1 Test Orders

Please visit http://sqlfiddle.com/#!17/5a0147 to answer the questions. Put your answers on the right hand side of the screen. The URL should change when you enter the answers, please send us back the URL once you have provided all the answers.

1. With the table test_orders, write an SQL query that calculates the first(!) date (YYYY-MM-DD) on which a customer's cumulative purchase_revenue has reached or exceeded 100€. Output columns: date (DATE), customer_id (INTEGER)

2. With the table "test_orders", write an SQL query that returns only customers whose cumulative purchases are above 100€. Output columns: customer_id (INTEGER)

3. With the table "test_orders", write an SQL query that calculates the number of days between the first and last purchase for each customer and the revenue per day in this timespan. Output columns: customer_id (INTEGER), days_between_last_and_first_purchase (INTEGER), revenue_per_day (FLOAT)

4. Table "test_orders" stores the purchase_revenue based on the individual items in table "test_order_details". This means that purchase_price times number_items for all sold items should equal test_orders.purchase_revenue. Write a query to verify each individual purchase, making sure that the condition purchase_revenue = number_items*purchase for sold items holds true. If this verification fails, output the difference between the (purchase_items*number_items) - purchase_revenue. Output columns: customer_id (INTEGER), order_id (INTEGER), verified (BOOLEAN, True if numbers match, otherwise False), difference (FLOAT, NULL If numbers match, otherwise the difference)

5. With table "test_order_details" write a query that calculates per customer how much additional money could have been made if the customer had not returned items, the original amount the customer spent, the amount that would have made if no items were returned and the percentage increase between the original revenue and the revenue if no items would have been returned. Output columns: customer_id (INTEGER), extra_revenue (FLOAT), original_revenue (FLOAT), new_revenue (FLOAT), percentage_increase (FLOAT)

6. Based on question 5, which customer would you pay the most / least attention to reduce the number of returned items and why?

7. Based on question 4, how could you improve table "test_orders" to simplify the JOIN between "test_orders" and "test_order_details"?