Report:

Sentence 1: "The Big Data platform for students is Blackboard"

Sentence 2: "Questions on MinHash project by NTNU students is on Piazza"

Sentence 3: "NTNU Big Data platform are Blackboard and Piazza"

Sentence 4: "The project data for students are on Blackboard not Piazza"

1.1)

1. ["big", "data", "platform", "students", "blackboard"]
2. ["questions", "minhash", "project", "ntnu", "students", "piazza"]
3. ["ntnu", "big", "data", "platform", "blackboard", "piazza"]
4. ["project", "data", "students", "blackboard", "piazza"]

1.2)

I'm assuming we have removed the specified types of words (articles, auxiliary verbs, prepositions, and conjunctions) and converted all words to lowercase.

["big", "data", "platform", "students", "blackboard", "questions", "minhash", "project", "ntnu", "piazza"]

1.3)

Sentence	big	data	platform	students	blackboard	questions	minhash	project	ntnu	piazza
1	1	1	1	1	1	0	0	0	0	0

2	0	0	0	1	0	1	1	1	1	1
3	1	1	1	0	1	0	0	0	1	1
4	0	1	0	1	1	0	0	1	0	1

realised the matrix above don't scale well if you open it in another editor or screensize, so input matrix is also below in different format.

```
[1, 0, 1, 0]  # big
[1, 0, 1, 1]  # blackboard
[1, 0, 1, 1]  # data
[0, 1, 0, 0]  # minhash
[0, 1, 1, 0]  # ntnu
[0, 1, 1, 1]  # piazza
[1, 0, 1, 0]  # platform
[0, 1, 0, 1]  # project
[0, 1, 0, 0]  # questions
[1, 1, 0, 1]  # students
```

Jaccard similarities are: {'1-2': 0.1, '1-3': 0.5714285714285714, '1-4': 0.42857142857142855, '2-3': 0.2, '2-4': 0.375, '3-4': 0.375}.

Sentences 1 and 3 are most similar, with a Jaccard similarity of 0.571

1.4)

MinHash Signature Matrix:

```
[[1 2 1 2]
```

```
 [1 2 1 2]
 [2 1 2 3]]
```

Similarity (Sentence 1, Sentence 2): They agree on 0 out of 3 hash functions.

Similarity (Sentence 1, Sentence 3): They agree on 3 out of 3 hash functions.

Similarity (Sentence 1, Sentence 4): They agree on 0 out of 3 hash functions.

Similarity (Sentence 2, Sentence 3): They agree on 0 out of 3 hash functions.

Similarity (Sentence 2, Sentence 4): They agree on 2 out of 3 hash functions.

Similarity (Sentence 3, Sentence 4): They agree on 0 out of 3 hash functions.

The most similar sentence pairs are:

Sentence 1 and Sentence 3

Both pairs have a similarity score of 3/3.

the result 1 & 3 is consistent from earlier.

Results from task 3:

```
Data reading...
2225 documents were read in 0.27927541732788086 sec

Starting to calculate the similarities of documents...
Calculating the similarities of 2474200 combinations of documents took 40.714067459106445 sec

Starting to create all k-shingles of the documents...
Representing documents with k-shingles took 0.5583624839782715 sec
```

```
Starting to create the signatures of the documents...
Signatures representation took 1.4184162616729736 sec

Starting to simulate the MinHash Signature Matrix...
Simulation of MinHash Signature Matrix took 264.1419196128845 sec

Starting the Locality-Sensitive Hashing...
LSH took 0.11357331275939941 sec

Starting to get the pairs of documents with over  0.6 % similarity...
The total number of candidate pairs from LSH: 169
The total number of true pairs from LSH: 166
The total number of false positives from LSH: 3
Naive similarity calculation took 40.714067459106445 sec
LSH process took in total -0.0010187625885009766 sec
```

Report questions:

1: by increasing the parameter k, we decrease the total numbver of shingles. Because as we get longer shingles, or bigger shingles, with k. Meaning we have more letters/words in a shingle. this gives us fewer unique shingles generated in a document. This also means that reducing k leads to more shingles.

Having more shingles may not be better to identify pairs of document with similarities over the threshold. While larger number of shingles might capture more fine grained details on the content, it might introduce noice and increase overhead during similarity calculations. An optimal value of k should be chosen based on a trade off between efficiency and granularity.

2: Increasing the number of permutations will generally reduce the number of false positives. This is because a higher number of permutations will improve the accuracy of the Minhash algorithm for approximating the Jaccard similarity between documents. As we get more permutations, the probability of collisions decreases, which leads to fewer false positives.

3: The optimal value of bands will depend on a trade-off on the number of comparisons made and the efficiency on computation. Increasing the number of bands will reduce the number of comparisons needed, as each band performs independent hasinh. Resulting in less candidate pairs to compate. However, increasing b will also increase the probability of false positives due to collision in the hash buckets.

4.

The total number of candidate pairs from LSH: 169, and the LSH took 0.113573312759399141 sec.

Naive Method:

Number of candidate pairs checked: 2474200 (out of 2474200 combinations)

Execution time: 47.78489422798157 seconds

We can see the LSH method is extremely more efficient than the naive method. Both in terms of the amount of documents checked and run time. it offers significant time savings and computational efficiency. Especially for larger datasets.