Technical University of Denmark

Department of Applied Mathematics and Computer Science

TEST exam 2024 *based on* the exam in 2022.

02170 Database Systems – Written Examination Part 1

Examination date: 17th May 2022
Course title: Database Systems
Course number: 02170
Aids allowed: All
Weighting: Part 1: 50 %, Part 2: 50 %.
The weighting is only indicative and will ultimately be decided during the evaluation.

This document together with the files ***answers2022.sql*** and ***create-takeaway-db-without-constraints.sql*** constitutes Part 1 of the written examination.
Part 2 is a digital multiple choice assignment which should be accessed separately.

Your answers to the questions in this document should be inserted at the right place in the ***answers2022.sql*** file. There must at most be one answer to each question. Remember to write your student ID at the top of the file.

Feel free to add any comments. Comments must be placed in SQL comments, i.e. just after two hyphens (--) without line breaks.

The file ***create-takeaway-db-without-constraints.sql*** can be used to create the database instance shown in this document. Note that referential integrity constraints are *deliberately omitted* from *create-takeaway-db-without-constraints.sql*, and they must be added for the database to be consistent.

When you have completed the ***answers2022.sql*** file, you should submit it digitally on eksamen.dtu.dk.

A database for a Takeaway Shop:

In both parts of this examination a database named *Takeaway* is considered. The database contains information about food items, customers and orders made in a takeaway shop. Below you find the relation schemas of the database and a relation instance of each of these, and some explanations are given.

FoodCategory(catId, catName)

catId	catName
bev	Beverages
fish	Fish dishes
meat	Meat dishes
start	Starters

FoodItem(itemId, description, catId, unitPrice)

itemId	description	catId	unitPrice
bburg	Big Burger	meat	100
nburg	Normal Burger	meat	75
sroll	Spring Rolls	start	45
sushi	Sushi Menu	fish	65

Customer(custNo, name)

custNo	name
1	Peter Pan
2	Daisy Duck
7	James Bond

FoodOrder(orderNo, custNo)

orderNo	custNo
1	1
2	1
3	7
4	7

OrderLine(orderNo, lineNo, itemId, quantity, unitPrice)

orderNo	lineNo	itemId	quantity	unitPrice
1	1	sroll	2	45
1	2	nburg	5	75
2	1	sushi	1	65
2	2	nburg	2	75
3	1	nburg	3	75

Food items and food categories: The shop sells food items belonging to various food categories. A food category has a unique id *catId* and a name *catName*. The *FoodCategory* table describes the possible food categories. Each food item has a unique id *itemId*, a *description*, and a *unitPrice* (i.e. the price for one food item), and it belongs to one of the possible food categories specified by a *catId*. The *FoodItem* table describes the food items sold by the shop.

Customers: Before a customer can place orders in the shop, the customer must be registered with a unique customer number *custNo* and a *name* in the *Customer* table.

Orders and order lines: Registered customers can place orders by requesting some food items and stating the quantity of each food item. When a customer has placed an order, the order is given a unique number *orderNo*. Furthermore, information about that order is stored in the *FoodOrder* table and the *OrderLine* table: In the *FoodOrder* table it is registered which customer placed that order. In the *OrderLine* table information about the ordered food items is registered: Each order has a collection of order lines. Each of these order lines has (besides the *orderNo)* a line number *lineNo* (which is unique within that order, but may be used for other orders as well), the *itemId* of an ordered item, the *quantity* ordered, and the *unitPrice* charged per item.

Question 1 State an SQL query, which returns a table that for each food category has a row giving its *catId*, *catName* and the highest *unitPrice* of all of the food items that belong to that food category.

Question 2 State an SQL query, which returns a table containing the *itemId* and *description* of food items that have never been ordered.

Question 3 Define an SQL function named *total_cost_for_customer*, that with a customer number as argument, will return the total cost of all orders made by the customer. The return type should be INT. If the function is given a non-existing customer number or a number of a customer who has not ordered any food items, it should return NULL (without making any special error handling).

Question 4 State an SQL query, which uses the function to return a table that for each registered customer has a row giving the customer's *custNo* and the total cost of all orders made by the customer. The total cost should be NULL, if the customer has not ordered any food items.

Question 5 State an SQL query, which returns a table containing the *catId*, the *catName* and the number of associated food items for each food category that has more than one food item associated.