(1)
a) Describe the algorithms basic idea.
The algorithm first creates a reverse index for all values in the tables. This allows to check in which columns a certain value exists.
For each column we say that all other columns are included in this column.
Then for each value in the inverted index we iterate over all columns in which the value exists. For each colmn we remove all references to columns which are not part of the columns for the current value.
Finally we iterate over all columns and check whether any columns are included in the current column. If this is the case, we emit the corresponding inclusion dependency.

b) If you used an algorithm from literature, provide a reference to the according publication.

We implemented the IND discovery algorithm from the IND slide 28.

c) Provide one or two arguments why it is or could be better than related algorithms.

The algorithm uses the inverted index to reduce the number of scanning over every inoput column to 1. This is better than a naive brute force approach.

d) If your algorithm implements an adption of optimization of existing approaches, describe these briefly.

No adaption in the implementation.

(2)
a) How many valid, unary inclusion dependencies did your algorithm find on the provided datasets?

Input data set	Super INC	Binder
TPC H	Out of Memory Exception	85
TPC H w/o line item & orders	38	38
WDC	112	112

b) How long did it take?

Used hardware: 2,6 Ghz Intel Core i7, 16GB DDR 3, Mac OS 10.12

Input data set	Super INC	Binder
TPC H	Out of Memory Exception	14:09s
TPC H w/o line item & orders	654ms	312ms
WDC	16ms	637ms

c) Did you discover any limitations of your approach?

Our approach is limited by the available main memory as we store the whole inverted index in main memory. Therefore, the approach is limited by the size of all unique values in all tables.

d) Why is it a good idea to ignore NULL values in IND discovery?
The values of columns of a join should be also in the columns of the original table. This is not the case, if columns without NULL values are joined an outer join. If an outer join is materialized and checked for inclusion dependencies with the original tables some inclusion dependencies are not found, if NULL values are not ignored. Therefore, NULL values should be ignored.