Task 1C –

1. Product comparison is done by comparing each string in ‘title’ column in amazon\_small.csv to each string in ‘name’ column in google\_small.csv, the number of comparisons are #titles \* #name. Before comparing, the titles and names are preprocessed by removing characters such as ‘/,-,(), ., :, +’ and stopwords (unmeaningful words in English by nltk library) in strings from both csv files. This is to ensure the similarity function calculation is accurate. For example, the word ‘microsoft.’ would be different from ‘microsoft’ and common words such as ‘I am’ is not as meaningful as ‘microsoft word’. Inside the comparison loop, similarities between two strings are calculated using ‘Jaccard similarity’ function. The ‘Jaccard Similarity’ is defined as ‘Number of common unique words in both strings / (# unique words in ‘title’ string + #unique words in ‘name’ string – Number of common unique words in both strings)’. This function has advantages such as the score is not affected by number of repeated words, and the metric also suits the data since both ‘title’ and ‘name’ strings are ‘keywords’ of a product, and if they have more common words, they’re more likely to refer to the same product. For every ‘title’, the maximum Jaccard scores found by comparing to ‘names’ will be stored (if all the similarity scores for a title are 0, these titles will not be recorded). If these maximum scores pass the threshold ‘0.22’, it is assumed to be a true match and written to csv. The threshold is 0.22 because 1-2 common words especially in long sentences are inadequate to conclude they refer to the same product. Using this algorithm, recall found is 0.9230769230769231, while precision is 0.916030534351145. It accurately matches 120 rows due to Jaccard Similarity function, while 10 pairs that should be matched are not matched due to 0.22 threshold. There are also 11 pairs that aren’t supposed to be matched due to them exceeding threshold but these words aren’t meaningful and perhaps due to preprocessing without stemming and lemmatization.

2. This algorithm links correctly most articles with highest common words in their titles and names, while having O(n^2) comparison is not very effective (can be reduced by blocking). The Jaccard similarity gives a fairly accurate value given the words are preprocessed beforehand. However, highest similar score does not always mean that they refer to the same product. To avoid this, we can further check the *uncommon* words in title/name if they exist in the description. Since this algorithm takes one largest similarity value, this fails when title in first csv refers to more than a title in second csv. Instead of storing the last ID with maximum score, we can store all values with same scores and extract all the match combinations. It also fails when two articles are the same with no common words in titles and names, therefore checking description may be useful. When there are no similarities in all the table keys when these articles should be linked together, revising the data collection should be done. Stemming and lemmatization words such as ‘studies’ to ‘study’ is needed for more accurate Jaccard score, as well as removing verbs such as ‘learn, because ‘learn to microsoft’ is different from ‘learn to win’. Missing titles/names are rare, but we could impute titles/names found in description.

3. The blocks are based on prices found in ‘titles/names’ column. Preprocessing is done by removing alphabets and $. The block names are steps of 25$ starting from 0, such as 0-24$, 25-49$ and so on. For example, a price of 27.88 will be in block 25-49. Each ID from amazon.csv and google.csv are then assigned to these blocks and exported to csv.

4. In overall, the blocking method has 64.3% Pair Completeness, with 836 matches are in the same block assuming 100% match, while 464 pairs that should be matched are never in the same block, which is a trade-off for having less comparisons. The Reduction Ratio calculated is 82%, which means the number of comparisons is only 20% out of comparing all possible in csv. The complexity of assigning to blocks takes linear time since we are iterating over the column price and immediately assigning each ID to the price range. The time complexity for blocking comparisons is better compared to naïve data linkage’s O(n^2) if the price is widely-distributed, since the blocking depends solely on price range. If most prices are located in the same range, we will need to do more comparisons. The worst case is if there is only one block, we would need to compare all records. Linking products according to the same price might not be accurate, as we got 464 pairs not located in the same blocks, this is because the price in amazon and google are most likely to be different even if they refer to the same product. A solution to this is to create another key for the blocking, for example by using the combination between manufacturer and price or product key and price. The blocking method is also not viable if there are lots of missing prices (such as 0$), instead we can create a new column called ‘product key’ containing product possible names of each item taken from title.