# Midterm Project Report

NAME: Varshith

NJIT UCID: sl1203

Email Address: sl1203@njit.edu

Date: 03-10-2024

Professor: Yasser Abduallah

Course: CS 634 (105) Data Mining

## Title: Comparative Analysis of Data Mining Algorithms

### Abstract

This project focuses on implementing and comparing three different algorithms for frequent itemset mining and association rule generation: brute force, Apriori, and FP-Growth. Furthermore, we have created transactional databases that represent real-world retail environments, allowing us to derive association rules based on parameters defined by the user. The primary aim is to evaluate the performance of each algorithm in terms of efficiency, accuracy, and scalability.

The input parameters for our analysis include minimum support and confidence levels, which the user specifies to filter out less significant patterns from the data. By examining these algorithms under varying conditions, we hope to gain valuable insights into their applicability in real-world scenarios and to identify which algorithm performs best in different situations.

### Introduction

### This project is an exploration into the effectiveness of three distinct algorithms used for frequent itemset mining and association rule generation. In today’s data-driven world, data mining has become essential for discovering valuable patterns and insights from extensive datasets, serving as a foundation in various sectors such as retail, finance, and healthcare.

In this project, I have **used** three pivotal algorithms: brute force, Apriori, and FP-Growth. Each of these algorithms employs a unique strategy to extract frequent itemsets and derive association rules. My aim is to conduct a thorough comparative analysis that highlights the strengths and weaknesses of each algorithm, specifically in terms of their efficiency and accuracy. This evaluation is intended to provide valuable insights into how these algorithms can be effectively utilized in practical, real-world situations.

### Core Concepts and Principles

**Frequent Itemset Discovery**: The main objective of the Apriori Algorithm is to uncover frequent itemsets, which are essentially collections of items that consistently occur together in various transactions. By identifying these itemsets, we can gain valuable insights into customers' purchasing behaviors and preferences. This understanding can help businesses tailor their marketing strategies and product offerings to better meet the needs and desires of their customers, ultimately leading to enhanced customer satisfaction and increased sales.

**Support and Confidence**: Support indicates how often an item or itemset appears in transactions, reflecting its importance. In contrast, confidence measures the likelihood of one item being purchased when another is already bought, indicating the strength of the relationship between items. These metrics guide our analysis and help in making sense of the data.

### Association Rules:

The identification of strong association rules is crucial for uncovering which items are often purchased together by customers. Analyzing these patterns enables businesses to refine their sales strategies effectively. This insight allows them to provide targeted recommendations to customers based on their purchasing history, ultimately enhancing the overall shopping experience. For example, if a customer frequently buys bread and butter together, the retailer can suggest similar items or offer special discounts on these products, thereby encouraging more sales and fostering customer loyalty.

### Project Workflow

1. **Data Creation**:

In this step, I create transaction data for various retail scenarios that represent different stores. Each transaction includes a list of items purchased by a customer, organized into two columns: transaction\_id and the corresponding items. The data is hard-coded rather than randomly generated, ensuring that I achieve consistent and reproducible outputs.

2**. Determining Minimum Support and Confidence:**

The code prompts for the **minimum support and confidence levels,** enabling the filtering of less significant patterns from the analysis. By specifying these parameters, We can refine the results to focus on the most meaningful associations in the transaction data.

3**. Iterating Through Candidate Itemsets**:

We apply the Apriori Algorithm by generating candidate itemsets of increasing sizes, starting with single items and progressing to pairs, triplets, and beyond. This iterative approach resembles brute force, exploring various item combinations.

4**. Calculating Support Counts**:

For each candidate itemset, we calculate its support by counting how many transactions contain that itemset. Itemsets meeting the minimum support threshold are retained, while others are discarded.

5. **Calculating Confidence**:

Next, we assess the confidence of the generated association rules, measuring the strength of connections between items. This involves comparing the support values of individual items and itemsets.

6**. Generating Association Rules**:

We extract association rules that satisfy both the minimum support and confidence requirements, providing valuable insights into frequently co-purchased items.

### Results and Conclusion

All three algorithms produced the same outputs in terms of the number of frequent itemsets and association rules generated. However, **FP-Growth outperformed** the others in terms of execution time, making it the most efficient choice for this dataset. The Brute Force method also demonstrated relatively quick performance, but both it and Apriori were slower than FP-Growth.

This consistency in results across the algorithms indicates the reliability of the findings, while the faster execution of FP-Growth suggests that it is a preferable method for future analyses involving larger datasets.

### Screenshots:

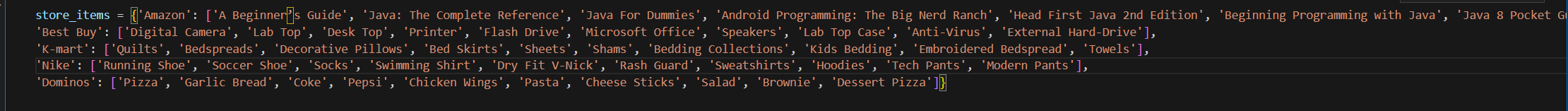


Figure: Items for each database

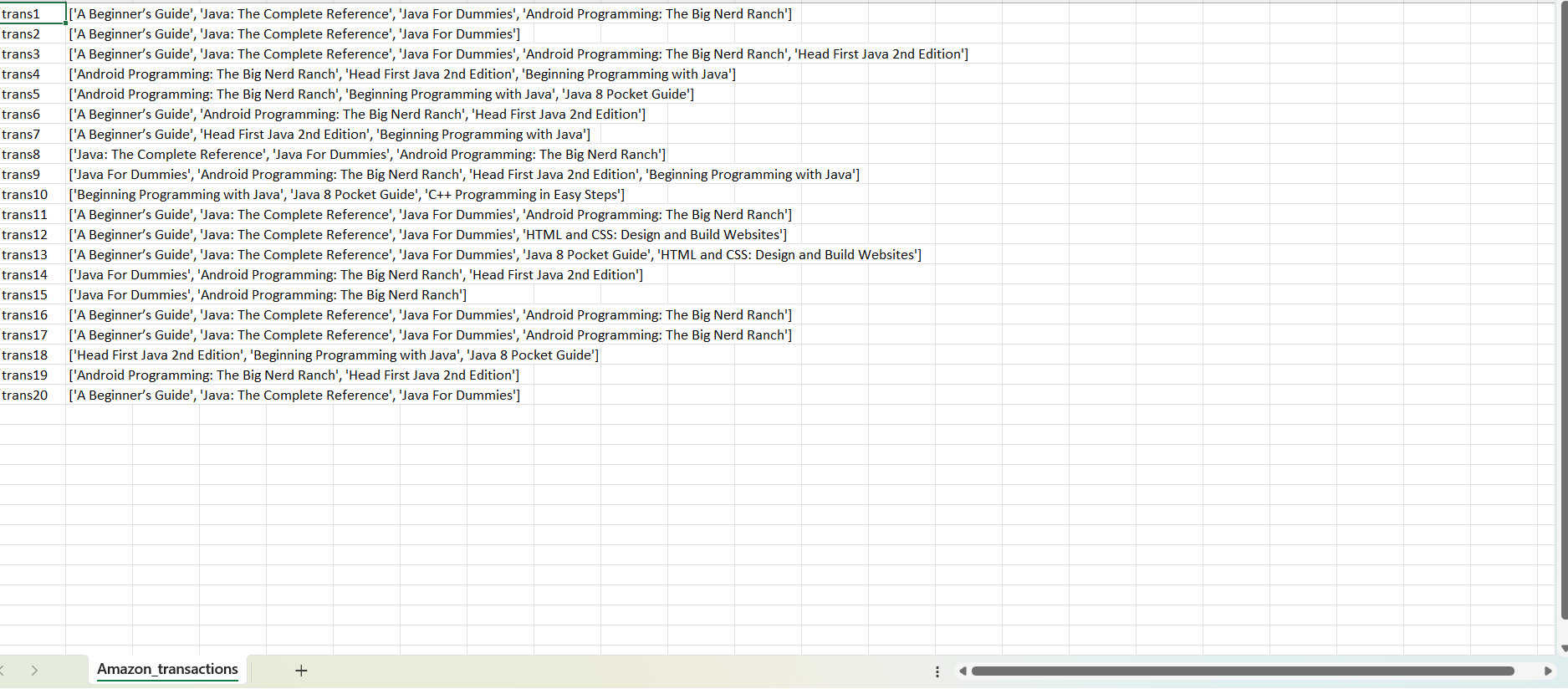


Figure: Amazon Transactions Database

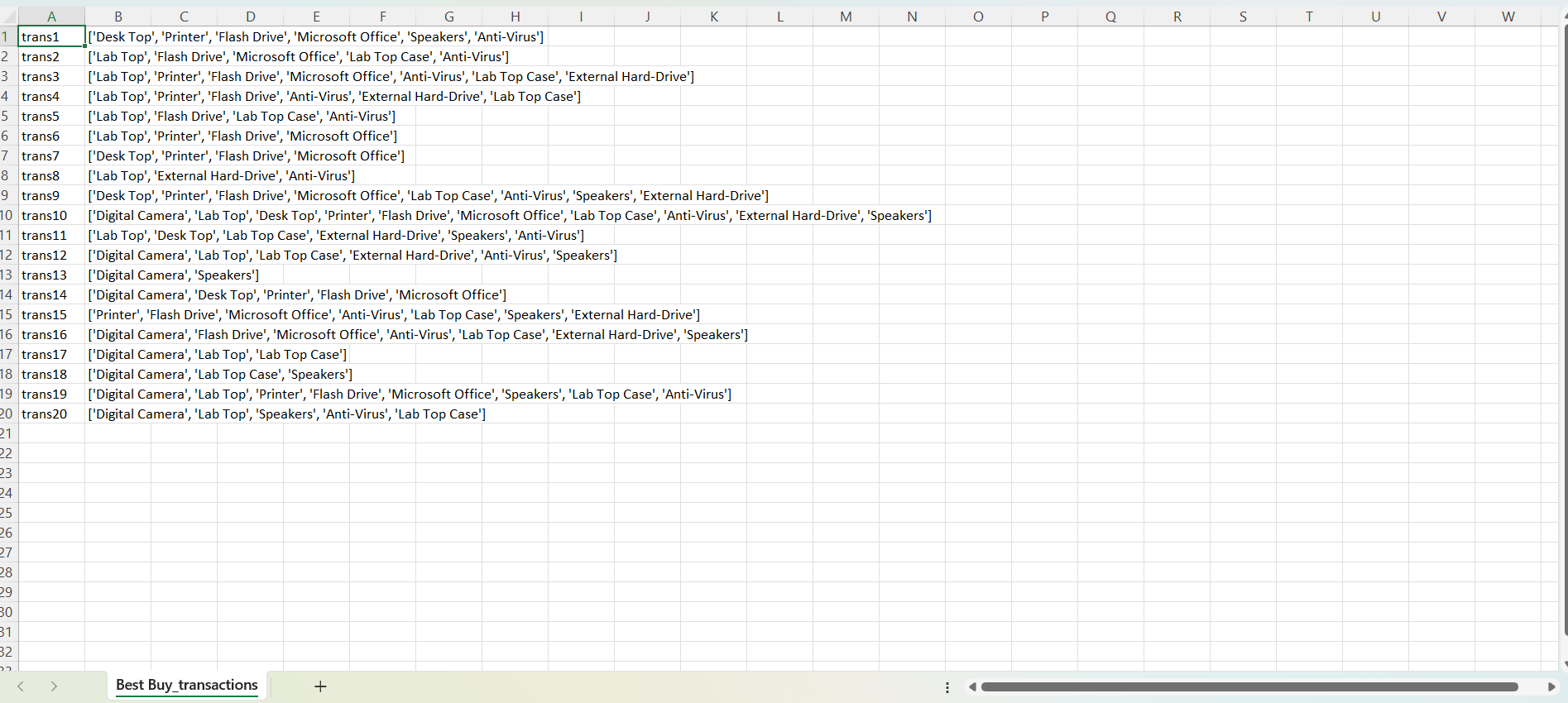


Figure: BestBuy Transactions Database

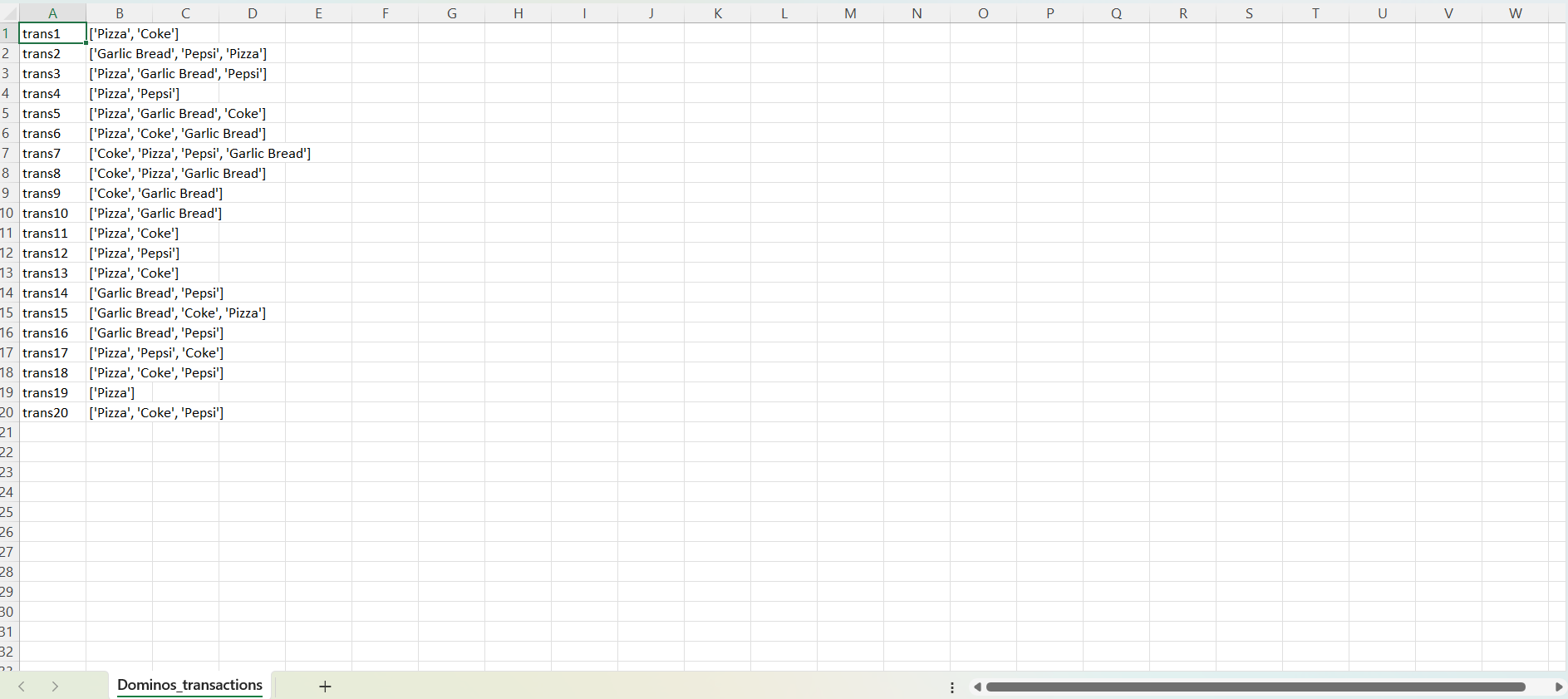


Figure: Domino's Transactions Database

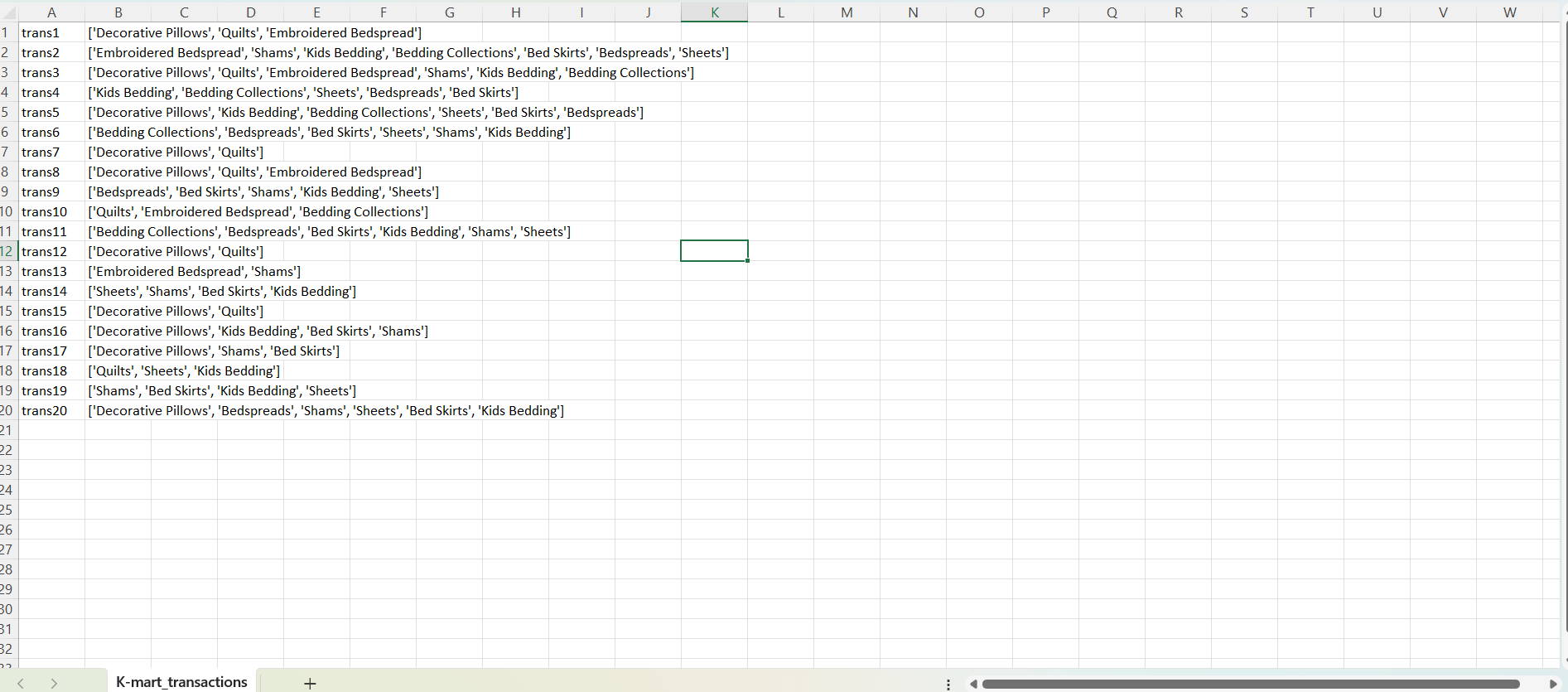


Figure: K-Mart Transactions Database

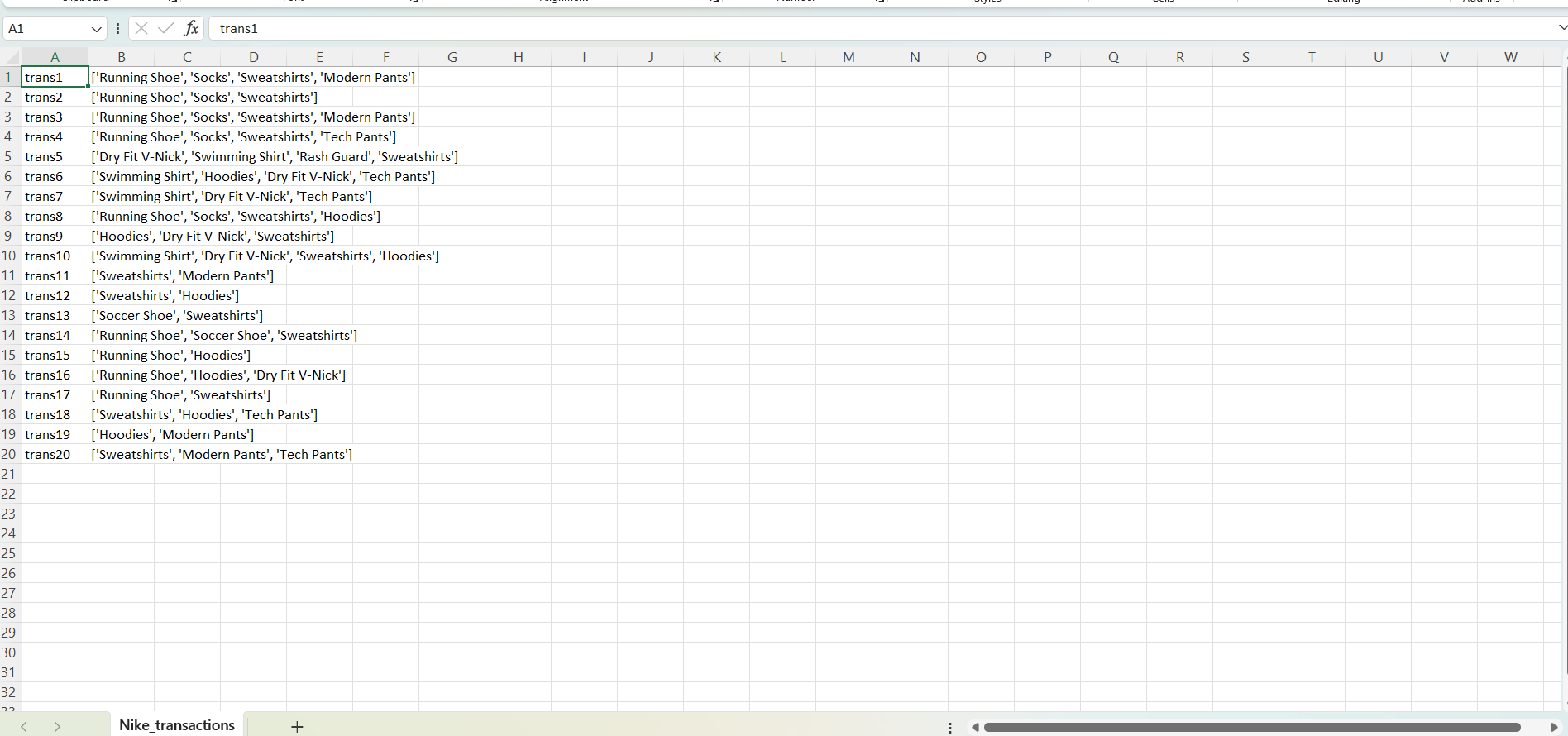


Figure: Nike Transactions Database

### Screenshots of the Code From Ipynb File:

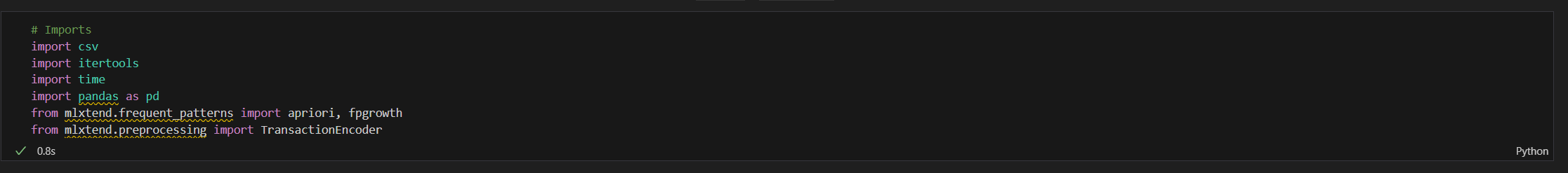


Figure: Imports

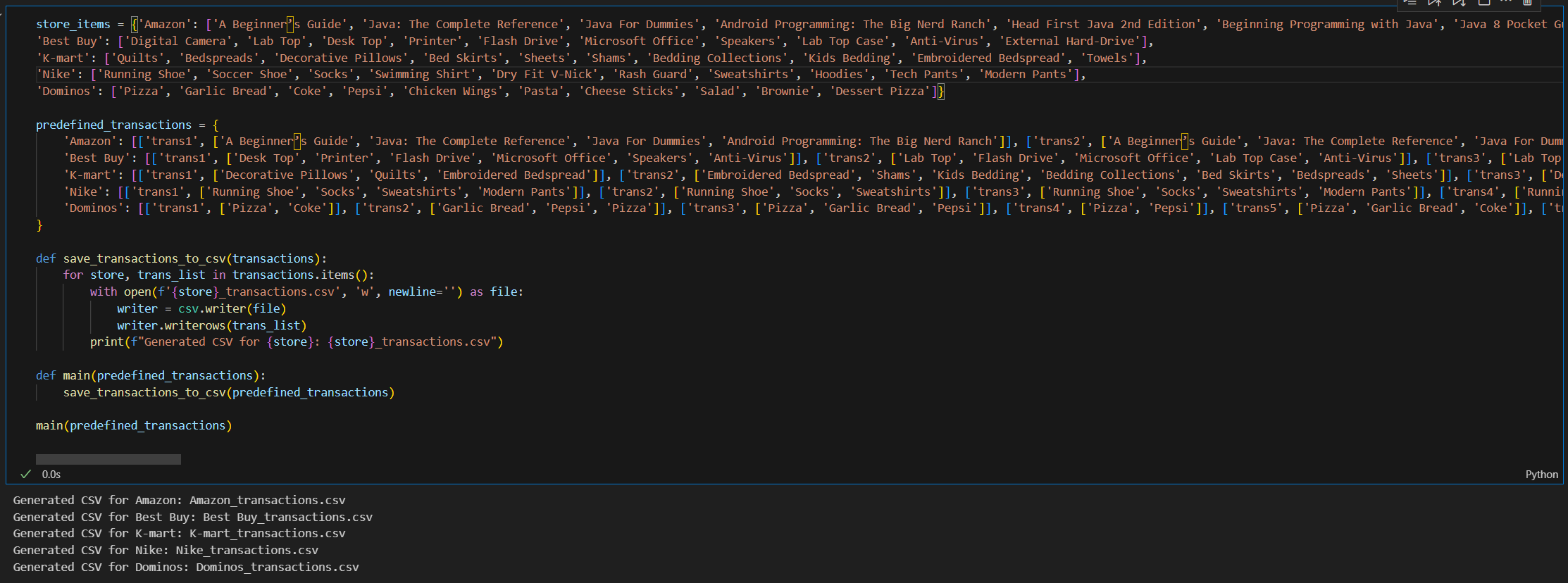
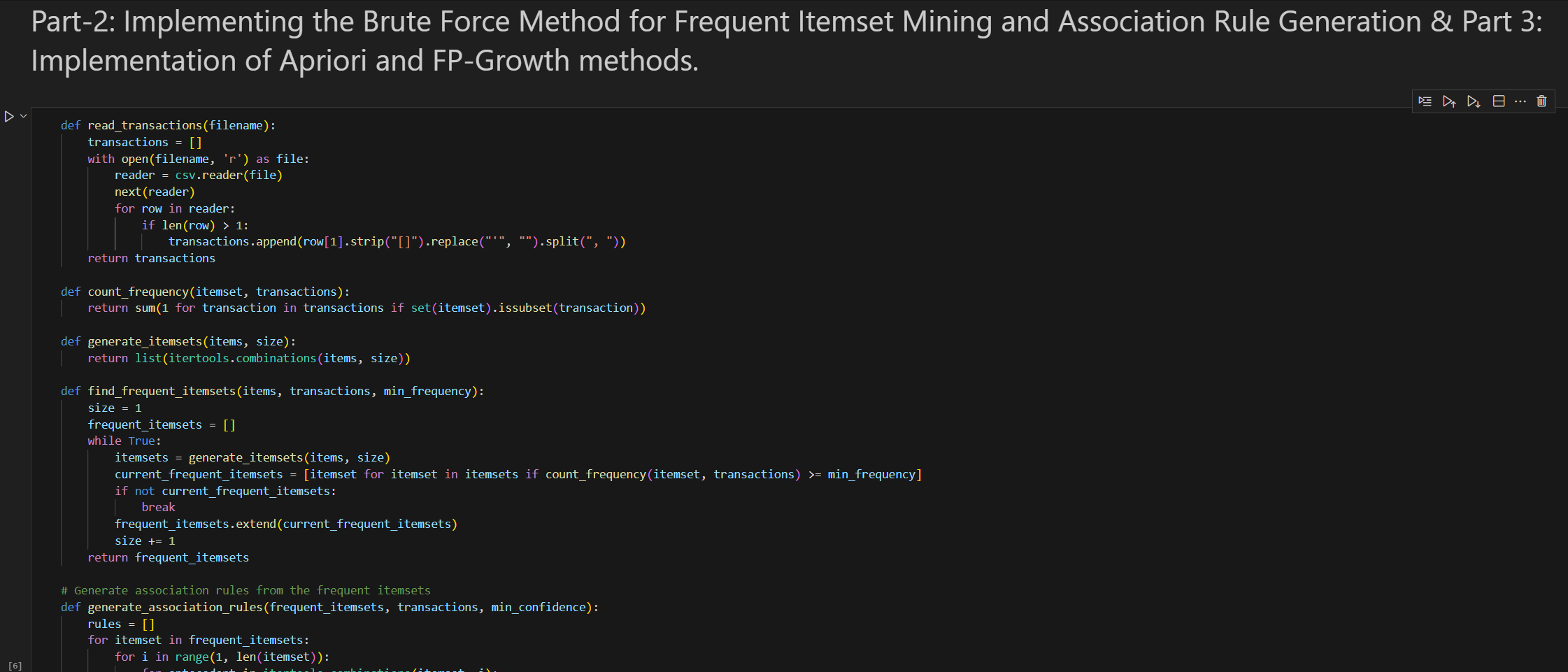
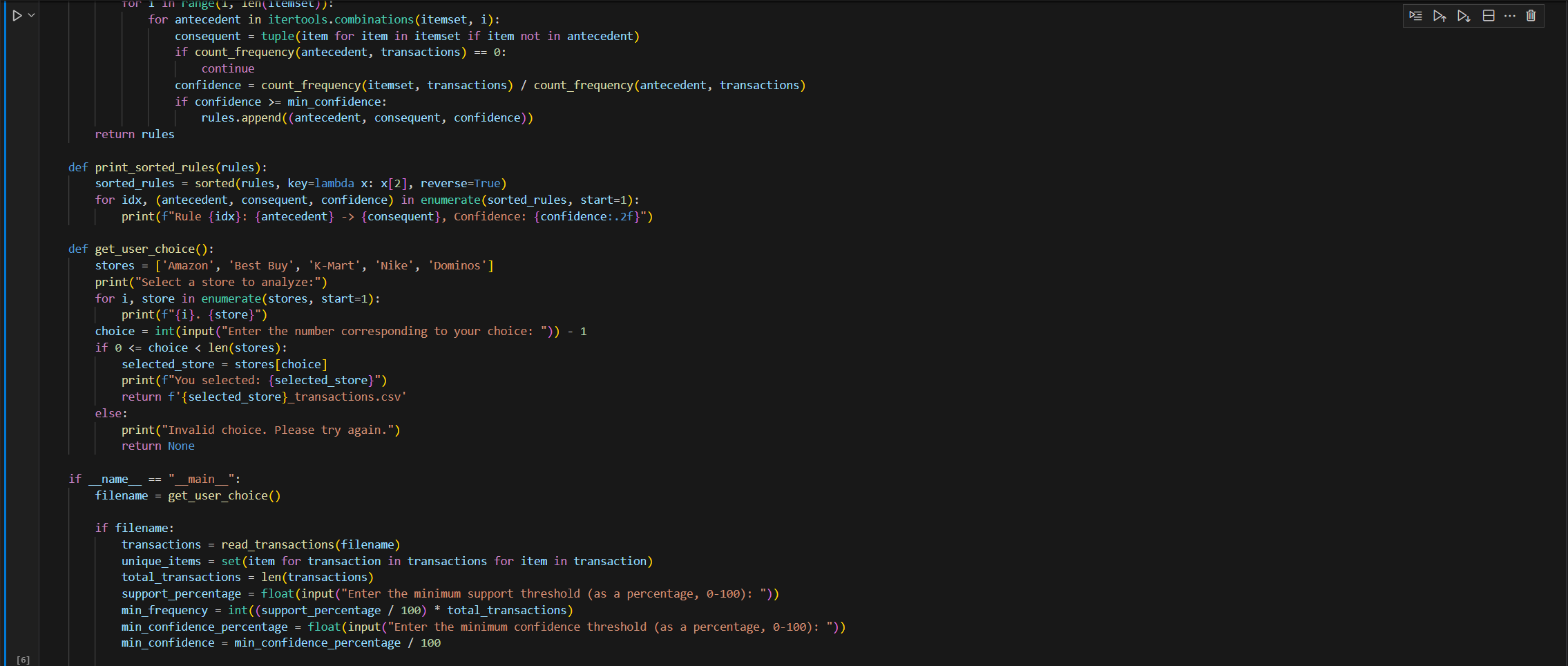


Figure: Databases Creation





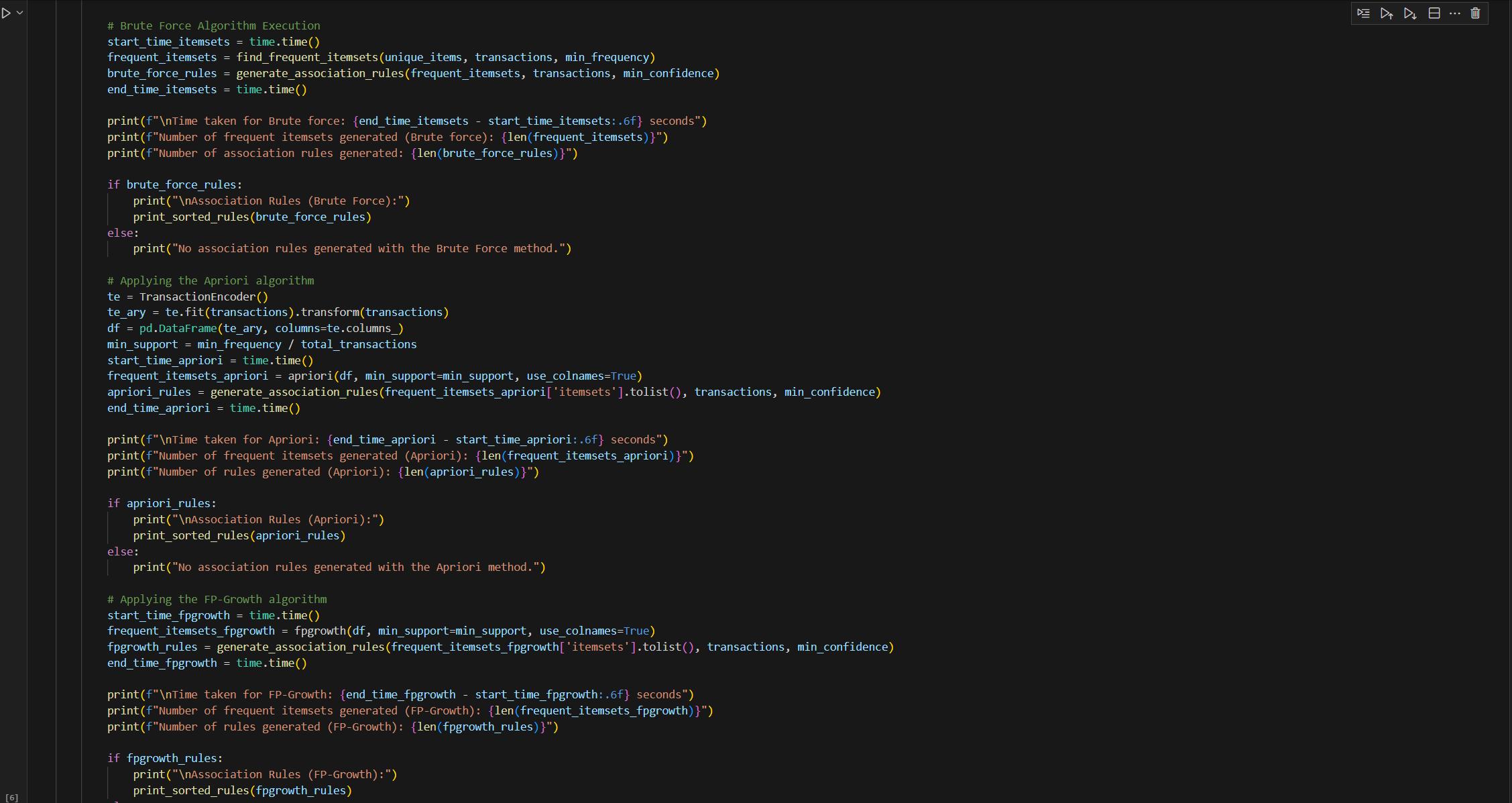


Figure: Algorithms

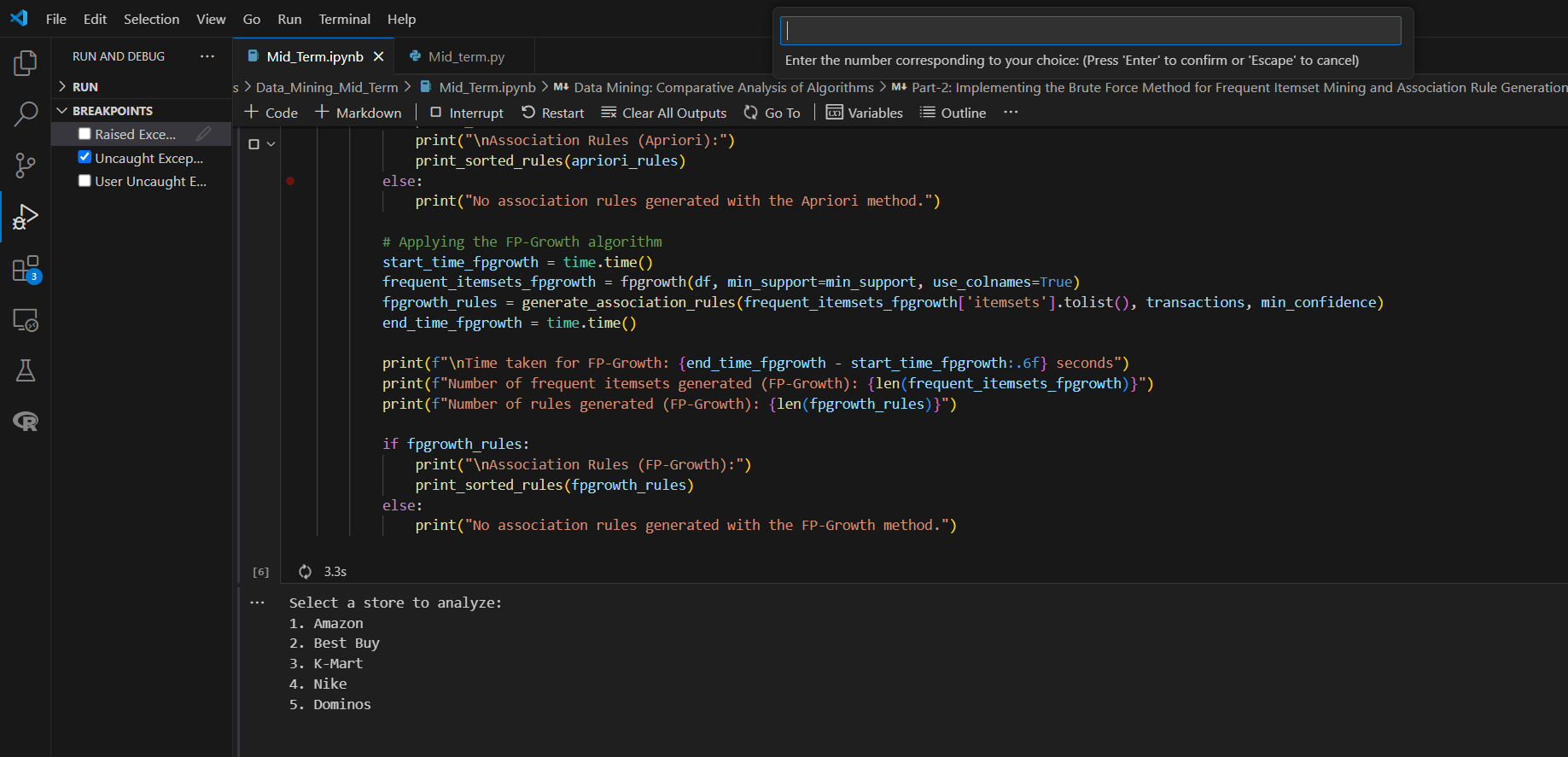


Figure: Taking User Input for the Store

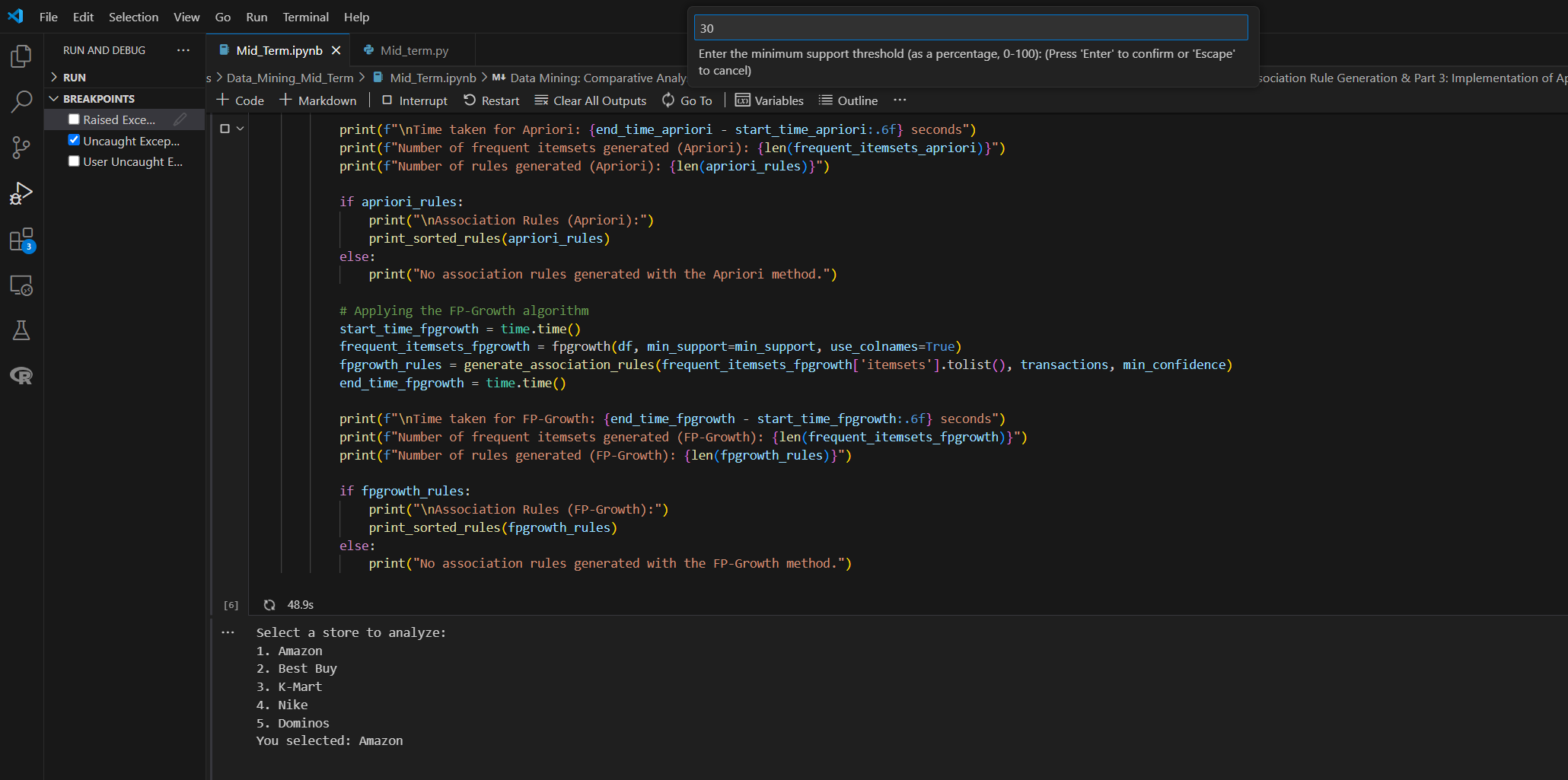


Figure: Taking user input for the Minimum Support Threshold

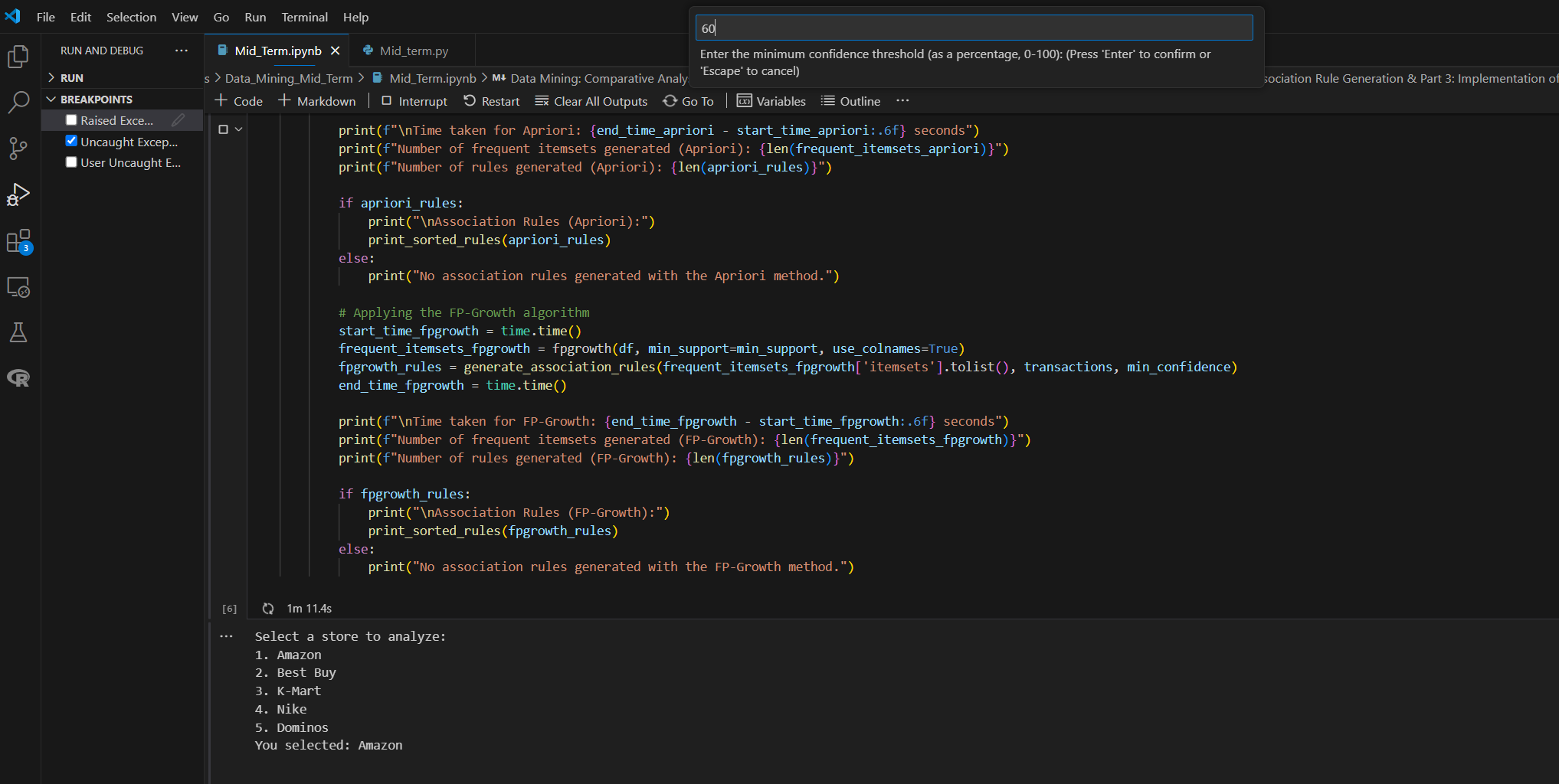


Figure: Taking User Input for the Minimum Confidence Threshold

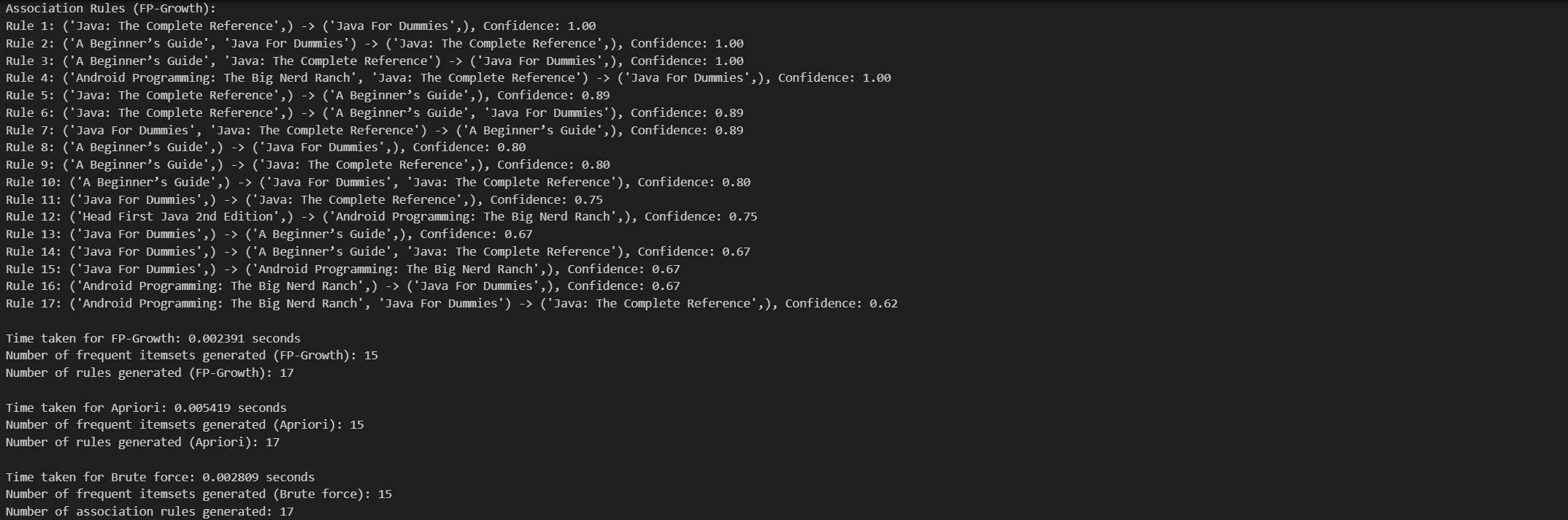
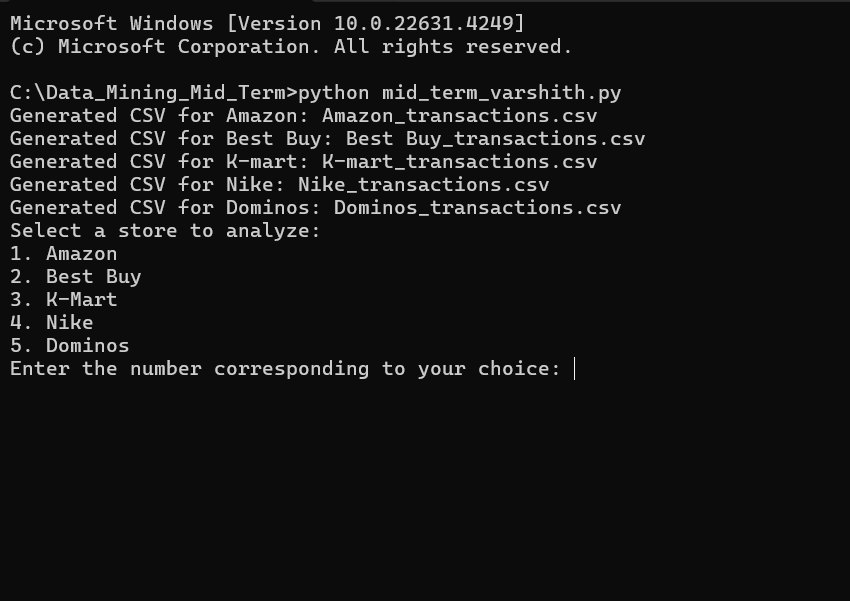


Figure: Output

### Screenshots From Terminal Running Source Code:



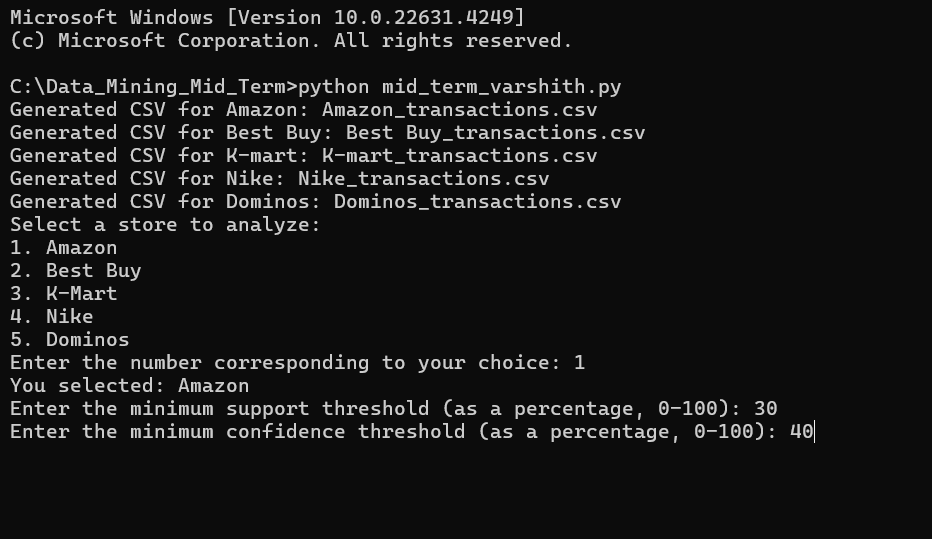


Figure: User Inputs

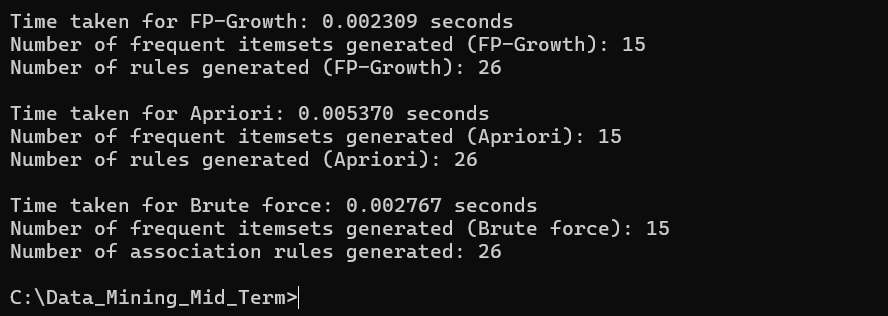


Figure: Output

### Steps To Run the Code:

**For .ipynb file**

1. **Run the Code**: Click on "Run All" to execute the program.
2. **Select Store**: When prompted, select a store by entering the corresponding number (1-5) from the following options: (1.Amazon, 2.BestBuy,3.K-Mart, 4.Nike, 5.Domino’s).
3. **Minimum Support Threshold**: Enter the minimum support threshold as a percentage (0-100).
4. **Minimum Confidence Threshold**: Enter the minimum confidence threshold as a percentage (0-100).
5. **Review Output**: After inputting the values, the program will execute and display the results for frequent itemsets and association rules for each algorithm.

For .**py** file

1. Open command Prompt or Terminal in the work folder.
2. And then execute “python Varshith\_MidtermProj.py”
3. **Select Store**: When prompted, select a store by entering the corresponding number (1-5) from the following options: (1.Amazon, 2.BestBuy,3.K-Mart, 4.Nike, 5.Domino’s).
4. **Minimum Support Threshold**: Enter the minimum support threshold as a percentage (0-100).
5. **Minimum Confidence Threshold**: Enter the minimum confidence threshold as a percentage (0-100).
6. **Review Output**: After inputting the values, the program will execute and display the results for frequent itemsets and association rules for each algorithm.

Others: LINK TO GITHUB REPOSITORY.