**Section 1: Load Balancing**

Internal models typically process each store independently of other stores, and computational

resource requirements for each store increase based on the number of transactions recorded.

We aim to parallelize these models by processing multiple stores concurrently i.e.

multithreading.

Given a user defined parameter k, write an algorithm to partition the dataset into k chunks

such that:

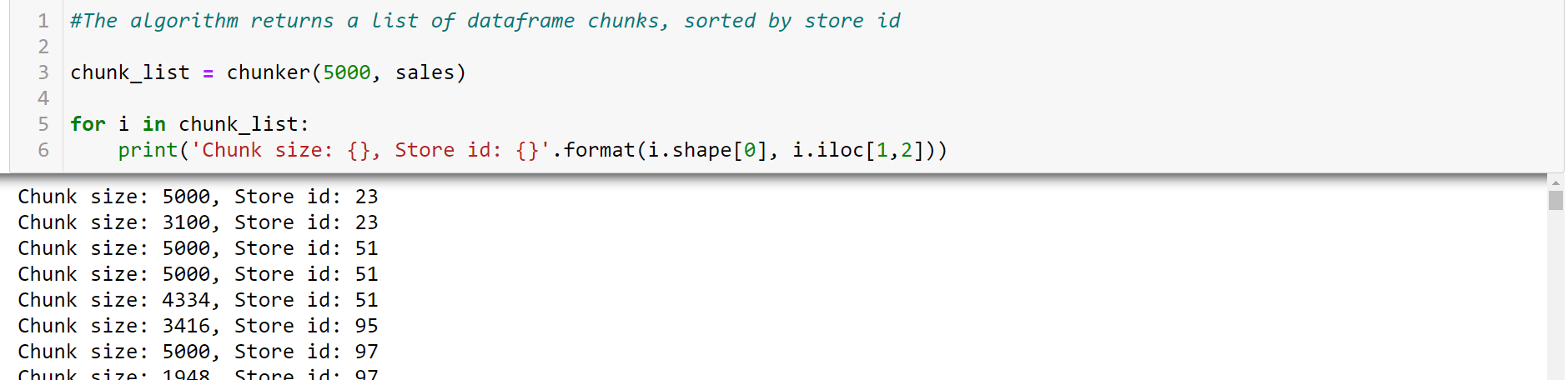
- Transactions for one store are in the same chunk

- Each chunk has approximately the same total number of transactions

**Answer:**

In the coding solution I came up with, I first grouped the data by the store\_id to separate different stores. Each store then has it’s data cut in pieces of 5000 and no chunk has data from multiple stores. For example, since store 23 has 8100 data points, the first chunk is from store 23 and of size 5000, second chunk is store 23 and size 3100, third chunk is from store 51 and size 5000 etc. The code can be seen in the screengrabs from the notebook attached below.





**Section 2: Shopper Analytics – Store Segmentation & Association Rules**

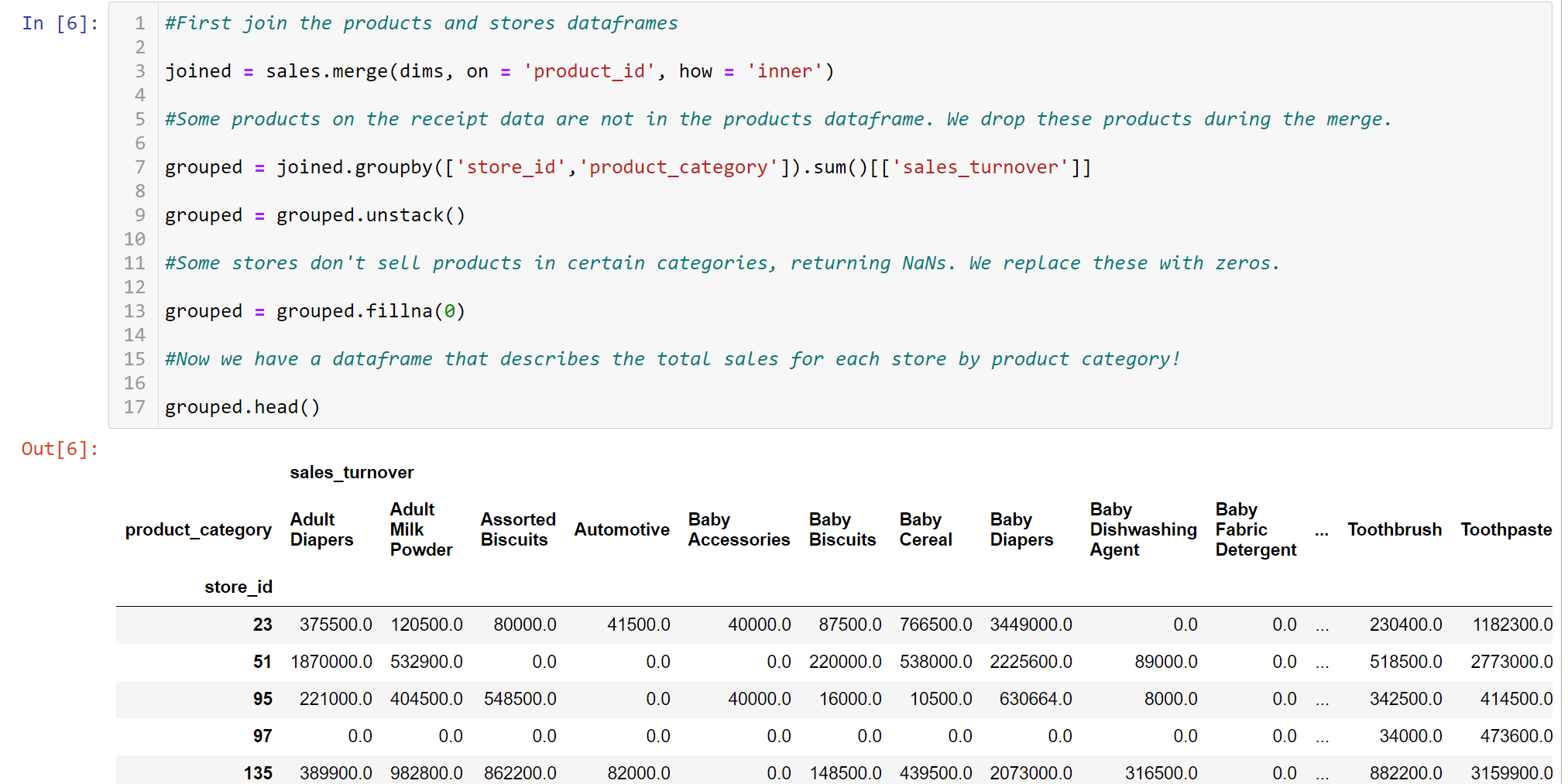
1. Clients would like to understand the types of stores that are present in the panel. Arrange

stores into mutually exclusive groups and describe the groups. Code libraries are allowed

for this question.

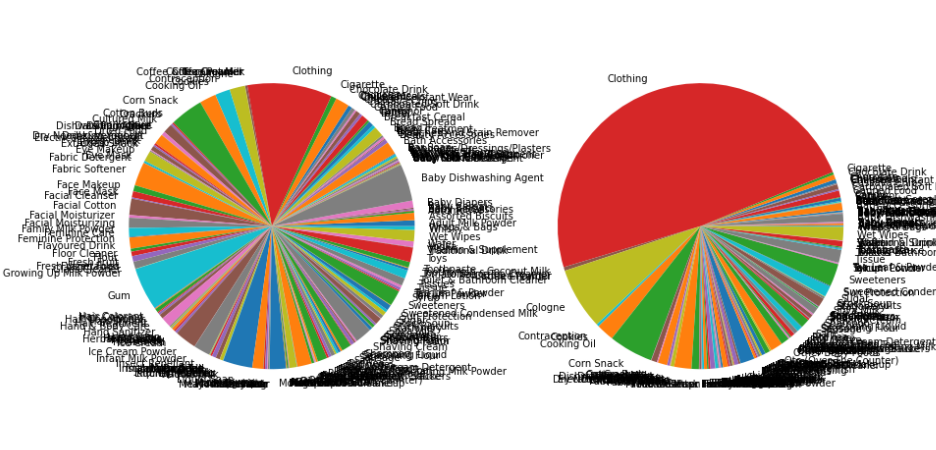
**Answer:**

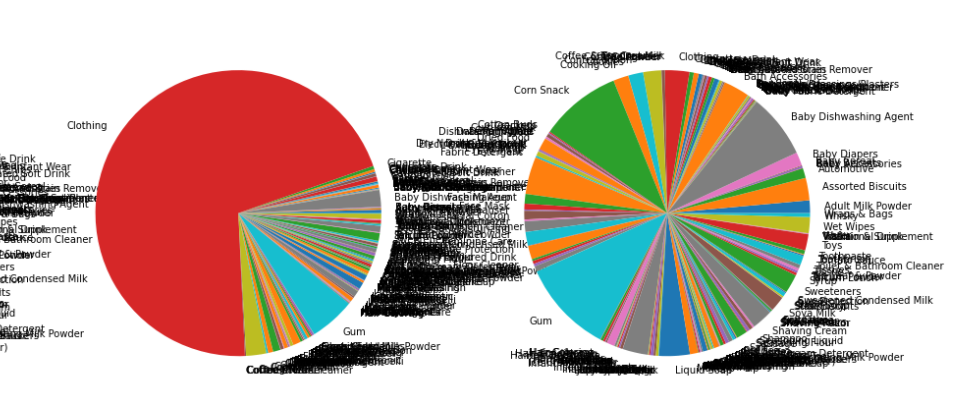
I first joined the two dataframes (one containing product information and another, receipt information). Some products that appeared on the receipt data did not have the corresponding metadata on product category, etc. and were excluded from the analysis. After moving things around we have a dataframe that shows the total sales of each store, separated by product category.

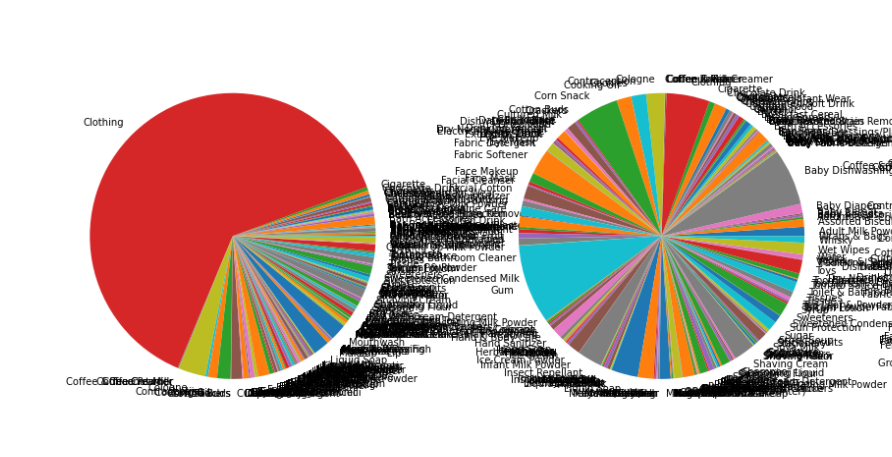


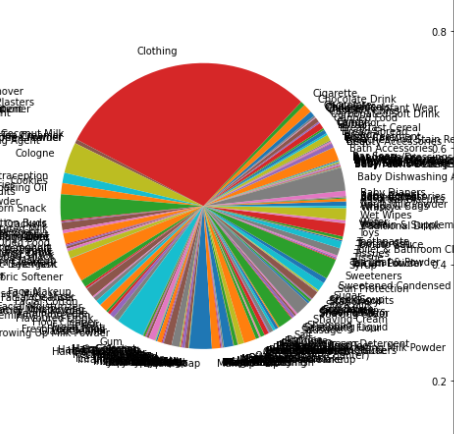
I then used an unsupervised K-means clustering algorithm to try and find hidden clusters or patterns to separate stores based on their distribution of product sales. I selected the number of clusters based by cross-validation and the optimal number of clusters selected was 7.

For each of the clusters I then plotted a pie chart for each cluster centroid to visualize what the product sales distributions were like for each cluster. The pie charts are shown below:









I apologize for the messy labels as I couldn’t figure out how to get only the labels I wanted to appear… However it is clear that although there are 7 clusters, there appear to be 2 main types of stores. One type of store gets most of its revenue through the sale of clothing items, while the other type gets its revenue through a balanced distribution of different product categories.

**2.**

**These store owners are interested to identify cross-selling opportunities in their stores via 1**

**to 1 product associations. However, due to limited resources, we are only able to select two**

**stores.**

**a. Select two stores from the dataset in which you have identified as ready for a pilot**

**study and provide your rationales.**

For the 1-to-1 product association pilot, i selected two stores belonging to two clusters with the most distinct difference in sales revenue make-up: cluster 4 and cluster 3. The store from cluster 4 gets it's sales revenue mostly from clothing, and the one from 3 has sales balanced across different categories.

The store chosen from cluster 3 has the store\_id 294, and the one from cluster 4 has store\_id 23. These two stores were chosen from the cluster as they are considered large enough for the study (receipts > 5000).

**b. Identify opportunities via 1-1 product associations (if\_bought\_this\_sales\_item\_id ->**

**likely\_to\_buy\_this\_sales\_item\_id), evaluate and rank the opportunities based on**

**relevant metrics. Code libraries used should only be limited to linear algebra and**

**dataframe manipulation.**

In this section I attempted to build an algorithm that compares prior and posterior probabilities for pairs of items. I wanted to build something that could calculate the probability (Bought A | Bought B), and to compare that with Probability (Bought A), in order to see whether having bought product B, a customer would be more or less likely to buy product A. The conditional probability (Bought A | Bought B) was decomposed into (Bought A and B) / (Bought B) to make the coding simpler.

As the algorithm would have to check the conditional probability for all item pairs, which would take too long, I only ran the algorithm for item pairs where at least one item was responsible for >1% of the number of purchases in the store.

I built the algorithm in a general series of steps listed below:

1: Define a function to check, for one receipt, if two items were purchased together

2: Define a function to check, for one receipt, if an item was purchased without the other

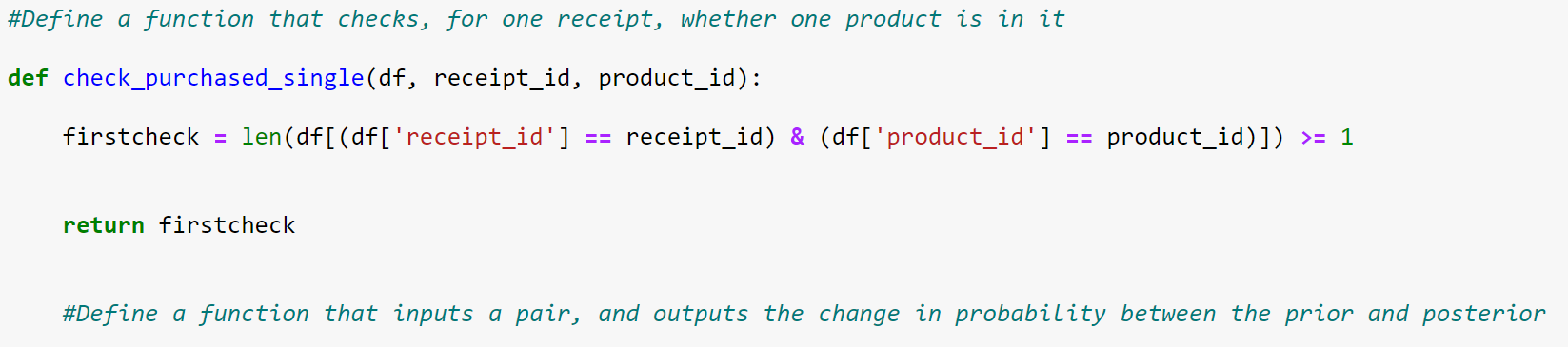
3: Define a function to return all item pairs where at least one item was significant enough for consideration (responsible for >1% store purchases)

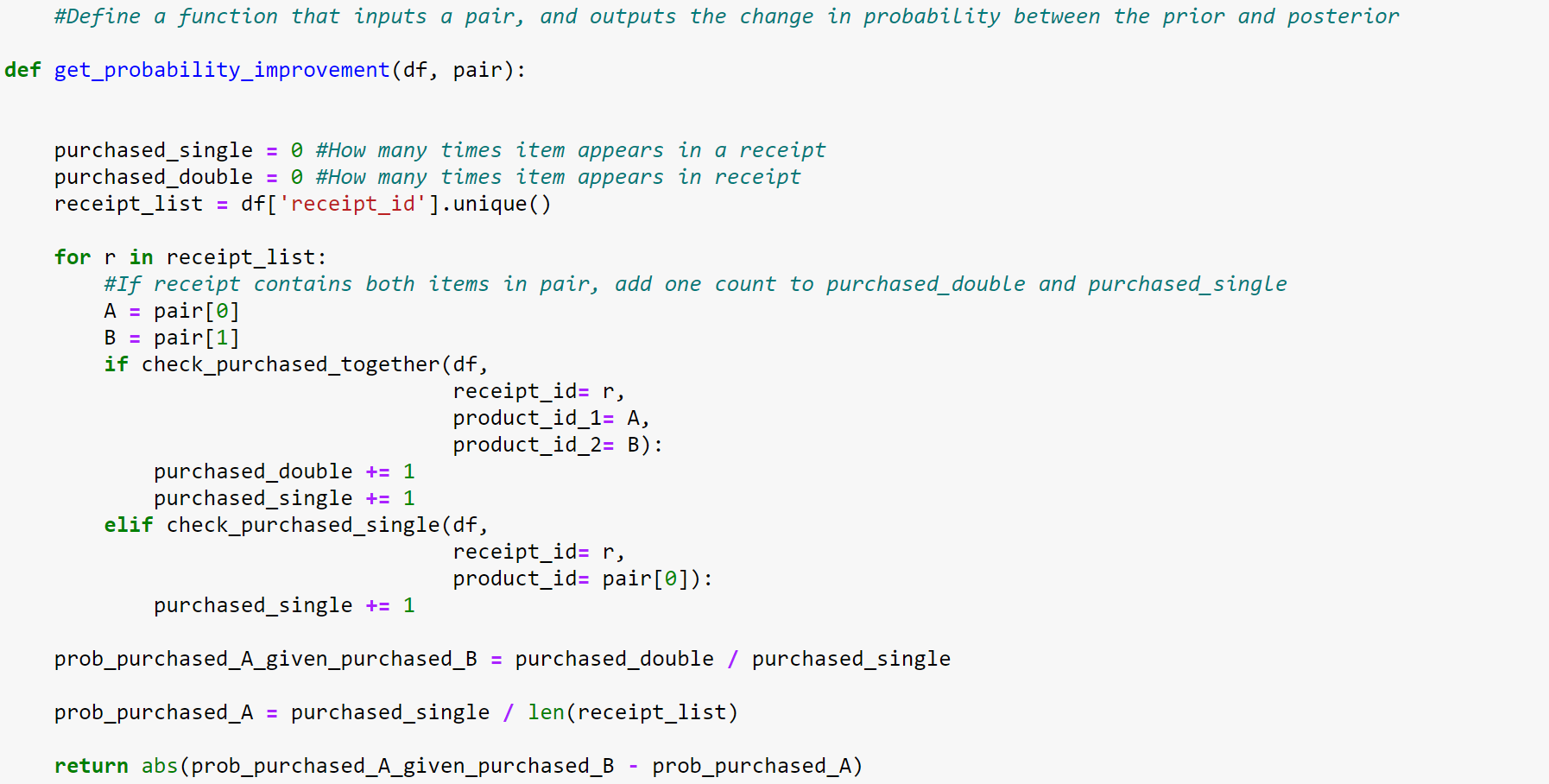
4: Define a function to get, for one pair, the improvement in probability points between the prior probability, P(Bought A), and the posterior, P(Bought A | Bought B).

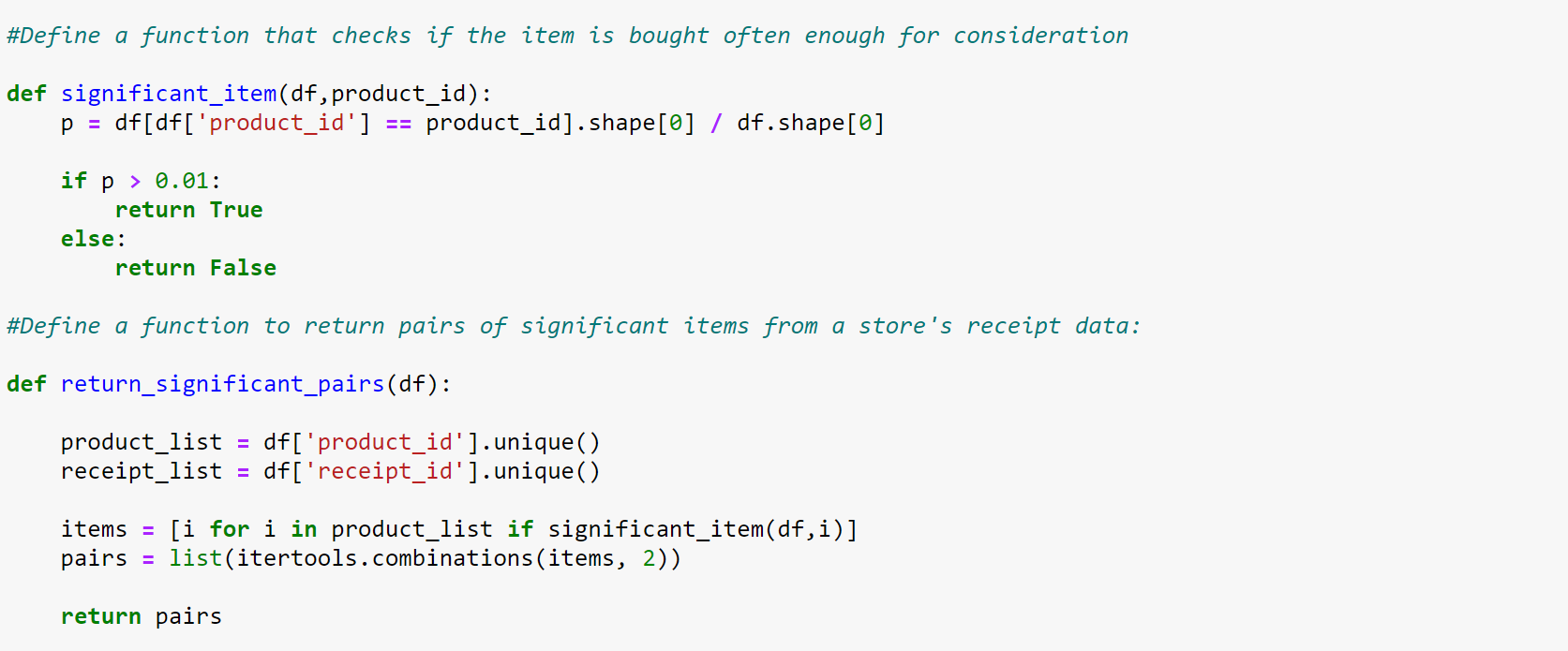
5: Define a function to calculate this improvement for all significant item pairs in a store. The total improvement in probability points is the metric by which I measured the opportunity the store has to benefit from performing analysis on 1-1 product association.

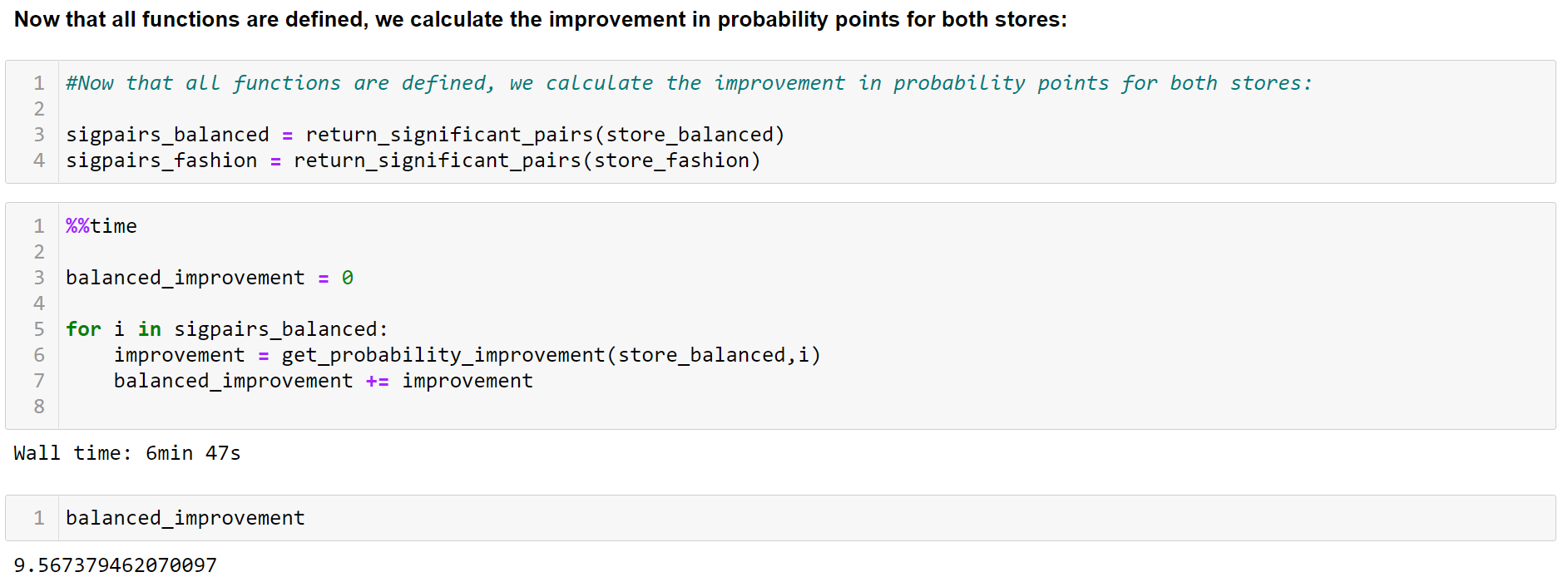
The code is shown below in a series of screenshots. I apologize for the font being quite small, so please take a look at my notebook if it is difficult to read here!













Based on the results, we can see that the improvement in probability points for the "balanced" store (Cluster 1) is 9.567, significantly higher than the improvement for the "clothes-focused" store (Cluster 4) at 0.99. This means that for the balanced store, having the right data on whether a customer bought certain products would give much more information on whether he would buy another product than the same analysis for the clothes-focused store.

A downside of this algorithm process was that the calculations for the improvement for the clothes-focused store came only from one item-pair, as no other items were significant enough to be considered in the algorithm (item must be responsible for at least 1% of store sales), hence the improvement of 0.99 (a perfect prediction for one pair). The algorithm is also not very efficient. The algorithm takes 7 minutes to run for a store of about 10,000 receipt transactions and 20 significant item pairs, and does not seem very scalable for very large datasets. It would likely be more efficient to use implementations of matrix multiplication and matrix properties instead of for loops in my functions.

The threshold of select significant item pairs (an item must be involved in >1% of transactions) is also not scalable to very large organisations, as it is unlikely that an item would meet that threshold if the total store catalogue is very large. In this case the threshold should be lowered to an appropriate number, maybe 0.1% or 0.05%.

**c. Compare the differences in the opportunities between both stores and share your**

**findings.**

There is definitely a larger opportunity to increase sales through analysing 1-1 product associations in the store that obtains revenue through a balanced distribution of goods, compared to the store that obtains revenue through mostly clothing.

This might be because clothing items tend to have many different variations of the same product type (50 different kinds of shirts in a store), so finding significant associations between specific designs may be unlikely. A better analysis in this type of store might be to compare 1-1 product type associations (whether buying a shirt makes it more likely to buy a pair of shorts) instead of looking for associations between the individual products themselves.