Supervision 1 Questions

1. Discuss the practical exercises in :
   https://www.cl.cam.ac.uk/teaching/2021/Databases/relational-tick.html

   Please include your answer to every question with a short description of the query and your solution

2. A relational implementation of an Entity-Relationship (ER) model typically attempts to avoid **redundancy**. But what does this mean exactly?
   Is **redundancy** the same thing as **duplication** of data values?

3. Construct an Entity-Relationship model for the following scenario. Suppose we are conducting several experiments. Each experiment has a name and a description (text). Each experiment can be associated with many *runs*. Each run is associated with some input parameters and some output values.

   For intuition, consider the following example. Suppose our experiments use *simulation* to explore the behavior of a distributed algorithm where nodes exchange messages with their neighbours and eventually compute some result. Each experiment might be associated with a different network topology such as linear sequence of n nodes, a ring of n nodes, an n by m grid of nodes, a clique of n, a binary tree of nodes with depth n, so on. The input parameters for each run might be a seed for a random number generator and one or more size-related parameters (depending on the experiment). The output of any run might be the total number of messages exchanged by the distributed algorithm and the time needed for termination. Each run of an experiment will update our database and our database should be able to support SQL queries that summarise the results.

   Yes, the specification is somewhat vague. Intentionally so! You may find that your are forced to make some simplifying assumptions in order to make progress.

4. Discuss possible relational implementations of your model from above.

5. Suppose we have an experiment called "grid" that has input parameters "random_seed", "grid_width", and "grid_height" with output parameters "message_count" and "run_time". For fixed values of "grid_width" and "grid_height" we have run thousands of experiments just varying the "random_seed" value. Using your implementation from (4), we now want to write an SQL query that groups all runs of "grid" by "grid_width" and "grid_height" and returns for each group the average for each of the outputs. Recall that the average is computed by the aggregate function **AVG**.