching category found for the category\_id in the amazon\_products table.
* Finally, the outer query wraps this subquery and applies a count(\*) aggregation function, counting the number of records in the result set.
* This count represents the number of records in the amazon\_products table where there is no corresponding category name found in the amazon\_categories table.
* Here we find out that there are several null values in the category\_name column as well where the count stand at 52847.

%sql

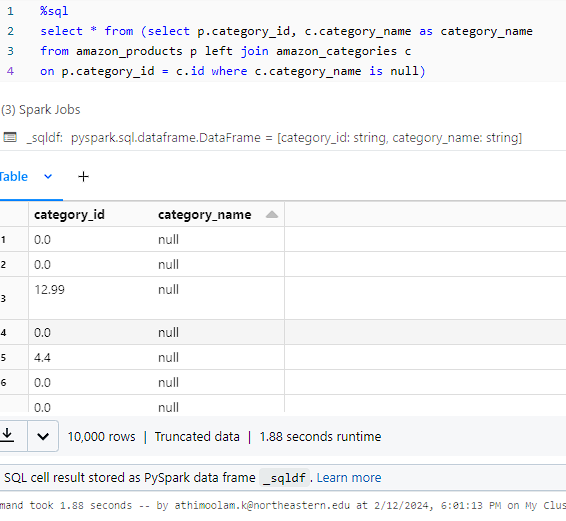
select count(\*) from (select p.\*, c.category\_name as category\_name from amazon\_products p left join amazon\_categories c on p.category\_id = c.id where c.category\_name is null)

****

* In the next query we are trying to see the data before deciding on the removal of the null valued data.
* Here instead of the count(\*) function we use the select function to display the data present in the table.
* As we can see the data here is not proper and they are junk values. Hence it is important that we either remove the junk values or fill them with relevant values.
* Since the junk values were not relevant it is better that we removed the data from the table to make sure it is clean.
* This removal can be done with final table loading.

%sql

select \* from (select p.category\_id, c.category\_name as category\_name from amazon\_products p left join amazon\_categories c on p.category\_id = c.id where c.category\_name is null)

****

* Once we understand the null values and junk values we also need to make sure that we only use the required columns.
* Here 2 columns namely imgUrl and productURL are not useful for our analysis as they contain the URL for the image and procut respectively.
* Hence while creating the final table with the below mentioned query we remove these columns to make the table as per the requirement.

In the next SQL query, it creates or replaces a table named amazon\_data. It populates this table by selecting specific columns from the amazon\_products table and joining it with the amazon\_categories table.

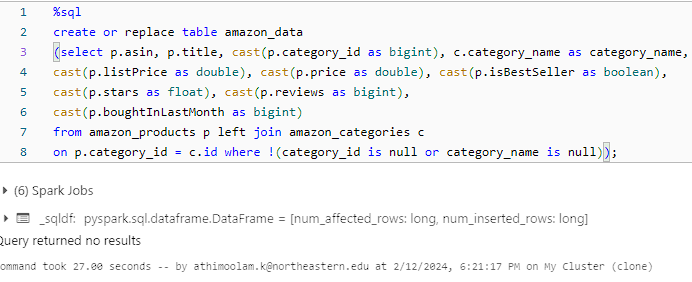
Here's a breakdown of what it does:

1. It selects columns asin, title, category\_id, listPrice, price, isBestSeller, stars, reviews, and boughtInLastMonth from the amazon\_products table (p alias).
2. It selects the category\_name column from the amazon\_categories table (c alias).
3. It performs a left join between amazon\_products and amazon\_categories based on the category\_id column.
4. It filters out rows where either category\_id or category\_name is null.
5. It casts certain columns to specific data types:
6. category\_id is cast to bigint.
7. listPrice and price are cast to double.
8. isBestSeller is cast to a boolean.
9. stars is cast to a float.
10. reviews and boughtInLastMonth are cast to bigint.

The resulting table amazon\_data will contain data from amazon\_products along with the corresponding category name from amazon\_categories, ensuring that only rows with valid category information are included and with the specified data types.

%sql

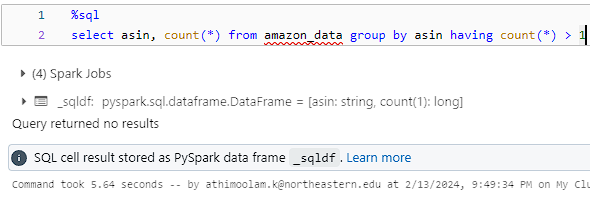
create or replace table amazon\_data(select p.asin, p.title, cast(p.category\_id as bigint), c.category\_name as category\_name,cast(p.listPrice as double), cast(p.price as double), cast(p.isBestSeller as boolean),cast(p.stars as float), cast(p.reviews as bigint),cast(p.boughtInLastMonth as bigint)from amazon\_products p left join amazon\_categories c on p.category\_id = c.id where !(category\_id is null or category\_name is null));

****

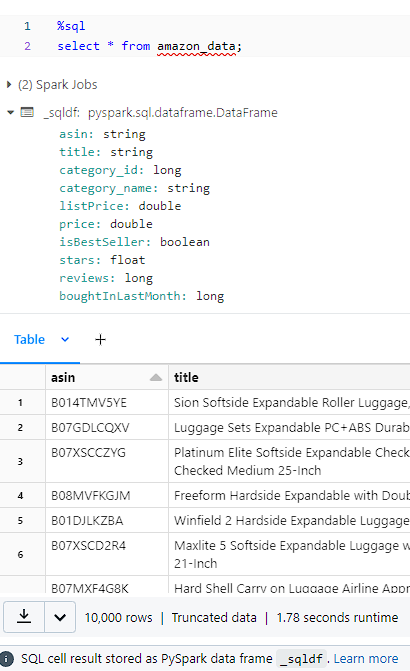
* Once the final table is created, we did a sanity check to identify if there are any duplicates with respect to the primary column ‘asin’.
* The screenshot below shows that there were no duplicate entries found in the data.

%sql

select asin, count(\*) from amazon\_data group by asin having count(\*) > 1



* Now once all the joining and data cleanup is done we have the final table with us names as amazon\_data.
* To check if the data is loaded properly we run the below query.
* As we can see that the data is loaded properly into the table hence we have the final table called amaon\_data with us for our analysis purposes.

****

**Analysis**

In this comprehensive analysis, we delve into various aspects of market dynamics, customer preferences, and demand analysis using three key visualizations: a correlation matrix, a bar graph, and a scatter plot. Each visualization provides unique insights into different facets of the business landscape, helping us understand market trends, pricing strategies, and consumer behavior.

**Correlation Matrix: (Market Trends and Pricing)**



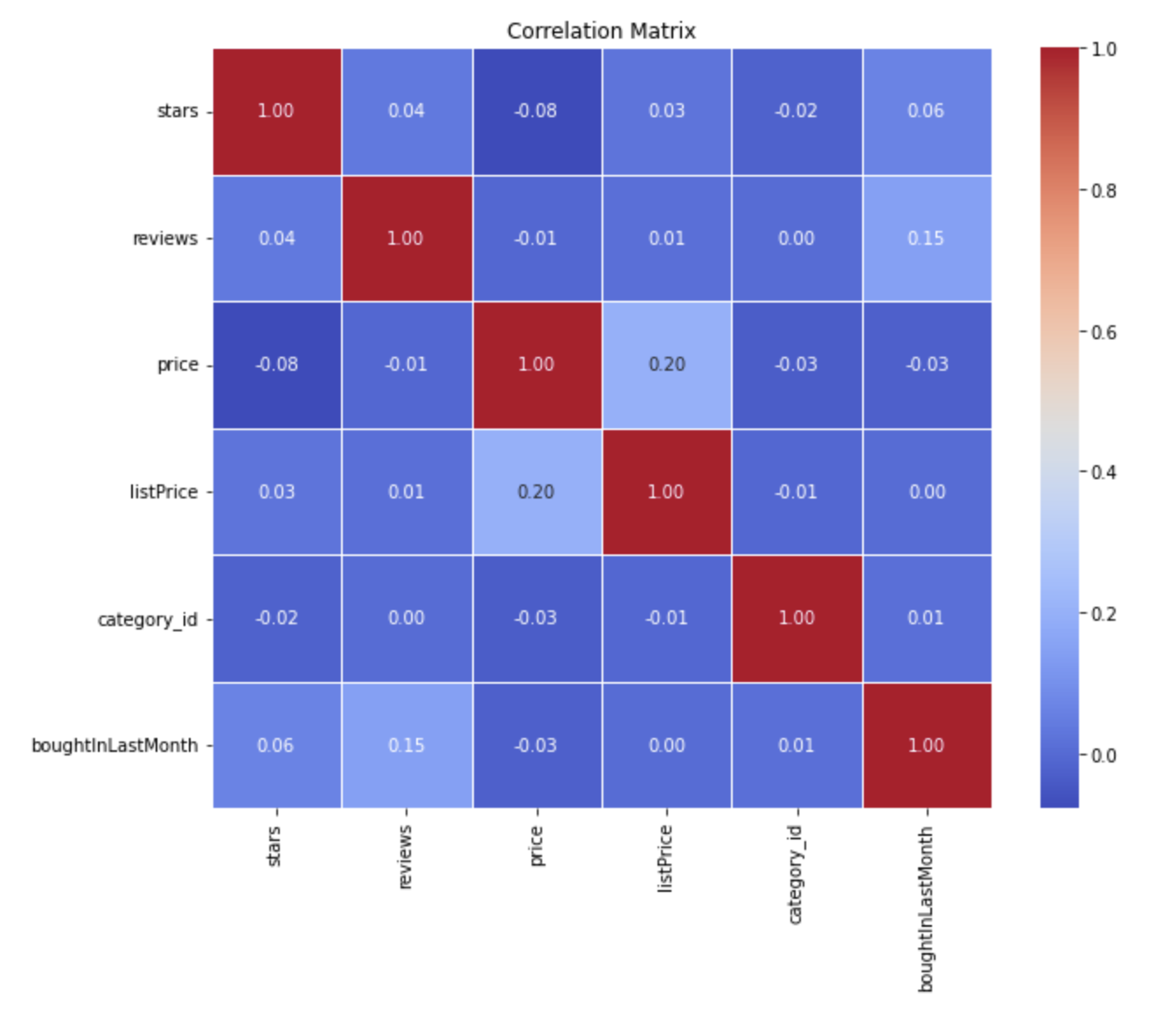
In the code snippet, we're conducting a correlation analysis on a dataset containing Amazon product information. We start by importing necessary libraries including NumPy, Pandas, Matplotlib, and Seaborn for data manipulation, analysis, and visualization.

Next, we specify the numerical columns of interest from the dataset that we want to include in the correlation analysis. These columns are 'stars', 'reviews', 'price', 'listPrice', 'category\_id', and 'boughtInLastMonth'. We then select these columns from the Amazon dataset, which is assumed to be stored as a DataFrame named 'amazon\_df'.

After selecting the relevant numerical columns, we proceed to create a correlation matrix using Pandas. The correlation matrix calculates the correlation coefficients between pairs of numerical variables. This matrix provides insights into the strength and direction of the linear relationship between these variables.

Once the correlation matrix is generated, we visualize it as a heatmap using Seaborn and Matplotlib. The heatmap colors each cell in the matrix based on the correlation coefficient, with warmer colors representing stronger positive correlations, cooler colors representing stronger negative correlations, and neutral colors representing weaker or no correlation. Additionally, we annotate the heatmap with the correlation coefficients for better interpretation.

Finally, we display the heatmap with the correlation matrix using Matplotlib, adding a title to provide context to the visualization. This visualization aids in identifying any significant correlations between the selected numerical variables, helping us understand potential relationships within the dataset.



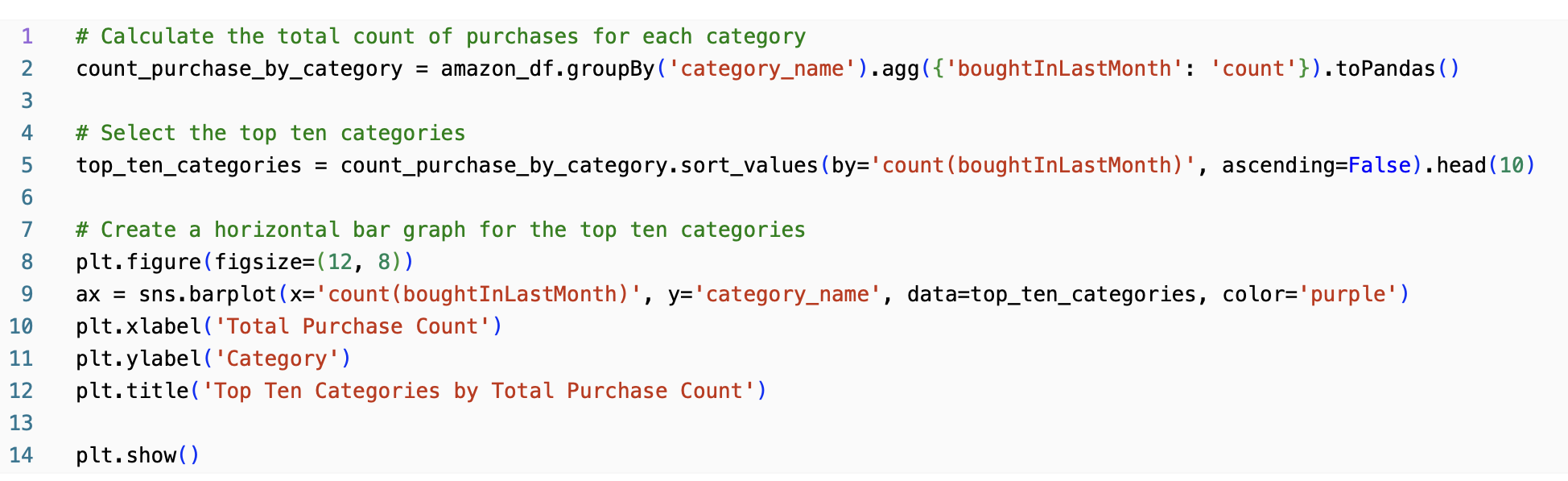
The correlation matrix allows us to examine the relationships between different numeric variables such as pricing, ratings, purchases and reviews. By assessing correlations between these factors, we gain valuable insights into how pricing strategies align with market trends and impact overall business success.

The correlation coefficient between list price and price of 0.20 suggests a positive but relatively weak correlation. This indicates that there is some tendency for list price and price to move together, meaning that as the list price increases, the actual price tends to increase as well, and vice versa. However, the strength of this relationship is not very strong, meaning there are likely other factors influencing the pricing strategy besides just the list price.

On the other hand, the correlation coefficient of 0.15 between boughtInLastMonth and reviews suggests a slightly stronger positive correlation. This implies that there is a moderate tendency for products that have been bought more frequently in the last month to also have higher numbers of reviews.

This correlation could be explained by the fact that products that are bought more frequently tend to attract more attention from customers, leading to a higher likelihood of customers leaving reviews. Additionally, products with more reviews may also appear more trustworthy or popular to potential buyers, further driving their sales.

**Bar Graph: (Customer Preferences)**



From this code, we're analyzing the total count of purchases for each product category within the Amazon dataset.

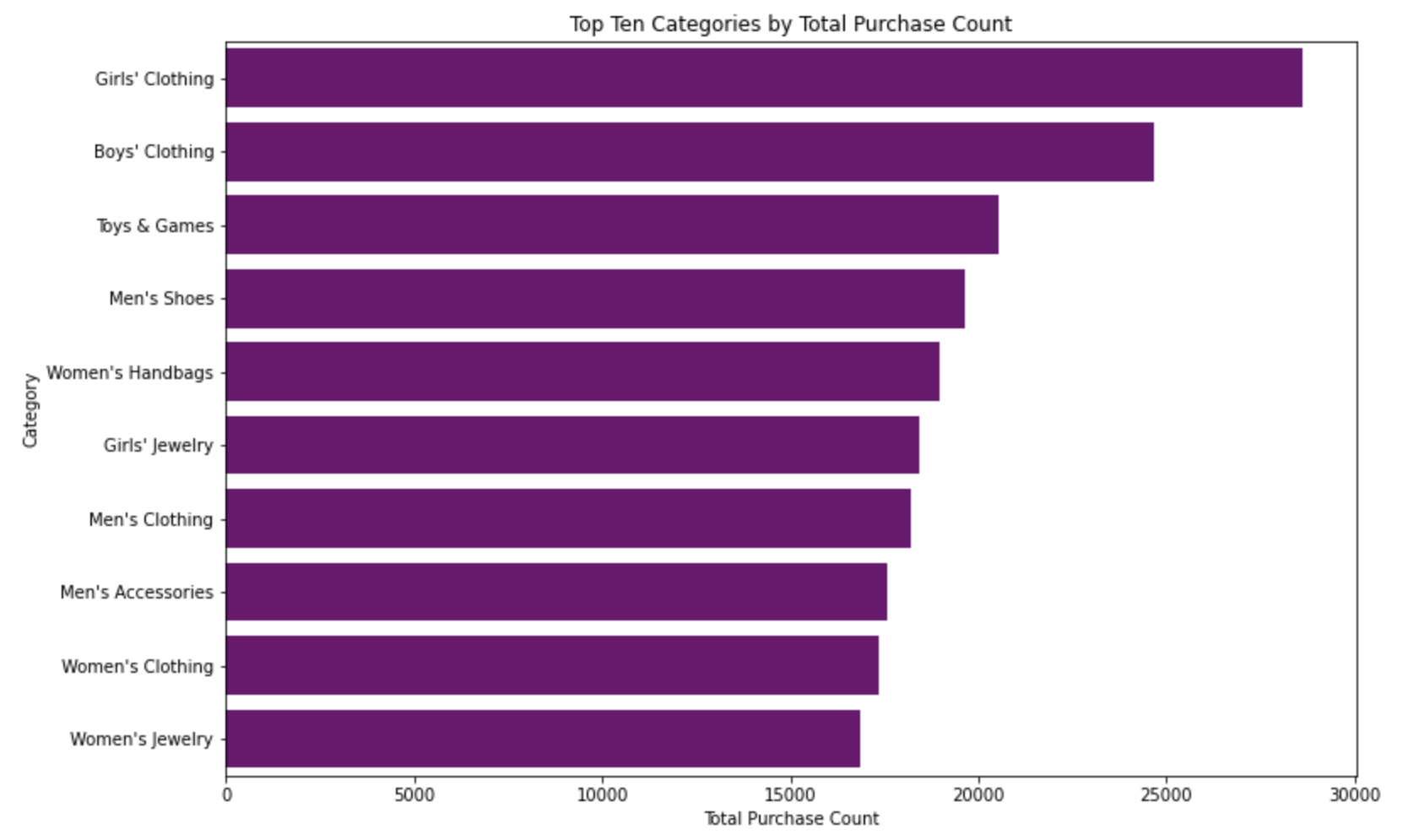
First, we group the dataset by the 'category\_name' column and calculate the count of purchases ('boughtInLastMonth') for each category. This aggregation is performed using PySpark's groupBy and agg functions, resulting in a Pandas DataFrame containing the total purchase count for each category.

Next, we select the top ten categories based on their total purchase counts. This is achieved by sorting the DataFrame in descending order based on the purchase count and then selecting the first ten rows.

We then create a horizontal bar graph to visualize the top ten categories by their total purchase counts. Seaborn's barplot function is used for this purpose, with the 'count(boughtInLastMonth)' column representing the x-axis (total purchase count) and the 'category\_name' column representing the y-axis (category). We set the color of the bars to purple for visual appeal.

Additional customization is applied to the plot, including labeling the x-axis as 'Total Purchase Count', the y-axis as 'Category', and providing a title for the plot ('Top Ten Categories by Total Purchase Count').

Finally, we display the bar graph using Matplotlib, showcasing the relative popularity of each category based on their purchase counts. This visualization aids in identifying the most sought-after product categories on Amazon, providing valuable insights into customer preferences and market trends.



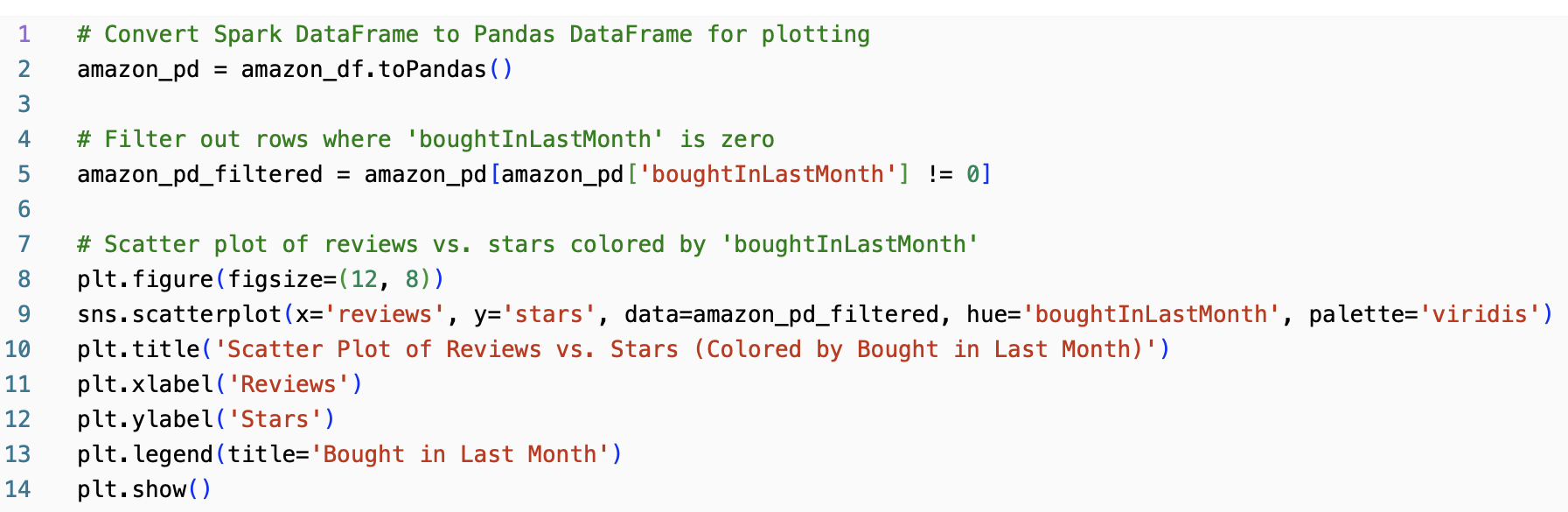
The bar graph provides a visual representation of customer preferences across various product categories. By analyzing the total purchase count within each category, we can identify dominant market segments, understand consumer buying patterns, and tailor marketing strategies to better meet customer demands.

The bar graph depicting the top ten categories by total purchase count on Amazon reveals several compelling trends. Notably, children's clothing emerges as a dominant force, with categories like Girls' Clothing, Boys' Clothing, and Toys & Games occupying the top spots. This indicates a significant preference for these items among Amazon shoppers, possibly driven by the constant need to update children's wardrobes and provide entertainment.

Furthermore, the popularity of women's and men's fashion is evident, with several fashion-related categories making the list, such as Men's Shoes, Women's Handbags, Men's Clothing, Women's Clothing, and Women's Jewelry. This underscores the diverse shopping habits of Amazon customers, highlighting a penchant for both practical and stylish apparel and accessories.

Interestingly, the graph also reveals a clear distinction between children's and adult categories, with a noticeable gap in purchase counts. This suggests distinct shopping behaviors and priorities between shoppers seeking items for themselves versus those purchasing for children. Additionally, the relatively similar purchase counts for certain adult categories like Men's Shoes, Women's Handbags, and Girls' Jewelry indicate comparable levels of popularity, hinting at potential areas for further market analysis and strategic targeting.

**Scatter Plot: (Demand Analysis)**



In this code snippet, we're creating a scatter plot to analyze the relationship between product reviews and star ratings, with an additional distinction based on whether the product was bought in the last month.

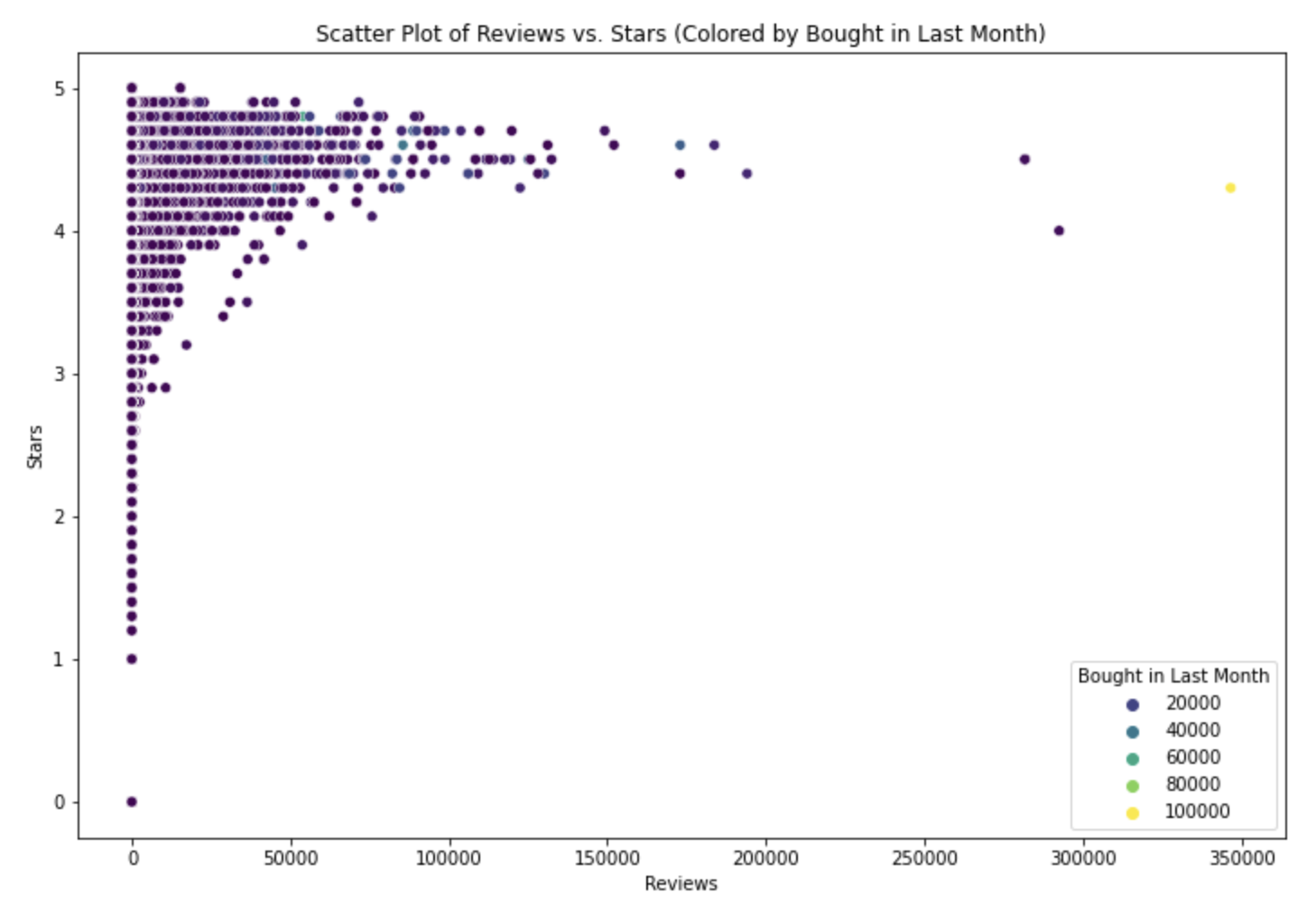
First, we convert the Spark DataFrame 'amazon\_df' to a Pandas DataFrame named 'amazon\_pd' to facilitate data manipulation and visualization using Pandas and Matplotlib.

Next, we filter out rows where the 'boughtInLastMonth' column is zero, indicating products that were not purchased in the last month. This step ensures that we focus only on products that have recent purchase activity.

We then proceed to create the scatter plot using Seaborn's scatterplot function. The 'reviews' column is represented on the x-axis, while the 'stars' column is represented on the y-axis. Each data point in the scatter plot corresponds to a product, with its position determined by the number of reviews and star rating. Additionally, the points are colored based on the 'boughtInLastMonth' column, with different colors representing products bought in the last month and those that were not.

Further customization is applied to the plot, including setting the title ('Scatter Plot of Reviews vs. Stars (Colored by Bought in Last Month)'), labeling the x-axis ('Reviews'), and labeling the y-axis ('Stars'). Additionally, a legend is added to provide clarity on the color coding, indicating the product bought in the last month.

Finally, we display the scatter plot using Matplotlib, allowing us to visually inspect any patterns or trends in the relationship between reviews, star ratings, and recent purchase activity. This visualization aids in understanding customer sentiment, the impact of recent purchases on reviews, and potential outliers in the data.



The scatter plot offers a nuanced view of demand dynamics by illustrating the relationship between product reviews, star ratings, and recent purchases. By examining the clustering of data points and any discernible patterns, we can uncover insights into customer satisfaction levels, the impact of recent purchases on reviews, and potential outliers that warrant further investigation.

Upon analyzing the scatter plot, it becomes apparent that there's a clear relationship between product reviews and star ratings. The clustering of data points in the upper right quadrant suggests an overall positive sentiment, with most products receiving higher star ratings accompanied by a significant number of reviews. This indicates that customers are generally satisfied with these products and are motivated to leave feedback about their experiences.

The distinction based on whether the product was bought in the last month adds an interesting layer to the analysis. Points are colored blue or orange, with blue likely representing the quantity of a purchased product. While there's a slight concentration of blue points in the upper right quadrant, suggesting that recently bought products might tend to have slightly higher average reviews and counts, the overlap between blue and orange points is considerable.

**Pros and Cons of ETL Process:**

**Pros:**

* **Data Quality and Consistency:** ETL allows for thorough cleaning and transformation of data before it enters the target system, ensuring accuracy and consistency. This enhances data-driven decision making.
* **Flexibility and Control:** ETL offers greater control over data manipulation, allowing for complex transformations and tailoring to specific needs. This is beneficial for data warehousing environments.
* **Performance and Efficiency:** By transforming data before loading, ETL can optimize storage and improve query performance in the target system.
* **Security and Compliance:** ETL allows for implementing data security measures and ensuring compliance with data regulations.
* **Proven Technology:** ETL is a well-established technology with many mature tools and experienced professionals available.

**Cons:**

* **Development Time and Cost:** Designing and implementing ETL pipelines can be complex and time-consuming, requiring skilled professionals and potentially expensive tools.
* **Inflexibility and Latency:** Changes to data sources or transformations require modifying the entire pipeline, leading to delays and reduced adaptability. This can be a challenge for agile environments.
* **Limited Scalability:** Traditional ETL processes can struggle with large, high-volume data sets, impacting performance and scalability.
* **Operational Complexity:** Managing and maintaining ETL pipelines can be intricate, requiring constant monitoring and error correction.

**Difficulties and issues:**

1. **Null Values:** To be able to join the 2 tables we needed the column category\_id to be not null. But when we checked the product data we could find there were some null values. hence before joining we had to clean these null values so that there were no junk values in the final table.
2. **Data cleanup:** We had to create a temporary table to analyze the category\_name data column and make sure that it is not null. Once we created the temporary table we found the null values and rechecked the catgory\_id column to see if it had any data but there were some junk values. Hence as the category\_name and category\_id both did not make sense we had to decide to drop this data.

**Pros, Cons, & Hurdles Of Analysis**

**Pros:**

1. Comprehensive Analysis: By employing multiple visualization techniques such as correlation matrices, bar graphs, and scatter plots, we gain a holistic understanding of various aspects of the business landscape including market trends, customer preferences, and demand dynamics.

2. Insights Generation: Each visualization provides unique insights that complement and reinforce each other, enabling us to uncover hidden patterns, relationships, and trends within the dataset.

3. Effective Communication: Visualizations offer an intuitive and visually appealing way to communicate complex findings and insights to stakeholders, facilitating better decision-making and strategy formulation.

4. Code Reusability: The code snippets provided are modular and reusable, allowing for easy adaptation and extension to analyze similar datasets or explore different aspects of the business.

**Cons:**

1. Data Limitations: The analysis is contingent upon the quality and completeness of the dataset. Incomplete or inaccurate data can lead to skewed insights and erroneous conclusions.

2. Interpretation Challenges: While visualizations aid in data exploration and interpretation, they can sometimes be subject to misinterpretation or ambiguity, especially if proper context is not provided or if underlying assumptions are not adequately addressed.

3. Scalability Issues: For larger datasets, processing and analyzing data using certain techniques such as correlation matrices may become computationally intensive and time-consuming, potentially limiting scalability.

**Challenges:**

1. Code Migration Issues: Transitioning from Python Jupyter Notebook to Databricks posed challenges due to differences in the underlying environment and data processing capabilities. Attempting to replicate the existing code in Databricks resulted in numerous errors and inconsistencies, primarily stemming from discrepancies in data cleaning methodologies between the two platforms. Additionally, integrating existing code and workflows into the Databricks platform proved to be more complex than anticipated, requiring significant adjustments and troubleshooting efforts.

2. Data Cleaning Discrepancies: The data cleaning processes implemented in Python Jupyter Notebook are not the same as what we implemented in Databricks, leading to unexpected outcomes and errors during code execution. Variances in data formats, handling of missing values, and preprocessing techniques between the two environments contributed to the difficulties in achieving consistent results. Data preprocessing, including cleaning and preparing the dataset for analysis, was particularly time-consuming and labor-intensive.

3. Integration Complexity: Aligning data processing steps, dependencies, and libraries across different platforms added layers of complexity to the migration process. The intricacies of integrating existing code and workflows into the Databricks platform required a steep learning curve and adaptation period for the team. Acquiring familiarity with Databricks-specific functionalities, syntax, and best practices presented a significant hurdle in achieving seamless integration and execution.

4. Interpretation Ambiguity: Interpreting the results of the analysis, especially in the context of real-world business implications, may involve subjective judgment and uncertainty. Addressing interpretation ambiguities required an iterative problem-solving approach, with experimentation, debugging, and collaboration essential in refining the code and data processing pipelines to align with the requirements and capabilities of the Databricks platform.

**Key Strategies for Effective Analysis**

1. Leveraging Python Libraries: Making use of widely-used Python libraries like Pandas, NumPy, Matplotlib, and Seaborn streamlines the process of handling data, performing various analyses, and creating insightful visualizations. These libraries offer powerful tools and functions that expedite data manipulation and enhance visualization capabilities.

2. Thorough Documentation: Providing comprehensive documentation alongside the analysis code enhances transparency and reproducibility. Detailed explanations and comments offer insights into the analysis process, making it easier for team members to understand, replicate, and build upon previous work. Clear documentation also aids in troubleshooting and maintaining the analysis pipeline over time.

3. Modular Code Design: Breaking down the analysis into modular code snippets promotes code reusability, simplifies maintenance, and facilitates collaboration among team members. Modular design allows different components of the analysis to be developed, tested, and modified independently, resulting in more manageable and scalable codebases.

4. Best Practices in Visualization: Adhering to best practices in visualization design ensures that visualizations are effective in conveying insights. Proper labeling, color coding, and annotation help to clarify complex information and highlight key findings. Following visualization best practices enhances the interpretability and impact of the visual representations of data analysis results.

5. Iterative Approach to Analysis: Adopting an iterative approach to analysis involves continuously refining and improving the analysis based on feedback and new insights. By incorporating feedback from stakeholders and revisiting analysis assumptions, teams can enhance the accuracy and relevance of their findings. Iterative analysis allows for the exploration of alternative approaches and the adaptation of the analysis to evolving requirements, leading to more robust and actionable insights.

**Conclusion**

In this project, we embarked on a detailed exploration of Amazon's vast dataset, aiming to extract meaningful insights that could inform strategic decision-making. Leveraging a combination of SQL, Python, and visualization tools within the Databricks platform, we navigated through the complexities of data integration, cleaning, and analysis to derive actionable conclusions.

Our analysis journey began with the establishment of an efficient ETL process, laying the groundwork for subsequent investigations. Through meticulous examination of market trends, pricing dynamics, and customer preferences, we uncovered valuable patterns and relationships within the data.

Key findings emerged from our analysis, shedding light on various aspects of Amazon's business landscape. From correlations between pricing strategies and customer reviews to insights into popular product categories and demand dynamics, each visualization provided unique perspectives that contributed to a holistic understanding of the dataset.

Despite encountering challenges such as data preprocessing complexities and interpretation ambiguities, our team successfully navigated through these hurdles, ensuring the integrity and reliability of our findings. By adhering to best practices and adopting an iterative approach, we refined our analysis, ensuring that our conclusions were robust and actionable.

In conclusion, this project underscores the importance of data-driven decision-making in today's competitive market environment. By harnessing the power of data analytics, businesses can gain valuable insights that drive innovation, optimize operations, and ultimately, achieve strategic success. Our findings serve as a testament to the transformative potential of data analytics in unlocking new opportunities and driving sustainable growth.

References:

Shah, H. (2023, March 13). *The Pros and cons of ETL and ELT for data warehousing*. Medium.<https://hardiks.medium.com/the-pros-and-cons-of-etl-and-elt-for-data-warehousing-1b22e565614d>

Trotino, G. (2023, September 21). *What are ETL tools? advantages, disadvantages and Innovations revealed*. Delivering Data Products in a Data Fabric & Data Mesh.<https://www.k2view.com/blog/what-are-etl-tools>

*Amazon Products Dataset 2023 (1.4M Products). (2024, February 14). Kaggle.* [*https://www.kaggle.com/datasets/asaniczka/amazon-products-dataset-2023-1-4m-products*](https://www.kaggle.com/datasets/asaniczka/amazon-products-dataset-2023-1-4m-products)

*D. (2021, August 31). Learn to Use Databricks for Data Science.*

*YouTube. https://www.youtube.com/watch?v=mvezlGqmpYc*