**Abstract:**

This project implements and evaluates three association rule mining algorithms: Brute Force, Apriori, and FP growth. By applying these algorithms to transactional data from five well-known retailers—Amazon, Best Buy, Nike, Walmart, and Target—we analyze their ability to uncover purchasing patterns and generate meaningful association rules. The project highlights the practical application of data mining techniques in the retail industry, offering insights into the comparative performance of each algorithm, customer purchasing behavior, and the trade-offs in selecting different association rule mining methods.

**Introduction:**Association rule mining is a crucial technique in data analysis, particularly useful in sectors like retail and e-commerce. This project aims to implement and compare three major association rule mining algorithms:

* **Brute Force**: A custom-built method that exhaustively evaluates all possible itemsets.
* **Apriori**: A well-known algorithm for frequent itemset generation, implemented using the mlxtend library.
* **FP-Growth**: A tree-based method for mining frequent patterns, also implemented using mlxtend.

These algorithms are tested on transaction datasets from five prominent retail chains—Amazon, Best Buy, Nike, Walmart, and Target. Through this comparative study, we aim to identify each algorithm's relative strengths and limitations in the context of retail data analysis.

**Frequent Itemset Discovery:**

Frequent set discovery lies at the core of association rule mining, focusing on identifying sets of items that frequently occur together within transactions. This project explores and compares different methods for uncovering these frequent itemsets, ranging from the exhaustive Brute Force approach, which evaluates all possible combinations, to more efficient algorithms like Apriori and FP-Growth. These comparisons shed light on how various techniques handle the complexity of itemset discovery.

**Support and Confidence Metrics:** are two key measures guiding the analysis in this project. Support reflects how often an itemset appears within the transaction dataset, providing a foundation for identifying frequent patterns. Confidence measures the likelihood that one item is purchased given the purchase of another, which is crucial for understanding the strength of relationships between items. These metrics are vital for filtering and ranking the association rules, ensuring the discovery of meaningful patterns that can be applied to real-world business scenarios.

The ultimate goal of this project is to generate Association Rules that reveal customer purchasing patterns. These rules provide valuable insights into customer behavior, offering applications for personalized product recommendations and store layout optimization. By analyzing these patterns, businesses can make informed decisions to enhance customer experience and maximize sales opportunities. A critical component of this project is the comparison of Algorithm Efficiency. The computational efficiency of the Brute Force, Apriori, and FP-Growth algorithms is measured through their execution times. This comparison highlights the trade-offs between exhaustive search methods and more optimized approaches, helping to assess which algorithms are best suited for large-scale retail data analysis.

Finally, the project emphasizes the importance of Data Preprocessing in ensuring the success of data mining tasks. The process of loading and preparing transactional data plays a significant role in the overall effectiveness of association rule mining. By illustrating proper preprocessing techniques, this project demonstrates how clean and well-structured data can enhance the performance of the algorithms and lead to more accurate and insightful results.

**Project Workflow**

**Data Preparation and Loading:**

* CSV File Creation: The transaction data provided by the professor was expanded and systematically organized into Excel sheets, with separate sheets created for each store: Amazon, Best Buy, Nike, Walmart, and Target.
* CSV Loading: A custom function, load\_transactions\_from\_csv, was developed to read these CSV files and convert the data into a format that is appropriate for further analysis.

**User Interaction:** The program includes a user-friendly interface that enables users to select a specific store for analysis and input desired thresholds for support and confidence. Input validation functions are implemented to ensure that all user entries fall within acceptable ranges before proceeding with the analysis.

**Algorithm Implementation:**

1. **Brute Force Algorithm:**
   1. A custom algorithm is implemented using Python’s intertools to generate combinations of items.
   2. This method exhaustively checks all possible itemsets against the defined minimum support threshold.
   3. It generates association rules based on frequent itemsets that meet the minimum confidence requirement.
2. **Apriori Algorithm:**
   1. This approach leverages the Apriori implementation from the mlxtend library.
   2. It efficiently generates frequent itemsets following the apriori principle, which reduces the search space.
3. **FP-Growth Algorithm:** Also implemented using the mlxtend library, the FP-Growth algorithm employs a tree-based structure to efficiently mine frequent patterns from the dataset.

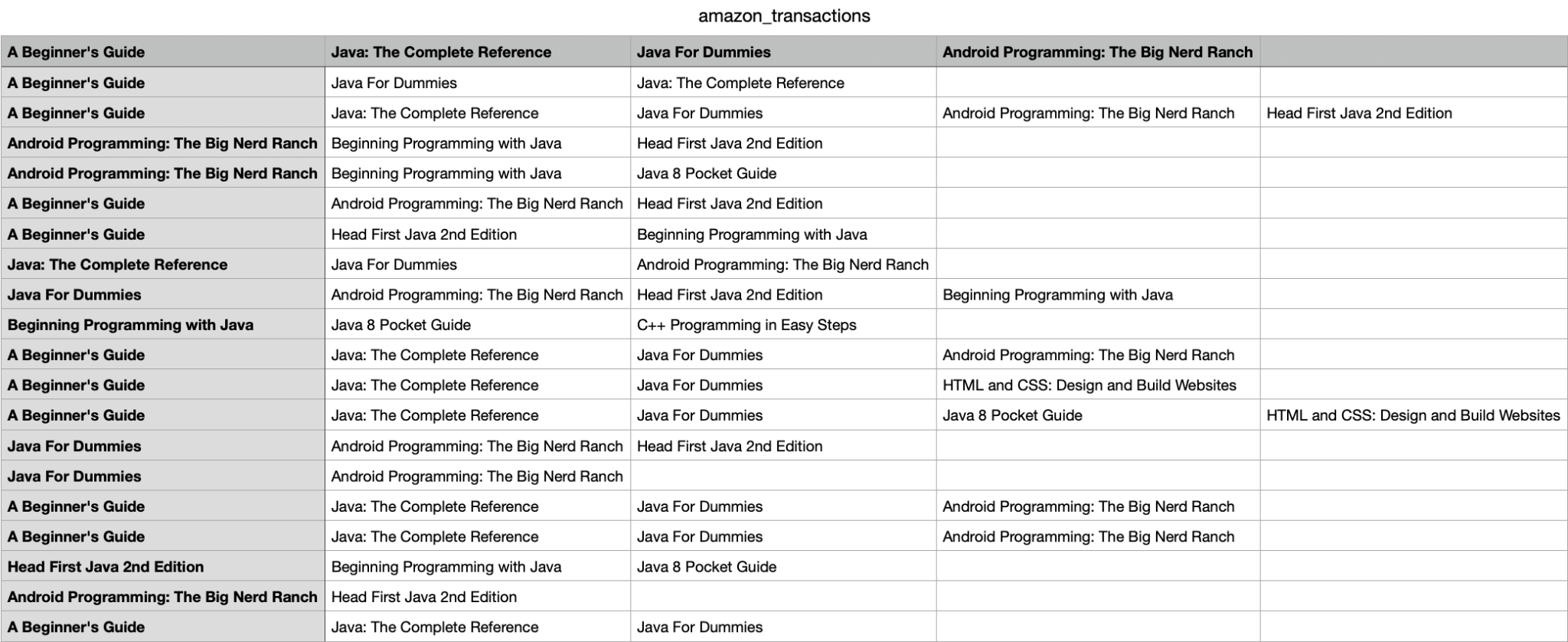
**Performance Measurement:** Execution time for each algorithm is measured using Python’s time module. This allows for direct comparisons of computational efficiency across the Brute Force, Apriori, and FP-Growth algorithms.

**Result Comparison and Analysis:** The project compares the number of frequent itemsets and association rules generated by each algorithm. It checks for consistency in the results produced by all three methods and identifies the fastest algorithm for each specific analysis.

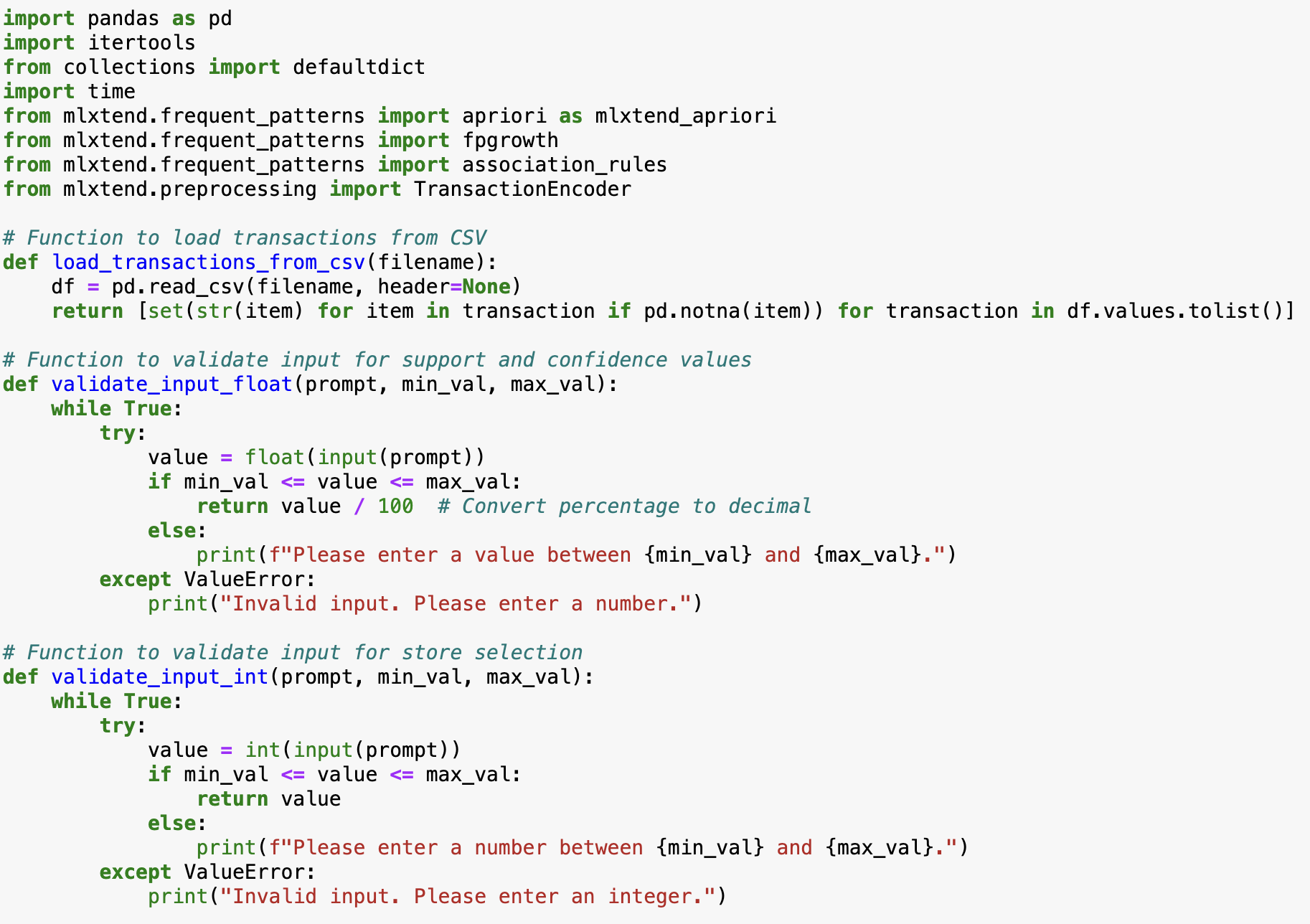
**Rule Display and Interpretation:** The project implements functions to display the generated association rules in a clear and readable format. These functions also provide insights into the strength of the discovered associations by reporting their corresponding support and confidence metrics.

**Item Analysis:** Item counts and their respective support values are calculated and displayed. The analysis also determines whether individual items meet the defined support threshold, helping to assess the relevance of frequent itemsets further.

**Screenshots:**

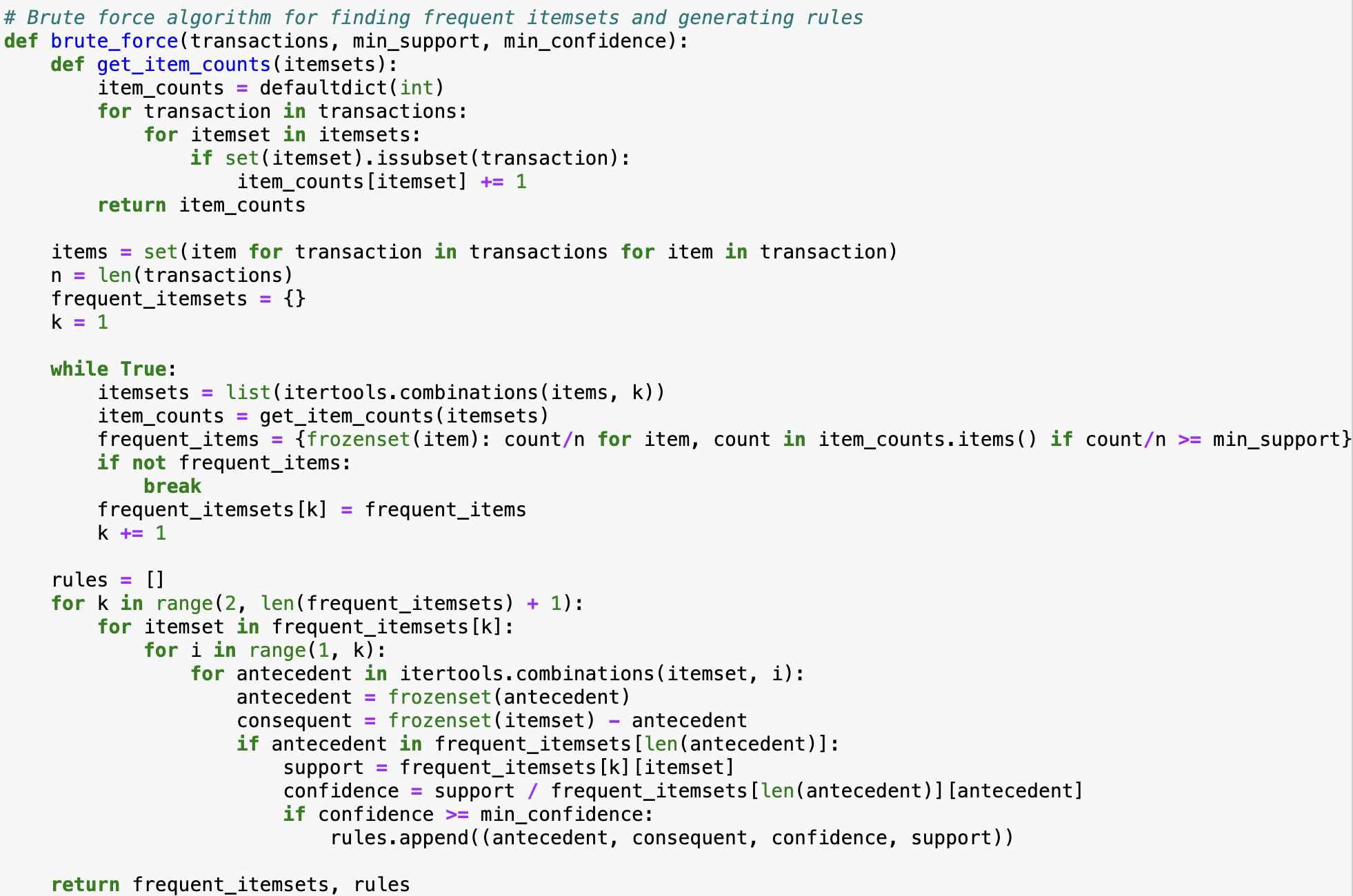


**Figure 1:** Amazon Transactions



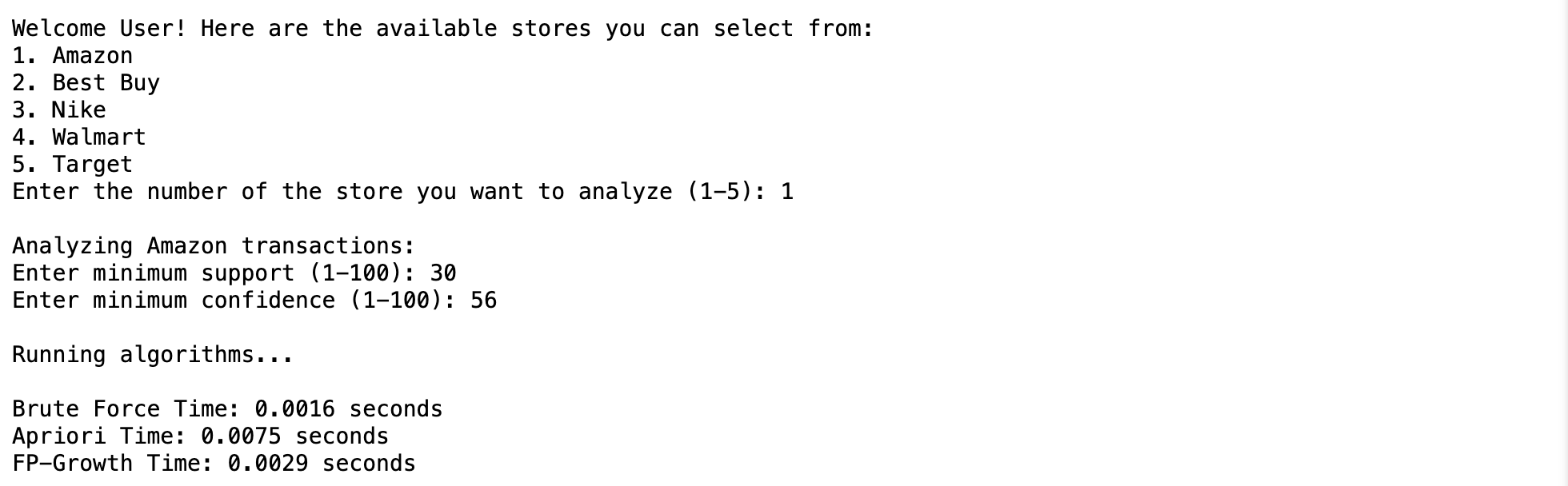


**Figure 2:** Importing transaction from CSV

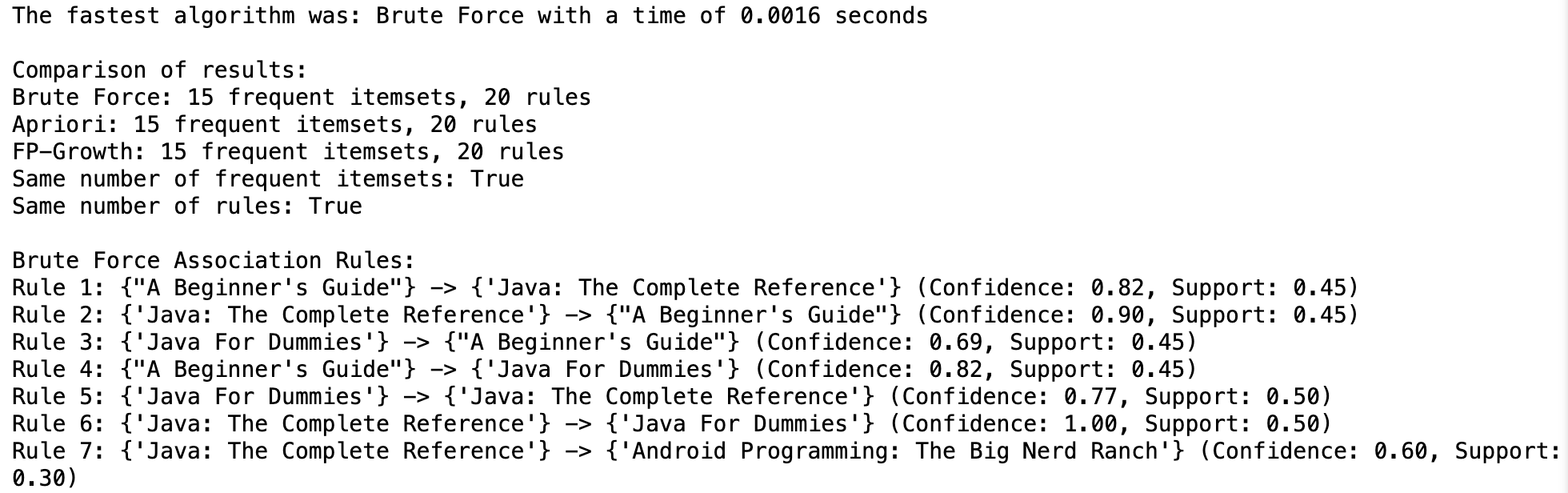


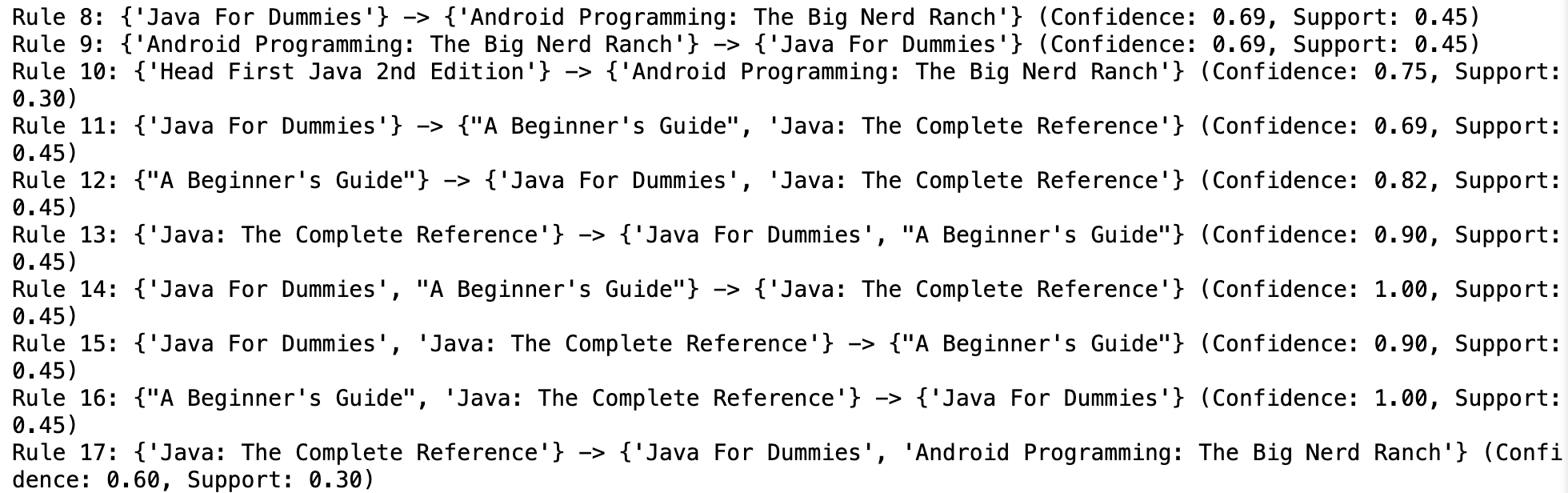


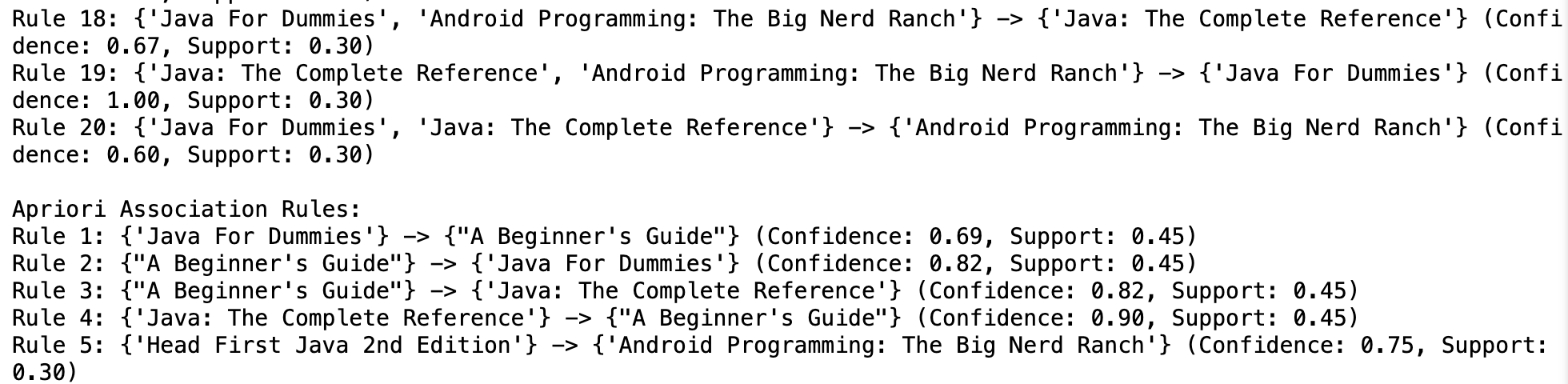
**Figure 3:** Implementing Brute Force and Apriori Algorithm

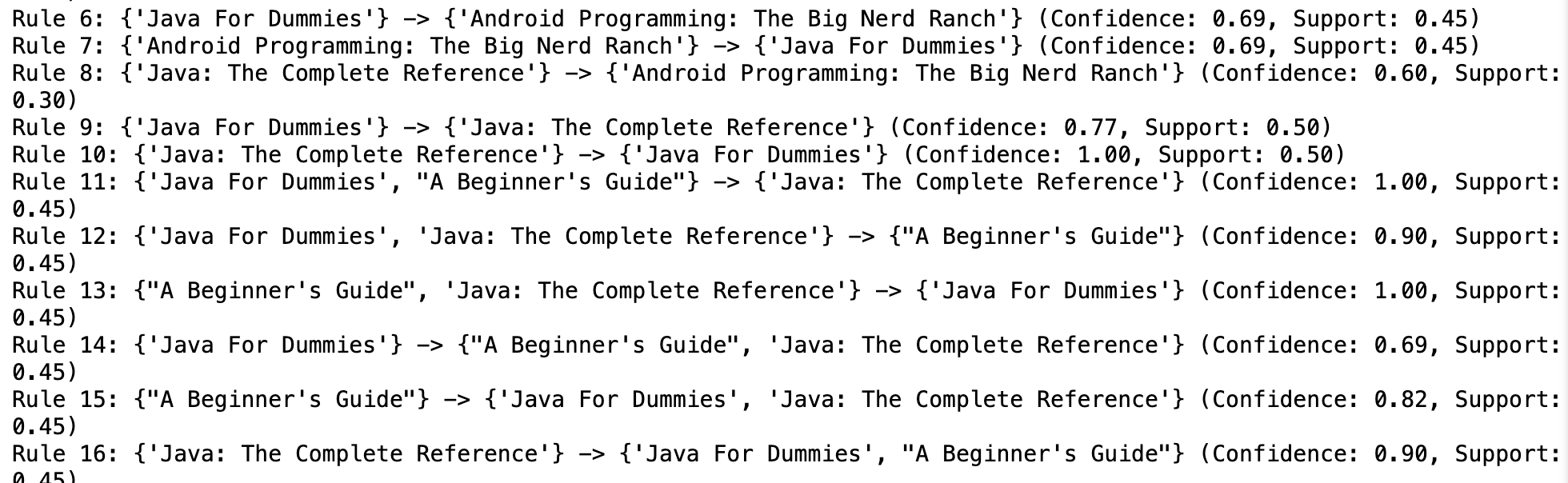


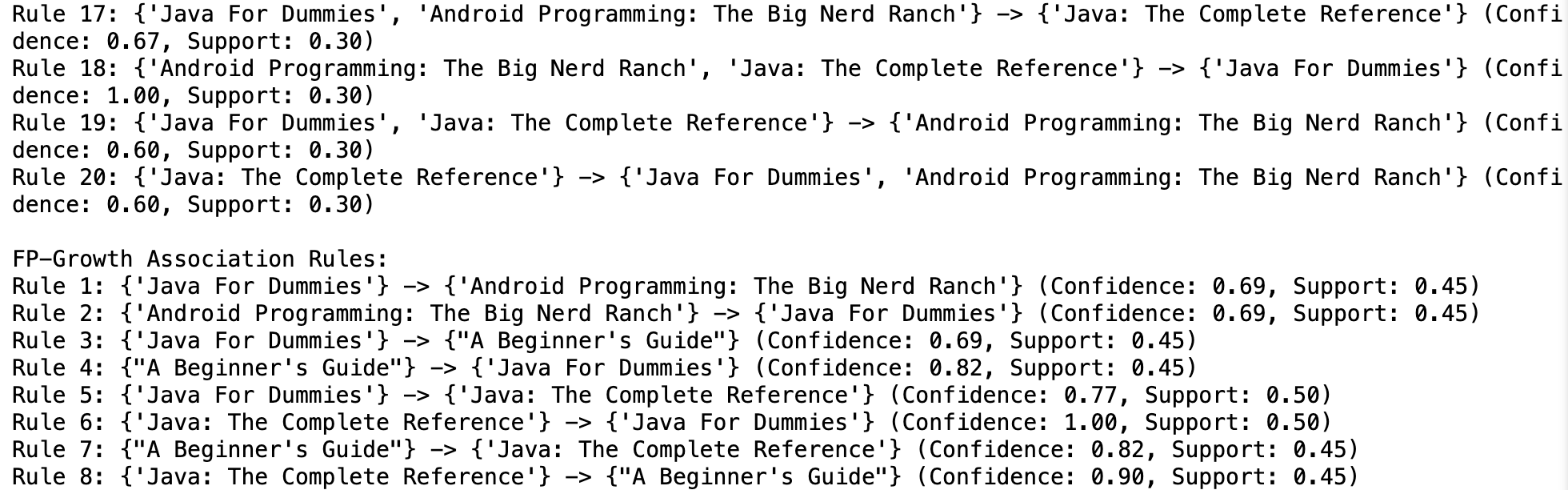
**Figure 4:** User selects a store and enters minimum support and confidence

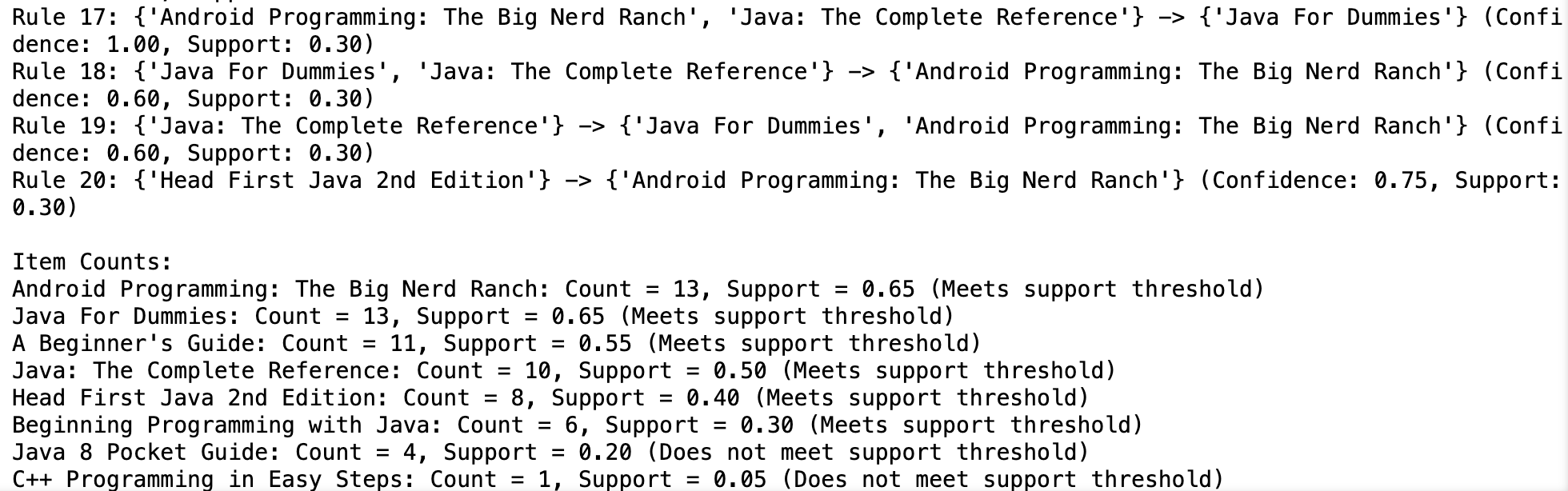
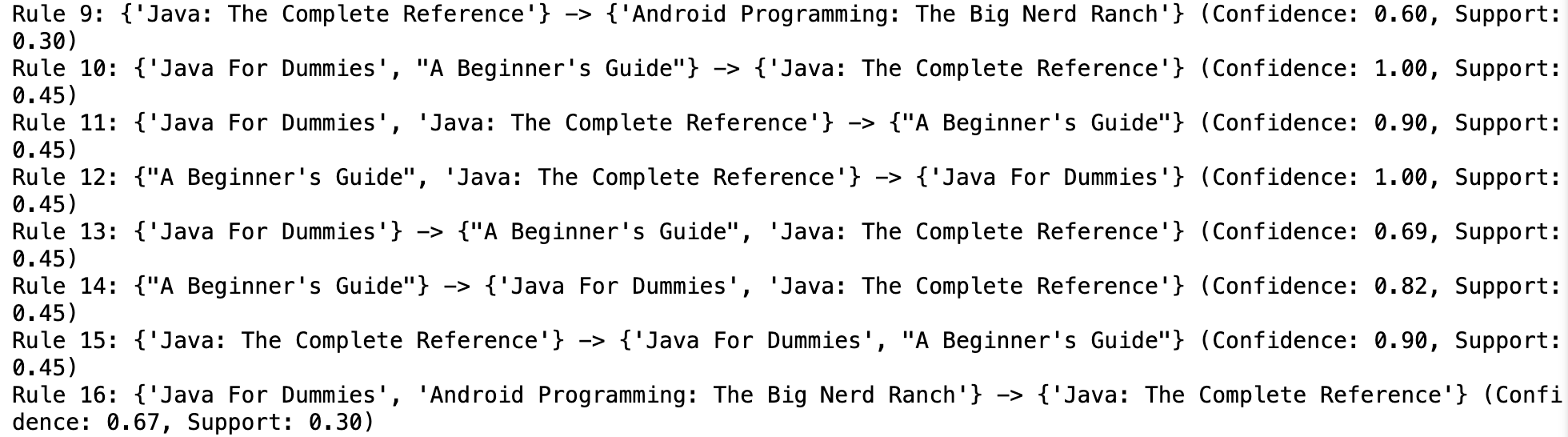


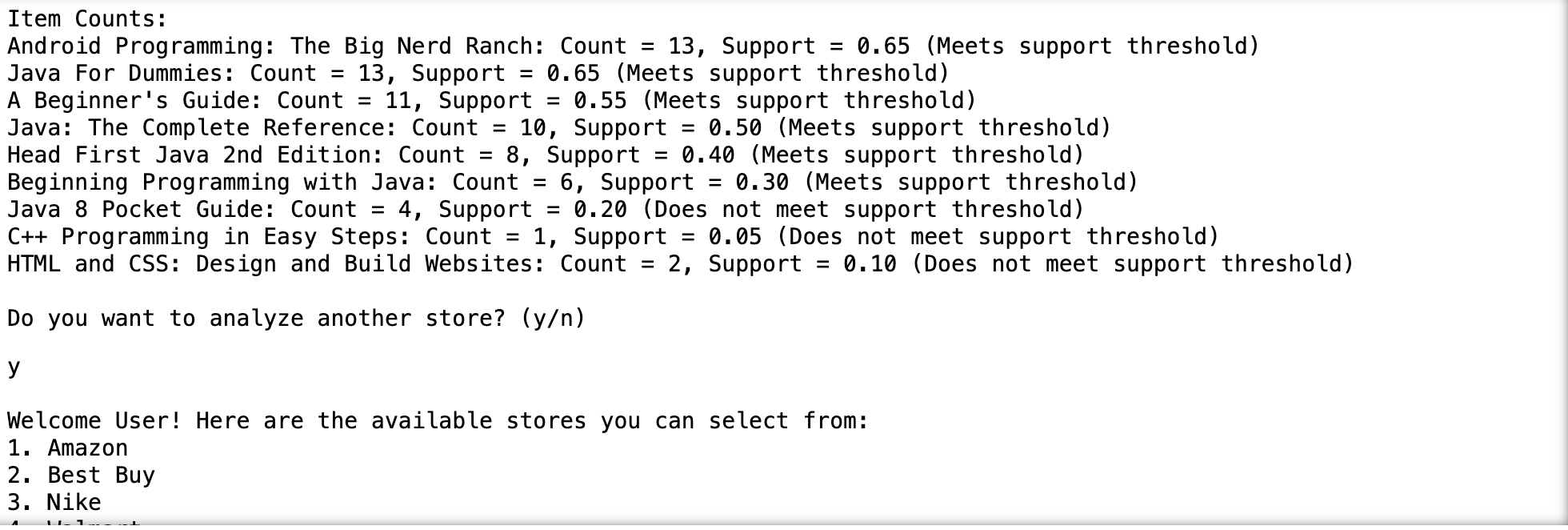




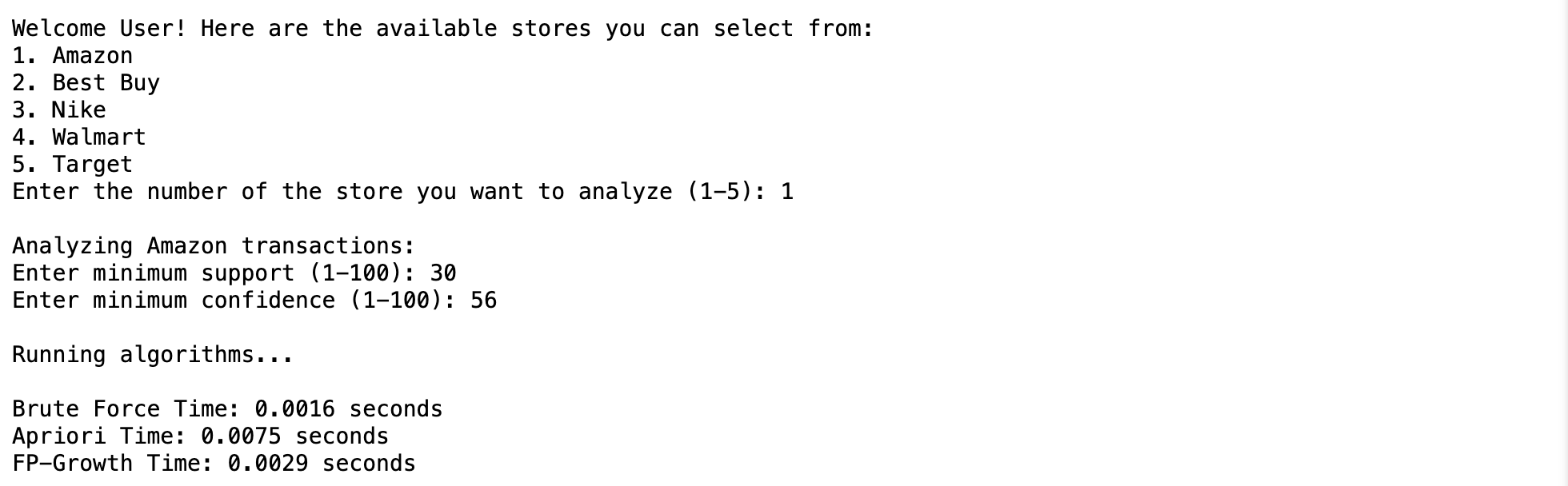








**Figure 7:** We get each algorithm time, fastest algorithm, association rules & item count



**References:**

Github: <https://github.com/tejas100/Tejas_Belakavadi_Kemparaju_DM_midtermproject>

**References**

1. [Mlxtend library documentation](http://rasbt.github.io/mlxtend/)
2. [Brute Force](https://stackoverflow.com/questions/25899839/how-to-write-a-brute-force-algorithm)
3. ChatGPT