1)

- *What is the maximum number of record matches?* (m)
- *What is the corresponding number of non-matches in this circumstance?* (n − m)
- *Now suppose a blocking methodology using b blocks is employed.*
- *How many comparisons will be needed in the best case?* (None: e.g. consider if all records in first database are in block 1, all records in second database are in block 2, all other blocks are empty (and we have a flag knowing if a block is empty)
- *How many comparisons will be needed in the worst case?* When b=1: (m × n)
- *How many comparisons will be needed on average? (state any assumptions)* assuming uniform distribution across blocks (m/b×n/b×b) [assuming records are uniformly spread across the blocks. In first database blocks have m/b records, in second database they have n/b records. So m*/b*n/b comparisons for each block. Then multiply by number of blocks
- *What is the advantage of using large b?* fast (but inaccurate)
- *What is the advantage of using small b?* accurate (but slow)

2) Can answer based on knowledge from question 1. If the two datasets are very large, then the number of comparisons without blocking will be m*n If the size of each is 1,000,000, then number of comparisons is $1*10^{12}$. Compare this against a blocking strategy, where on average it is (m/b×n/b×b) and if block size is 10000, then comparisons are $1*10^{12}/(10^4)=10^8$. For a good blocking strategy (uniform spread across blocks) then accuracy may be almost as good as strategy without blocking, but much, much faster.

3)

Explain how the data duplicate detection problem relates to the data linkage problem. (Data duplicate detection refers to finding records in the same dataset that belong to the same entity. For example in a medical dataset if a patient changed their name, we would like the history to link back to the same user. Thus in structured databases (or csv's) we could have two records from the same patient with different names although having identical data otherwise,

What are the major differences? (When we match in the same dataset it is more likely that we have to do fuzzy (approximate) matching - as in the name change example. There could also be privacy issues if the matching is not done properly. How do we do matching without snooping at data?).

4i) What are the reasons two records could be linked incorrectly? (They do not correspond to the same entity. E.g. two persons with accidentally the same name, but not the same date of birth)

4ii) Suppose a false positive (FP) is two records that are linked by the system, which a human believes should not have been linked. Suppose a false negative (FN) is when two records are not linked by the system, which a human believes should have been linked. A true positive (TP) is two records linked by the system which a human believes should have been linked and a true negative (TN) is two records not linked by the system which a human believes should not have been linked.

– *What are the relative sizes of the categories TP, TN, FP, FN*? (TP m TN n-m FP and FN ≪ m) *In practice, how might one calculate these sizes?*

Construct the confusion matrix

	Actual:Link=true	Actual:Link=false
Prediction:link=true	Small (TP)	Small (FP)
Prediction: link=false	Small (FN)	Very large (TN)

- *It is desirable to minimise both FP and FN, but it may be difficult to minimise both simultaneously. Give an example application where minimising FP is more important than minimising FN*.
-
- Centrelink scenario? Don`t want bogus matches between centrelink benefit record and employer income record because it brings grief to the person mistakenly sent a letter claiming money is owed (Centrelink seemed to be minimizing FN though!)
-
- *Give an example application where minimising FN is more important than minimising FP* (mailing list – target people with email about alumni opportunities, based on similarity of their name to historical enrolment list. Want to maximise the number of people reached (Reduce FN) and less concerned about emailing someone wrongly (FP)