Introduction to Databases
Coursework 2 (SQL)

Dr Paolo Guagliardo

v1.2 (09/11/2021)

1 Database Schema

For this assignment we use a schema that models a typical business scenario where customers place orders consisting of several products in different quantities, and these orders are invoiced and paid for. The SQL schema declaration is in the file `schema.sql` published on Learn; please familiarise yourself with it before writing your queries.

Observe that you cannot make any assumptions besides what is allowed by the schema; in particular, you cannot make assumptions based on the data contained in any test instance that may be made available to help you work on the assignment. Your queries will be executed on different instances (of the same schema) on which such assumptions may not hold. The only assumptions you can rely on are those enforced by the integrity constraints of the schema, with which all instances must comply.

2 Assignment

The task is to write an SQL query for each specification given below. *The order and types of the columns in the output table are important*, but their names are not (as long as they comply with the other requirements). Each query must return the relevant columns in the specified order and with the specified types (use **`CAST`** if needed) as indicated in the last line of the corresponding query specification.

(01) **Orders per country between 2017 and 2020** [10 marks]

For each country, calculate the total number of orders placed by customers of that country in the years between 2017 and 2020 (extremes included). If all customers from a specific country did not place any orders in the given period, the country appears in the output with 0 total orders. Return the code of each country and the corresponding number of orders.

country code (`CHAR(3)`), *total orders* (`INT`)

(02) **Refunds for overpaid invoices** [8 marks]

Find invoices that have been overpaid, keeping in mind that there may be more than one payment towards the same invoice. For each overpaid invoice, return the invoice ID and the amount that should be reimbursed.

invoice ID (`INT`), *refund amount* (`NUMERIC`)

(03) **Date of the most recent order (or orders) of each customer** [8 marks]

For each customer, return their ID and name, and the date of their last order. For customers who never placed any orders, no rows must be returned. For each customer who placed more than one order on the date of their most recent order, only one row must be returned.

customer ID (`INT`), *customer name* (`TEXT`), *order date* (`DATE`)

(04) **Single-type orders** [12 marks]

Find orders that consist of *only one* type of products.[1] Return the order ID, the date on which it was placed, and the type of products it contains.

order ID (`INT`), *order date* (`DATE`), *product type* (`VARCHAR`)

(05) **Avid readers** [12 marks]

Find customers who, across all of their orders, have purchased more than twice as many books as other types of products. The number of times a product is ordered must take into account the quantity in which it appears in the order details. For each such customer, return their ID and the total number of books they purchased.

customer ID (`INT`), *total books* (`INT`)

(06) **Popular products within their type** [12 marks]

Find products that were ordered 50% more times than the average of all *other* products of the same type. The number of times a product is ordered must take into account the quantity in which it appears in the order details. For each such product, return its code and the total number of times it was ordered.

product code (`INT`), *times ordered* (`INT`)

(07) **Products usually ordered in bulk** [8 marks]

Find products whose quantity in the details of the orders in which they are included is, on average, at least 3. For each such product, return its code and price.

product code (`INT`), *product price* (`NUMERIC`)

(08) **Longest wait for next order** [16 marks]

For each customer who placed at least two orders, calculate the longest number of days ever elapsed between any one of their orders and the next. The only case when this number (which can be computed as the difference of two dates) will be zero is for customers who placed at least two orders and *all of their orders* were placed on the same day. Return the customer ID and the corresponding number of days (as an integer).

customer ID (`INT`), *number of days* (`INT`)

(09) **Recurring customers** [14 marks]

Consider all customers who have never placed two distinct orders on the same date, and who have placed at least 10 orders in total. Among such customers, find those for which the interval between each of their orders (except the last) and the next is less than 7 days on average.

For each such customer, return their ID and name, and the total number of orders they have placed.

customer ID (`INT`), *customer name* (`TEXT`), *total orders* (`INT`)

[1] This implies that orders without a detail are excluded.