Naïve data linkage implementation:
Despite both datasets containing a variety of information, only the titles were chosen for comparison because they were the only consistent feature. After some observations, in a lot of the cases products either had no description or very long descriptions which did not seem useful. Price and manufacturer were not used because a lot of products had empty entries. Every product in the google database would be compared to every amazon product using their respective titles and the comparison was performed using the similarity function, ***fuzz.partial_ratio*** from the “fuzzywuzzy” library. The function computes a similarity ratio predicated on Levenshtein distance between the two strings, which was a suitable choice due its ability to account for and reward the presence of sub-strings. The threshold match score for a pair of products to be considered matching, was initially set at 50, seeming like a good middle point between 0 and 100. However, after some testing with specific thresholds, 55 was decided. The highest scoring match over 55 for a pair of products would be considered the final match. Thus, each google product could only have one amazon match. If the match score for the amazon product was below 55, then the products would not be considered a match and therefore not appended to the dataset.

Data Linkage Evaluation
After the comparisons are complete, duplicate amazon products are still possible and the data frame of product matches containing amazon duplicates were dropped keeping only the highest scoring amazon and google product matches. A relatively high Recall score of 0.946 was recorded meaning the method managed to obtain a large subset of the truth product matches. A higher recorded precision means a higher percentage of my methods matches were correct. A very high score of 0.984 was recorded with only 2 false positives recorded out of all the matches in my dataset. However, this method is not perfect, and improvements can be made. Such an improvement may be to incorporate the descriptions of the products into the match score. The descriptions of matched products would tend to have long substrings of one another in each description in which the ***ratcliff_obershelp*** function from the "text distance” library could be suitable as it analyses the longest common substrings recursively. The score produced by this function along with the match score produced by the partial ratio function can both decide whether two products from the amazon and google database are considered a match. Overall, the method performed well according to the measures.

Blocking implementation
All the titles in the google dataset were added to one string. Then using **nltk** library along with the ***word_tokenize*** function, all the "stop words " contained in the google string were removed, since they would not be useful to form blocking keys. Only the title was chosen to create blocking keys because it was the only consistent feature, same reason as task 1a. A list of tuples was created containing the words from the string (blocking key) and their frequency count, the top 500 most frequent words were finalized as the blocking keys. The google dataset was used for this because it contained far more products than the amazon dataset (more than double the size) and thus a good choice to get a good estimation of the more frequent words that could be possible for blocking. For each dataset, the product id’s and titles were retrieved and each word in the products title was compared to the blocking keys in the list of tuples. If a word in the title matched one of the blocking keys, the product would be allocated to that block and thus the product id and block key would be appended to the dataset

Blocking Evaluation
The method scored a high Pair completeness of 0.814 to 3 decimal places meaning that the blocking method was able to correctly assign two matching products to the same block for 81.4% of the total true product matches. Reduction ratio basically measures how good the blocking method is at reducing the number of records to be compared. A very high Reduction Ratio of 0.941 was recorded meaning that the blocking method reduces the number of comparisons well. A potential improvement of the method could be to instead of getting the first 500 words, the blocking keys could be any word that occurs more than k times in the google title string e.g k = 5 and tests can be run to find the most optimal k value that performs the best. Another potential improvement is to instead compare the first half of words in the title to the blocking keys instead of comparing the whole title which would theoretically half the time taken for the blocking method. The time taken for the algorithm is linear with the number of products in the datasets. This is because if a new record was to be added, the record is being compared against all of the blocking keys in the list of tuples, in which only a constant ammount of new comparisons occur because the number of blocking keys do not change.