**Advance Database Technologies**

**Final Project Part 2 - Database Design**

**Spring 2024**

**Project Title:**

**Amazon: A2Z AmazeComparator**

**Team Names:**

* Kabir Chaturvedi (kschatur@iu.edu ) (Team Leader)
* Dhyey Joshi (dhyjoshi@iu.edu )
* Vedika Halwasiya (vhalwasi@iu.edu )

**DATABASE DESIGN**

**Introduction**

This document outlines the database design of the project titled "Amazon: A2Z AmazeComparator," which aims to create a digital marketplace with a sophisticated yet user-friendly interface. The software is based on neat, well-structured data that not only facilitates precise operations but also enhances accuracy and efficiency. It is constructed on meticulously cleansed and formatted data, ensuring both reliability and performance. Our platform enables customers to easily compare products and browse consumer reviews, shedding light on the vast array of options available on Amazon.

The essence of A2Z AmazeComparator lies in its intelligent features, such as exact product comparisons and intuitively integrated review analysis. These features streamline the shopping journey, empowering users to make well-informed decisions with ease. Consequently, the platform not only saves users' time but also elevates the shopping experience, thanks to readily accessible, relevant information at their fingertips.

**Conceptual Diagram/Schema**

Our platform is grounded in an SQL-based schema, detailed through an Entity-Relationship Diagram (ERD) that crystallizes the relationships between Products, Users, Reviews, Categories, and Product Categories. The ERD is the blueprint of our database, depicting the interconnectivity essential for enabling comprehensive product comparisons and deep analytical dives into consumer sentiment.

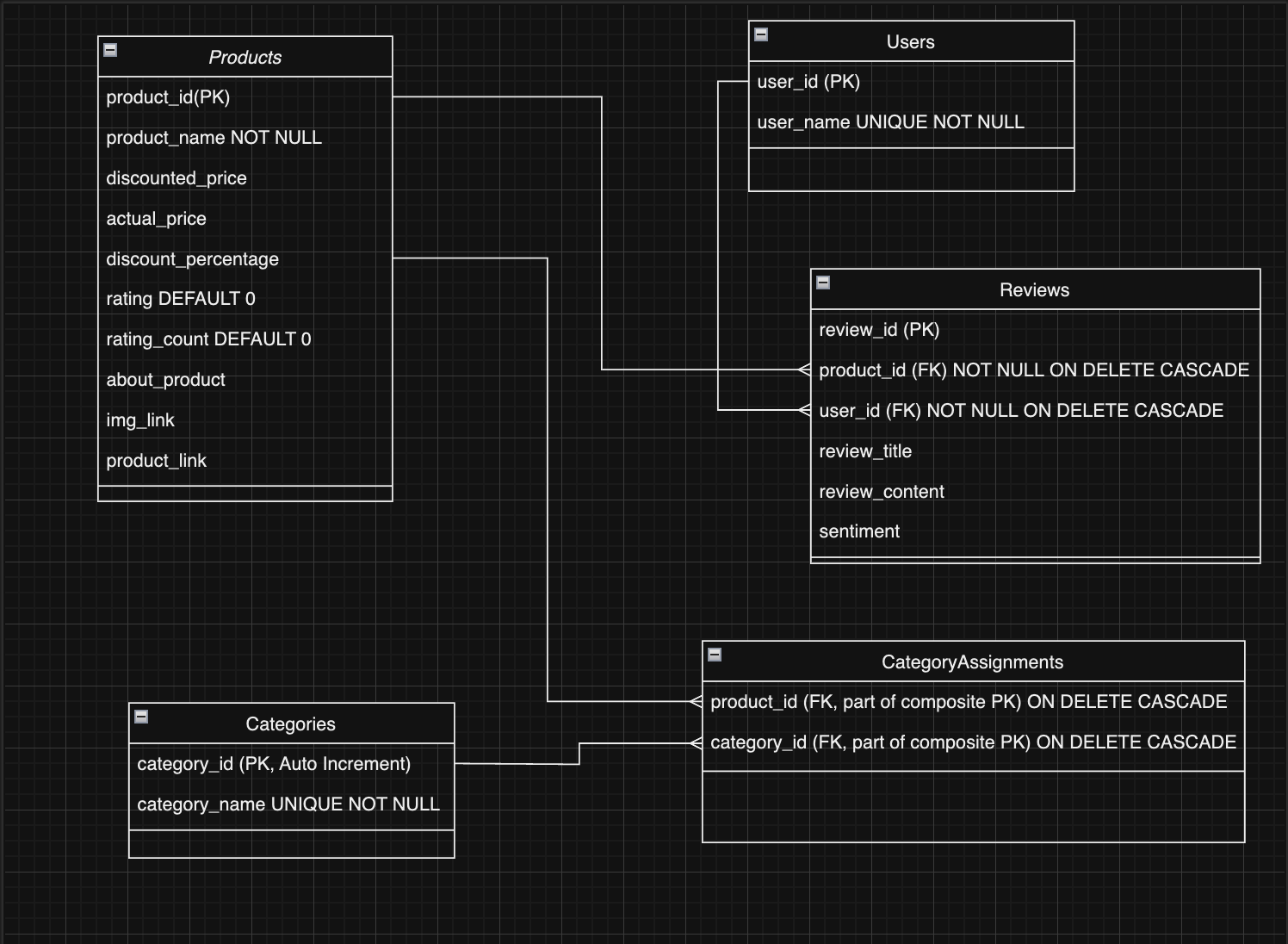


Figure 1: Amazon: A2Z AmazeComparator - Database Structure (ERD Diagram)

**Database Structure and Entities**

The database is comprised of several integral entities, each meticulously defined to encapsulate vital aspects of the Amazon shopping experience:

* **Products**: Central to the platform, this table catalogs every product's identity, including a unique product\_id, characteristics, pricing, and ratings, also linking to visual and descriptive content for an all-encompassing product profile.
* **Users**: A repository for user information, crucial for understanding and tailoring the user journey on the platform, ensuring a personalized and engaging experience. Contains user-specific data, identified by a unique user\_id, including usernames. This setup facilitates the tracking of user interactions and reviews on the platform.
* **Reviews**: This entity not only records reviews but also serves as the foundation for sentiment analysis, tying feedback to specific products and users for targeted insights. It is linked through foreign keys to both the Products and Users entities.
* **Categories**: Functions as the categorical organization system, facilitating efficient navigation and comparison within product sectors, enhancing the user's search experience by organizing products into specific categories.
* **CategoryAssignments**: A many-to-many link, this table reflects the multifaceted nature of products that may span across various categories, affording a more dynamic and interconnected product classification.

In our database, each table is meticulously crafted with various constraints to uphold data integrity and ensure accuracy: Primary Key Constraints are utilized to uniquely identify each record, which helps in avoiding duplicates and enables reliable cross-referencing between tables; Foreign Key Constraints are established to define clear relationships among tables, thus maintaining referential integrity and allowing for complex queries that involve multiple entities; Not Null Constraints, along with Unique Constraints, are applied to critical fields to ensure that essential data such as product links and usernames are consistently present and uniquely identified, safeguarding the database against erroneous or incomplete entries.

**DATABASE CONSTRAINTS**

The design of our database amazon\_sales\_project incorporates several constraints to ensure the integrity, accuracy, and reliability of the data. These constraints are vital for maintaining the structure of the database and enforcing rules on the data stored within it.

**Products Table:**

*Primary Key Constraint:* product\_id serves as a unique identifier for each product entry, ensuring no two products have the same ID.

*NOT NULL Constraint:* The product\_name field is required, meaning every product entry must have a name specified.

*CHECK Constraint on rating:* This constraint ensures the rating value falls between 0 and 5, inclusive, which standardizes rating values to a predefined range.

*Default Values:* Fields rating and rating\_count have default values set to 0, ensuring that new records start with a baseline value if not explicitly provided.

**Users Table:**

*Primary Key Constraint:* user\_id is designated as the primary key, guaranteeing uniqueness across user records.

*NOT NULL Constraint:* The user\_name field must have a value, ensuring every user has a name

**Reviews Table:**

*Primary Key Constraint:* review\_id is the primary key, ensuring uniqueness for each review.

*Foreign Key Constraints:* product\_id and user\_id serve as foreign keys, linking reviews to their respective products and users. This maintains referential integrity within the database.

*NOT NULL Constraints:* Both product\_id and user\_id are required to be non-null, guaranteeing every review is linked to a specific product and user.

*UNIQUE Constraint:* The combination of product\_id and user\_id must be unique, which implies a user can leave only one review per product.

*ON DELETE CASCADE:* This option is applied to foreign keys product\_id and user\_id, meaning if a product or user is deleted, all associated reviews will also be automatically removed from the database.

**Categories Table:**

*Primary Key Constraint:* category\_id is an auto-incrementing primary key, ensuring each category receives a unique identifier automatically.

*UNIQUE Constraint on category\_name:* Guarantees that each category name is unique across the table.

**CategoryAssignments Table:**

*Primary Key Constraint:* A composite key of (product\_id, category\_id) is used, ensuring each product-category pairing is unique.

*Foreign Key Constraints:* Both product\_id and category\_id are foreign keys that link back to the Products and Categories tables, respectively.

*ON DELETE CASCADE:* Applied to both foreign keys, product\_id and category\_id, this ensures that if a product or category is deleted, all related assignments in the CategoryAssignments table are also automatically deleted.

**DATA PREPARATION**

**Dataset Collection**

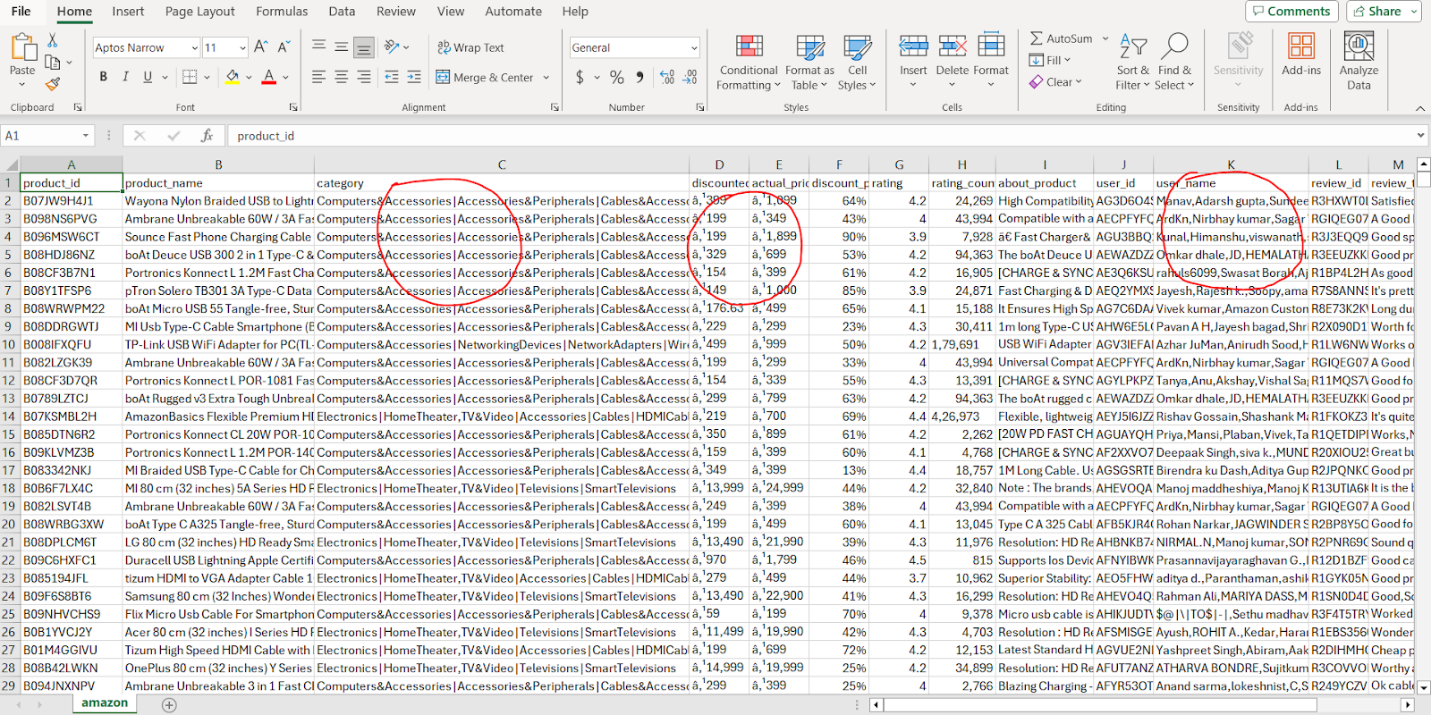
For this project, we have chosen to utilize a comprehensive dataset from Kagle titled [“Amazon Sales Data”](https://www.kaggle.com/datasets/bugraorhan/amazon-sales-data/data). This dataset includes a detailed record of product listings and customer reviews across various categories on Amazon, featuring 1465 entries with 16 distinct attributes such as product name, category, price, discount rate, rating, and user review.

This dataset enables us to extract and compare key product features, pricing, ratings, and sentiment analysis of reviews – essential components for our “A2Z AmazeComparator”. It is specifically curated to assist users in navigating the extensive product options on Amazon, simplifying the decision-making process by providing a side-by-side comparison of products.

This aligns perfectly with our project’s aim to enhance the shopping experience by offering a user-friendly platform where important product information is readily accessible, thereby saving time and effort in finding the best deals on Amazon.

***Our Amazon Sales Dataset: RAW & Require Normalization & Pre-Processing***

<https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset/data>



**Data Pre-Processing**

1. **Data Cleaning**: We began the data cleaning process focusing on identifying missing or null values across columns, especially in key fields like ***product\_id****,* ***product\_name****,* ***discounted\_price****,* ***actual\_price****,* ***rating****,* and ***rating\_count***. These columns are crucial for comparative analysis and insights into product popularity and customer satisfaction, necessitating accuracy and completeness. We also searched for duplicate entries to prevent biases, particularly in ***product\_id fields***, to avoid distorted product performance comparisons. Formatting adjustments were made, such as converting price fields to numeric values and standardizing ***rating\_count***. These steps ensured the dataset was prepared for further analysis and modeling, establishing a clean and accurate foundation.

2. **Normalizing Text & Numeric Field**: During the normalization step of our preprocess, we worked on making the data uniform by standardizing the fields used and making it easy for analysis. To begin, we treated all the textual identifiers, such as ***product\_id, user\_id,*** and ***review\_id,*** as strings because they are unique identifiers, and are not standard numbers. For the price columns, we removed the currency symbols (‘₹’) for the ***discounted\_price*** and ***actual\_price fields,*** removed the commas, and converted them into floats to ease numerical analysis. For the ***rating*** column, we converted it to a float and addressed a few non-numeric ratings by coercing them to maintain the dataset’s integrity. Moreover, we converted the ***discount\_percentage*** and removed its percent (‘%’) sign, transforming it into a float to simplify percentage calculation in our processes. Finally, we removed the commas from ***rating\_count*** used as thousands while converting it to an integer to accurately reflect the number of reviews that the product had. This process helped us streamline our data and gear up for our processes while maintaining consistency in data.

3. **Parse and Expand Nested Data**: In our comprehensive approach to parsing and expanding nested data within our dataset, we tackled fields laden with multiple values—***user\_id****,* ***user\_name****,* ***review\_id****,* ***review\_title****,* and***review\_content***. Recognizing these comma-separated lists' depth of information, we separated these values to form individual records for each, thus enabling a refined analysis of user interactions with products. This separation significantly enhanced our data's granularity, facilitating detailed sentiment analysis and pattern recognition in reviews. For instance, by separating ***user\_id*** and corresponding ***review\_id***, we facilitated a structured approach to link each review with its specific product and reviewer, thereby enhancing our ability to conduct detailed sentiment analysis and review pattern identification.

Further enriching our dataset, we addressed the complex structure of the **category** field. Utilizing Python's **literal\_eval** from the **ast** module, we transformed the **category** entries from string representations of lists into actual list-objects. This pivotal step allowed us to dissect the multi-layered category information into actionable insights. We then extracted unique categories, broadening our analytical scope to include category-based analyses. By iterating over each row, we mapped products to their respective categories, creating a detailed **product\_category\_mapping**. This not only streamlined our category analysis but also unlocked new dimensions for exploring product classifications, significantly amplifying our dataset's utility and facilitating nuanced market and product insights.

4. **Sentiment Analysis on Customer Reviews**: In our sentiment analysis, we conducted the work over the customer reviews (***review\_content***) comprehensively, starting with a detailed text pre-processing routine meant to clean the reviews for insight analysis. This, therefore, calls for the application of a custom text processing function whereby we converted the text into lowercase and stripped it of punctuation marks, and special characters, among other spaces, and thereafter removed the stop words, hence having the text reduced into its best form. We apply the following stemming techniques in text processing: the combined use of Porter, Lancaster, and Snowball stemmers followed by lemmatization using ***WordNetLemmatizer*** reduces words to a uniform, analytic, base, or root form.

During the text pre-processing stage, we measured the sentiment for each review using the ***SentimentIntensityAnalyzer*** from the NLTK library. The analysis returned a positive, negative, and neutral score of sentiments, as well as a composite score that sums up the general direction of the sentiment of the review titles. Based on these, each of the reviews was classified into positive, negative, or neutral sentiments, which in effect quantified the distribution of customer sentiment across the datasets. This rigorous method facilitated full elaboration of all the details regarding attitudes and preferences of consumers, yielding us granular data for use in decision-making and, in fact, strategy development.

5. **Normalization for Database Storage**: In our normalization process for database storage, we meticulously structured our dataset to better suit a relational database model, thereby enhancing data integrity and query efficiency. Initially, we extracted unique product details, user information, and reviews into separate tables, ensuring clear segregation between distinct entities such as products, users, and reviews. This separation facilitated a more organized data model, allowing for precise relationships and queries within the database.

Additionally, we transformed the complex, nested category data into a structured format, mapping products to categories through a normalized ***CategoryAssignments*** table, which linked ***product\_id*** with***category\_id***. This normalization process involved creating unique identifiers for each category, thereby streamlining the association between products and their respective categories. The creation of CSV files for ***products, users, reviews, categories, and product-category mappings*** was a critical step in preparing our data for database insertion, ensuring each entity was distinctly represented and ready for integration into the database schema.

To represent our normalization in a relational database, we designed and executed SQL scripts to create tables for ***Products****,* ***Users****,* ***Reviews****,* ***Categories****,* and ***CategoryAssignments***within our database ***“amazon\_sales\_project”***. Each table was carefully structured with appropriate data types, constraints, and relationships to ensure data integrity and support efficient querying. For instance, we established primary keys for entity identification, and foreign keys for relational integrity, and applied unique and not-null constraints where applicable to maintain the quality of our data.

# Authorship of the code - Kabir Chaturvedi

# Enhanced Products table creation with non-null constraints and default values

cursor.execute("""

CREATE TABLE IF NOT EXISTS Products (

product\_id VARCHAR(255) PRIMARY KEY,

product\_name VARCHAR(255) NOT NULL,

discounted\_price DECIMAL(10, 2),

actual\_price DECIMAL(10, 2),

discount\_percentage DECIMAL(5, 2),

rating DECIMAL(3, 1),

rating\_count INT,

about\_product TEXT,

img\_link TEXT,

product\_link VARCHAR(768) NOT NULL UNIQUE

);

""")

# Authorship of the code - Vedika Halwasiya

# Enhanced Users table creation with uniqueness constraint on user\_name

cursor.execute("""

CREATE TABLE IF NOT EXISTS Users (

user\_id VARCHAR(255) PRIMARY KEY,

user\_name VARCHAR(255) NOT NULL

);

""")

# Enhanced Reviews table creation with cascading deletes and a composite uniqueness constraint

cursor.execute("""

CREATE TABLE IF NOT EXISTS Reviews (

review\_id VARCHAR(255) PRIMARY KEY,

product\_id VARCHAR(255),

user\_id VARCHAR(255),

review\_title TEXT,

review\_content TEXT,

sentiment VARCHAR(50),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

""")

# Authorship of the code - Dhyey Joshi

# Categories Table

cursor.execute("""

CREATE TABLE IF NOT EXISTS Categories (

category\_id INT AUTO\_INCREMENT PRIMARY KEY,

category\_name VARCHAR(255) UNIQUE NOT NULL

);

""")

# Product categories Table

cursor.execute("""

CREATE TABLE IF NOT EXISTS CategoryAssignments (

product\_id VARCHAR(255),

category\_id INT,

PRIMARY KEY (product\_id, category\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id),

FOREIGN KEY (category\_id) REFERENCES Categories(category\_id)

);

""")

Following the table creation, we employed a custom Python function to load data from our CSV files into the MySQL database, ensuring each table was populated with the pre-processed and normalized data. This function included logic to handle duplicate entries gracefully and convert empty strings to **NULL** values for consistency with SQL data handling practices.

import csv

import mysql.connector

def load\_csv\_to\_mysql(csv\_file\_path, table\_name, cursor):

with open(csv\_file\_path, mode='r', encoding='utf-8') as csv\_file:

csv\_reader = csv.reader(csv\_file)

headers = next(csv\_reader) # Get column names from the first row

placeholders = ', '.join(['%s'] \* len(headers))

# Use INSERT IGNORE to ignore inserting duplicates

insert\_query = f'INSERT IGNORE INTO {table\_name} ({", ".join(headers)}) VALUES ({placeholders})'

for row in csv\_reader:

# Convert empty strings in the row to None (NULL in SQL)

row = [None if x == '' else x for x in row]

cursor.execute(insert\_query, tuple(row))

conn.commit()

load\_csv\_to\_mysql('./data/products.csv', 'Products', cursor)

load\_csv\_to\_mysql('./data/users.csv', 'Users', cursor)

load\_csv\_to\_mysql('./data/reviews.csv', 'Reviews', cursor)

load\_csv\_to\_mysql('./data/categories.csv', 'Categories', cursor)

load\_csv\_to\_mysql('./data/categoryassign.csv', 'CategoryAssignments', cursor)

Through this comprehensive normalization and database storage process, we significantly enhanced our dataset's usability and integrity, laying a solid foundation for robust data analysis and application development. Our methodical approach to creating normalized tables with proper data constraints, coupled with the careful insertion of data into these tables, exemplified our commitment to data quality and system scalability.

**Application Functionality**

**1.** **Product Comparison:**

In "Amazon: A2Z AmazeComparator," users have the luxury of comparing products side-by-side to make informed decisions. By leveraging detailed product data such as price, rating, and category, our platform simplifies the decision-making process. When users select products for comparison, our application dynamically retrieves and displays comprehensive information, highlighting differences in features, pricing, and consumer ratings. This feature is especially beneficial for users overwhelmed by choices or seeking the best value for their money, enabling them to visually contrast their options and select the product that best meets their needs and preferences.

cursor.execute("""

SELECT product\_id, product\_name, actual\_price, discounted\_price, rating

FROM Products

WHERE product\_id IN ('B08HDJ86NZ', 'B08DDRGWTJ');

""")

for row in cursor.fetchall():

print()

print(row)

**2.** **Search Products:**

Our search functionality enhances the user experience by allowing effortless navigation through our extensive product database. Users can easily search for products by entering a name, selecting a category, or specifying attributes of interest. This streamlined search process ensures that users can quickly find specific products or explore categories relevant to their interests. By providing a responsive and intuitive search mechanism, "A2Z AmazeComparator" empowers users to discover products efficiently, saving time and enhancing their shopping experience on our platform.

cursor.execute("""

SELECT product\_id, product\_name, actual\_price, rating

FROM Products

WHERE product\_name LIKE '%Wayona Nylon%';

""")

for row in cursor.fetchall():

print()

print(row)

**3.** **Review Sentiment Analysis:**

Understanding the sentiment behind user reviews is crucial for making purchase decisions. "Amazon: A2Z AmazeComparator" integrates review sentiment analysis, presenting an aggregated view of positive, negative, and neutral sentiments for each product. This functionality aids users in gauging overall customer satisfaction and pinpointing potential issues with products. By displaying sentiment analysis results alongside traditional ratings, our application provides a more nuanced perspective on product quality and user satisfaction, enabling shoppers to make more informed and confident decisions.

cursor.execute("""

SELECT R.review\_content, R.Sentiment

FROM Reviews R

JOIN Products P ON R.product\_id = P.product\_id

WHERE P.product\_id = 'B08DDRGWTJ';

""")

for row in cursor.fetchall():

print(row)

**4.** **Trending Products:**

Identifying and showcasing trending products is a key feature of "Amazon: A2Z AmazeComparator." By analyzing data on ratings and the volume of reviews, our platform highlights products that are currently popular among users, ensuring our customers are always informed about the latest trends and top-rated items. This feature not only enriches the shopping experience by simplifying the discovery of highly regarded products but also keeps users engaged with up-to-date content on what's trending, further enhancing the appeal of our platform.

cursor.execute("""

SELECT p.product\_id, p.product\_name, p.rating, COUNT(r.review\_id) as review\_count

FROM Products p

LEFT JOIN Reviews r ON p.product\_id = r.product\_id

GROUP BY p.product\_id, p.product\_name, p.rating

ORDER BY p.rating DESC, review\_count DESC

LIMIT 10;

""")

for row in cursor.fetchall():

print()

print(row)

**5.** **Discounts and Deals:**

Everyone loves a good deal, and "Amazon: A2Z AmazeComparator" excels in bringing the best discounts and deals to the forefront. Our platform meticulously curates products with the highest discounts, allowing users to easily access deals that offer significant savings. This feature is particularly advantageous for budget-conscious shoppers looking to maximize value without extensive research. By centralizing information on discounts and special offers, "A2Z AmazeComparator" ensures users don't miss out on opportunities to save while shopping for their favorite products.

cursor.execute("""

SELECT product\_id, product\_name, actual\_price, discounted\_price, discount\_percentage

FROM Products

WHERE discounted\_price IS NOT NULL AND actual\_price > discounted\_price

ORDER BY discount\_percentage DESC

LIMIT 10;

""")

# Fetch and print the results

for row in cursor.fetchall():

print(row)

**6.** **User Reviews and Ratings Dashboard:**

"A2Z AmazeComparator" offers a personalized user experience through a dedicated dashboard where users can review their contributions, including reviews and ratings. This feature empowers users to manage their input, allowing them to update or delete their reviews as their opinions or experiences change. The dashboard serves as a centralized platform for users to track their interactions and influence on the community, fostering a sense of ownership and engagement with the site. By providing users with control over their content, "A2Z AmazeComparator" enhances user satisfaction and encourages active participation in our review ecosystem.

user\_id\_to\_lookup = 'AE22E2AXODSPNK3EBIHNGYS5LOSA' #

cursor.execute("""

SELECT r.review\_id, r.product\_id, p.product\_name, r.review\_title, r.review\_content, r.sentiment, p.rating

FROM Reviews r

JOIN Products p ON r.product\_id = p.product\_id

WHERE r.user\_id = %s

ORDER BY r.review\_id DESC;""", (user\_id\_to\_lookup,))

# Fetch and print the results

for row in cursor.fetchall():

print(row)

**7.** **Category-Wise Product Browsing**

On "Amazon: A2Z AmazeComparator," users can effortlessly navigate through products sorted by categories, enhancing their browsing experience. Utilizing the structured organization of the Categories and CategoryAssignments tables, our platform displays a curated selection of products within each specific category. This approach simplifies product discovery, allowing users to explore items that align with their interests or needs in a more organized manner. Whether customers are looking for the latest tech gadgets or kitchen essentials, category-wise browsing streamlines the shopping process, making it easy and enjoyable to find exactly what they're searching for on our platform.

category\_name = "Electronics"

cursor.execute("""

SELECT p.product\_id, p.product\_name, p.actual\_price, p.rating

FROM Products p

JOIN CategoryAssignments ca ON p.product\_id = ca.product\_id

JOIN Categories c ON ca.category\_id = c.category\_id

WHERE c.category\_name = %s

ORDER BY p.rating DESC

LIMIT 10;

""", (category\_name,))

for row in cursor.fetchall():

print(row)

**8.** **Best Sellers in a Specific Category**

Highlighting best-selling products within specific categories is a hallmark of "Amazon: A2Z AmazeComparator." By analyzing ratings and review volumes, we spotlight the most popular products, guiding users toward highly recommended items in their category of interest. This feature not only aids users in discovering products that are proven successful in the community but also ensures that they are making informed decisions based on the collective experience of other shoppers. It's a valuable tool for those seeking top-tier products in any given category.

category\_name = "Speakers"

cursor.execute("""

SELECT p.product\_id, p.product\_name, COUNT(r.review\_id) AS review\_count, p.rating, c.category\_name

FROM Products p

JOIN Reviews r ON p.product\_id = r.product\_id

JOIN CategoryAssignments ca ON p.product\_id = ca.product\_id

JOIN Categories c ON ca.category\_id = c.category\_id

WHERE c.category\_name = %s

GROUP BY p.product\_id, c.category\_name

ORDER BY review\_count DESC, p.rating DESC

LIMIT 10;

""", (category\_name,))

for row in cursor.fetchall():

print(row)

**9.** **New Arrivals**

Keeping our users informed about the latest additions to the marketplace is key to ensuring they have access to the newest products. "Amazon: A2Z AmazeComparator" prominently features new arrivals, allowing users to explore the freshest offerings with ease. This section is regularly updated to include the latest products added to our catalog, ensuring that our users are always in the loop on new trends and innovations. It's a fantastic way for users to discover and engage with the newest products available.

cursor.execute("""

SELECT product\_id, product\_name, actual\_price, img\_link

FROM Products

ORDER BY product\_id DESC

LIMIT 10;

""")

for row in cursor.fetchall():

print(row)

**10.** **Top Rated Products**

"Amazon: A2Z AmazeComparator" is dedicated to showcasing excellence through our top-rated product features. Products with a rating of 4.0 and above are highlighted, making it easier for users to find items that have been universally acclaimed for their quality. This function serves as a beacon for those seeking the best of the best, guiding users towards products that have achieved the highest levels of customer satisfaction.

cursor.execute("""

SELECT product\_id, product\_name, rating

FROM Products

WHERE rating >= 4.0

ORDER BY rating DESC, rating\_count DESC

LIMIT 10;

""")

for row in cursor.fetchall():

print(row)

**11.** **Most Reviewed Products**

Understanding which products are sparking the most conversation is invaluable. On "Amazon: A2Z AmazeComparator," we identify and display the products with the highest number of reviews, allowing users to see what's currently trending in the community. This feature not only helps users identify popular products but also provides insights into the items that are engaging customers the most, based on the volume of reviews.

cursor.execute("""

SELECT P.product\_id, P.product\_name, COUNT(R.review\_id) as review\_count

FROM Products P

JOIN Reviews R ON P.product\_id = R.product\_id

GROUP BY P.product\_id

ORDER BY review\_count DESC

LIMIT 10;

""")

for row in cursor.fetchall():

print(row)

**12.** **Category-Wise Discounts and Deals**

Our platform takes pride in bringing the best deals to our users, organized neatly by categories. By highlighting categories with the most attractive discounts, "Amazon: A2Z AmazeComparator" makes it easy for bargain hunters to find outstanding savings in their areas of interest. This functionality ensures users never miss out on an opportunity to discover great deals, making it simpler to navigate through discounts and offers across various categories. It's an essential tool for users looking to maximize value while shopping.

cursor.execute("""

SELECT P.product\_id, P.product\_name, COUNT(R.review\_id) as review\_count

FROM Products P

JOIN Reviews R ON P.product\_id = R.product\_id

GROUP BY P.product\_id

ORDER BY review\_count DESC

LIMIT 10;

""")

for row in cursor.fetchall():

print(row)

**OVERALL CONTRIBUTION SUMMARY**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Task | Contribution | Avg Time Spend (hrs) |
| Dhyey Joshi | Conceptual Schema | Provided ideas and established relationships for decomposing the table into distinct, normalizable entities, such as the 'Products' table, which holds specific information like 'product\_id', 'product\_name', and 'about\_product', and similarly conceptualized other tables. | 2 hrs |
|  | Database | Designed constraints for 'Categories' and 'CategoryAssignments' to minimize redundancy and established key relationships across tables for data integrity. | 1.5 hrs |
|  | Code | Populated the decomposed tables—Products, Users, Reviews, Categories, and CategoryAssignments—by creating and importing separate CSV files. Also developed the first four functional queries relevant to our web app, 'A2Z AmazeComparator'.  Developed sentiment analysis algorithms and integrated them with product review data. | 3 hrs |
| Vedika Halwasiya | Conceptual Schema | Identified relevant columns to be included in the schema and the relationships between different entities. This involved defining how Products interact with ‘Categories’ through the ‘CategoryAssignments’ table and linking ‘Users’ to their ‘Reviews’ to facilitate personalized insights, among other connections. | 1.5 hrs |
|  | Database | Designed 'Users' and 'Reviews' table constraints to link reviews with users, enhancing personalized experiences and product insights, vital for our app's engagement and advancing in informed product selection. | 2 hrs |
|  | Code | Data normalization for the 'sales' dataset involved converting 'product\_id', 'user\_id', and 'review\_id' to strings and price columns to floats to ensure they're treated as unique identifiers. Led data preparation, mapping products to categories, and developed functionalities 5 through 8 for our web application, solidifying data integrity and enhancing user experience. | 3 hrs |
| Kabir Chaturvedi | Conceptual Schema | Described the various components and their relationships within the database schema. Designed the schema using Draw.io, collaborated on refining the ERD, and contributed to defining the database relationships. Assisted in schema validation and offered insights for future scalability considerations. | 2 hrs |
|  | Database | Executed data pre-processing tasks and created a database for our application by establishing a MySQL connection. Led the charge in data cleansing and database schema implementation, including the creation of a Product table with appropriate constraints and indices. | 1.5 hrs |
|  | Code | Implemented logic to extract and organize unique products, users, and reviews into separate, clean datasets. These contributions significantly streamlined the database's architecture, laying a solid foundation for efficient data retrieval and manipulation within our application. Also built queries and provided Functionality 10 to Functionality 12 related to our application. | 3 hrs |

**Conclusion**

Our team is energized by our progress on the "Amazon: A2Z AmazeComparator's" database and eagerly anticipates developing the front-end application. We aim to revolutionize Amazon product comparisons with a platform that's both innovative and user-friendly. As we transition to this crucial phase, we're focused on crafting an interface that enhances user experience through simplicity and advanced analytics. The upcoming unveiling of our platform promises to set new benchmarks in e-commerce shopping tools, offering users an unparalleled resource for informed decision-making. We're committed to redefining online shopping comparisons and excited to share our breakthrough with the community.