There is a conceptual design to logical design flaw in Assignment 5 that will require a workaround.

In the table schema given, in the Product relation, the model attribute is conceptually related to both the desktop and the laptop tables. However, when creating the Product table, you will find out that you are not allowed to have the foreign key "model" reference both the desktop and the laptop tables. A foreign key can only reference a single table, not multiple tables.

There are different ways that this can be fixed, as outlined in this answer/response set: see http://stackoverflow.com/questions/7844460/foreign-key-to-multiple-tables for possible solutions

Feel free to implement whichever workaround seems best to you, but the official work around I am providing is this:

Device (model, type, speed, RAM, HD, list_price, screen) // where primary key is model and type, type = desktop or laptop, and there is a constraint (semantic if necessary) where screen is null if type = desktop
Manufacturer (name, country, phone)
Product (*manu_Name, *model, *type)

I chose this due to its simplicity and because there is only one column difference between desktops and laptops (screen size) - if there were only a few columns in common, with many different specific to the device, then I'd have gone with having 3 tables: a device table (as a parent) and the individual desktop/laptop tables (so, for example, if we also had a printer device, then perhaps only the model and type would exist in the device table and all other attributes would be in the individual tables) - in that case, the Product table would use the model/type from the Device table as the foreign keys (so only referencing one table), not from the individual laptop/desktop/printer tables. That is certainly also a valid option for this assignment, particularly if you note that the business case appear to often have queries for just one of the device types, and it may speed things up to have smaller, dedicated tables.

Your choice - go for simplicity in table design and a little more complexity in your query design - or go for complexity in table design with the ability to only interact with a table with fewer rows for many of the queries.

In any case, include a couple of sentences to say why you went with the design you did and make sure that you use constraints/triggers where necessary to ensure valid data and consistency. It is ok to treat the “screen is null if type=desktop” as a semantic constraint (not checked by the db, but maintained by the person inserting data).