Blocking

- First, we loaded the two tables A (Amazon.csv) and B (Barnes.csv) into Magellan.

- Next, we pre-processed the two tables as the records were not clean. Pre-processing involved:
    1. Title attribute values had edition embedded in them. We cleaned these values and extracted edition from them.
    2. ISBN-13 attribute values were not in the same format in both the tables. We cleaned the ISBN-13 attribute values to bring them in the same format.
    3. Author attribute in each table is a set-valued attribute but the values were not properly separated/delimited. We cleaned Author attribute values in each table and made them in the same format by adding a # as the delimiter.

- Next, we want to perform blocking. But before that we created golden data so that we can measure the goodness (i.e., how many matching pairs are retained) of blocking. To create the golden data we performed attribute-equivalence based blocking on ISBN-13 attribute. This resulted in golden data G having 492 matching pairs. We manually went through these 492 pairs to ensure that they are indeed matching. We also used Magellan's debugger to see that we did not lose any matching pairs in this blocking.

- Next, we tried a cheap built-in blocker: **Overlap blocker**. We applied **one word** overlap on Author attribute.

Table A:	8600+ records
Table B:	9000+ records
Candidate set C (after blocking Table A & Table B):	1210156 records
No. of matching pairs retained after blocking (i.e., $C \cap G$):	492

- Next, we tried to reduce the size of the candidate set by again applying **Overlap blocker** on C using **one word** overlap on Title attribute.

Candidate set C:	1210156 records
Candidate set D (after blocking candidate set C):	339597 records
No. of matching pairs retained after blocking (i.e., $D \cap G$):	492

- The size of the candidate set D is still quite large and we wanted to reduce it to under 100K pairs. So, we applied a **rule-based blocker** on D using the following rule:
Title_Title_jac_qgm_3_qgm_3 (ltuple,rtuple) < 0.3

This rule says that drop a pair if the Jaccard similarity (with 3-gram tokenization) on the titles is less than 0.3. We settled in with a low threshold of 0.3 (after trying several thresholds such as 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, …) because this resulted in a candidate set that had a reasonable size as well as retained a high number of matching pairs. We had also tried another rule on Publisher attribute but since this attribute has a lot of missing values (given by null), we did not get a lot of reduction on candidate set size. So, we used the rule on Title as mentioned before.

Candidate set D:	339597 records
Candidate set E (after blocking candidate set D):	4198 records
No. of matching pairs retained after blocking (i.e., $E \cap G$):	488

- Candidate set E is our final candidate set for this part of the project.

Feedback on Magellan:
Overall we would rate our experience with Magellan as "good". However, we faced several issues while using this system. We briefly mention them below:

1. **Data type issue:** Initially we did not know how to specify the data types for the attributes in the tables. So, our ISBN-13 attribute in one table was treated as numeric and Magellan would show it in the format 9.78 E12. The ISBN-13 attribute in the other table was shown in the 13-digit format (maybe because all the values were in double quotes). As a result we were not to able to perform attribute-equivalence blocking on ISBN-13 to generate golden data. We thank Pradap for helping us resolve this.

2. **Need for a GUI while editing table:** If we have to edit a value in a table, we have to open the file in an editor (outside Magellan), edit the file and then reload it in Magellan. Having a GUI to support such edits will be really helpful. We mention this as an issue because only during blocking we realized that our records were not clean and we had to preprocess (as mentioned before in the document) several times before we could actually perform good blocking.

3. **Setting JVM path:** If we have to use similarity measures other than Jaccard and Levenshtein, we have to initialize a JVM. We were unable to initialize the JVM because Magellan was not picking the correct path to the JDK installed in our system. The solution to this issue was pretty simple (again we thank Pradap for this) but as we do not have much experience with Python-based systems, we were not able to figure that out.