*Task 6. Prepare answers to the following questions*

***6.1. What operations do the following functions perform: film\_in\_stock, film\_not\_in\_stock, inventory\_in\_stock, get\_customer\_balance, inventory\_held\_by\_customer, rewards\_report, last\_day? You can find these functions in dvd\_rental database.***

film\_in\_stock - this function helps to retrieve a list of inventory\_id with film\_id, in which it is stored in store and is in the stock.

If the no match then returns "No films in stock matching".

film\_not\_in\_stock - shows inventory item with film\_id and in which store it is but out of stock.

inventory\_in\_stock - checks if inventory item is currently available to rent.

get\_customer\_balance- calculates customer balance using late fees, payments made by a customer.

inventory\_held\_by\_customer - this function tells you whether a customer still holds an item. If function returns NULL, the item was returned already.

rewards\_report - this function shows customer rows for all customers who spent more than $min\_dollar\_amount\_purchased and made more than min\_monthly\_purchases

last\_day - returns the last day of month by by calculating the first day of the next month and subtracting 1 day.

***6.2. Why does ‘rewards\_report’ function return 0 rows? Correct and recreate the function, so that it's able to return rows properly.***

I thought the issue was due to the date (since the current date is 2025, so no data in that date range (payment\_date)), but even after modifying the code to return values from 2017, I was not successful.

***6.3. Is there any function that can potentially be removed from the dvd\_rental codebase? If so, which one and why?***

I think it's film\_in\_stock and film\_not\_in\_stock, because these functions duplicate logic that's already handled by inventory\_in\_stock. The inventory\_in\_stock function does a better job, as it checks whether a specific inventory item is in stock. film\_in\_stock simply returns inventory IDs of a given film at a store that are in stock, which can easily be replaced by a SELECT statement with a WHERE inventory\_in\_stock(...) condition.

***6.4.The ‘get\_customer\_balance’ function describes the business requirements for calculating the client balance. Unfortunately, not all of them are implemented in this function.***

***Try to change function using the requirements from the comments.***

/\*The requirement "IF A FILM IS MORE THAN RENTAL\_DURATION \* 2 OVERDUE, CHARGE THE REPLACEMENT\_COST" is missing so we need to add

to DECLARE

v\_rentfees DECIMAL(5,2); -- Initial rental fees

v\_overfees INTEGER; -- $1 per day overdue

v\_replacementfees DECIMAL(5,2); -- Replacement fees for severely overdue films

v\_payments DECIMAL(5,2); -- Payments made

\*/

**DROP** **FUNCTION** public.get\_customer\_balance(**int4**, **timestamptz**);

**CREATE** **OR** **REPLACE** **FUNCTION** public.get\_customer\_balance(p\_customer\_id **integer**, p\_effective\_date **timestamp** **with** **time** **zone**)

**RETURNS** **numeric**

**LANGUAGE** plpgsql

**AS** **$function$**

--#OK, WE NEED TO CALCULATE THE CURRENT BALANCE GIVEN A CUSTOMER\_ID AND A DATE

--#THAT WE WANT THE BALANCE TO BE EFFECTIVE FOR. THE BALANCE IS:

--# 1) RENTAL FEES FOR ALL PREVIOUS RENTALS

--# 2) ONE DOLLAR FOR EVERY DAY THE PREVIOUS RENTALS ARE OVERDUE

--# 3) IF A FILM IS MORE THAN RENTAL\_DURATION \* 2 OVERDUE, CHARGE THE REPLACEMENT\_COST

--# 4) SUBTRACT ALL PAYMENTS MADE BEFORE THE DATE SPECIFIED

**DECLARE**

v\_rentfees **DECIMAL**(5,2); --#FEES PAID TO RENT THE VIDEOS INITIALLY

v\_overfees **INTEGER**; --#LATE FEES FOR PRIOR RENTALS

v\_replacementfees **DECIMAL**(5,2);--#SUM OF PAYMENTS MADE PREVIOUSLY

v\_payments **DECIMAL**(5,2); --#SUM OF PAYMENTS MADE PREVIOUSLY

**BEGIN**

**SELECT** **COALESCE**(**SUM**(film.rental\_rate),0) **INTO** v\_rentfees

**FROM** film, inventory, rental

**WHERE** film.film\_id = inventory.film\_id

**AND** inventory.inventory\_id = rental.inventory\_id

**AND** rental.rental\_date <= p\_effective\_date

**AND** rental.customer\_id = p\_customer\_id;

**SELECT** **COALESCE**(**SUM**(**CASE**

**WHEN** (rental.return\_date - rental.rental\_date) > (film.rental\_duration \* '1 day'::**interval**)

**THEN** **EXTRACT**(epoch **FROM** ((rental.return\_date - rental.rental\_date) - (film.rental\_duration \* '1 day'::**interval**)))::**INTEGER** / 86400 -- \* 1 dollar

**ELSE** 0

**END**),0)

**INTO** v\_overfees

**FROM** rental, inventory, film

**WHERE** film.film\_id = inventory.film\_id

**AND** inventory.inventory\_id = rental.inventory\_id

**AND** rental.rental\_date <= p\_effective\_date

**AND** rental.customer\_id = p\_customer\_id;

**SELECT** **COALESCE**(**SUM**(**CASE**

**WHEN** (rental.return\_date - rental.rental\_date) > (film.rental\_duration \* '2 days'::**interval**)

**THEN** film.replacement\_cost

**ELSE** 0

**END**), 0)

**INTO** v\_replacementfees

**FROM** rental, inventory, film

**WHERE** film.film\_id = inventory.film\_id

**AND** inventory.inventory\_id = rental.inventory\_id

**AND** rental.rental\_date <= p\_effective\_date

**AND** rental.customer\_id = p\_customer\_id;

**SELECT** **COALESCE**(**SUM**(payment.amount),0) **INTO** v\_payments

**FROM** payment

**WHERE** payment.payment\_date <= p\_effective\_date

**AND** payment.customer\_id = p\_customer\_id;

**RETURN** v\_rentfees + v\_overfees - v\_payments;

**END**

**$function$**

;

***6.5. How do ‘group\_concat’ and ‘\_group\_concat’ functions work? (database creation script might help) Where are they used?***

'group\_concat' is used in CREATE OR REPLACE AGGREGATE function which combines

text values from multiple rows into a single string, separated by commas. It is used to calculate values across mutilpe rows.

'\_group\_concat' function takes two text values. If one of them is missing (NULL),

it returns the other one. If both are present, it joins them with a comma and a space.

'\_group\_concat' is used inside the 'group\_concat' aggregate function, so while '\_group\_concat' checks values inside,

‘group\_concat’ checks across rows. Function - to transform a single input, aggregate - to combine multiple inouts using function.

6.6. ***What does ‘last\_updated’ function do? Where is it used?***

This function helps to add current update date, so I when I use trigger "last\_updated", function adds current date when record was updated, changed. It is used when I need to automatically track the latest update date, so I do not need to write an UPDATE every time, for example, for when I need to know when customer status was changed or when some changes was made to the data.

***6.7. What is tmpSQL variable for in ‘rewards\_report’ function?***

***Can this function be recreated without EXECUTE statement and dynamic SQL? Why?***

It is a variable used to hold a SQL command as text, which is later run using EXECUTE.

Yes, the function can be rewritten without EXECUTE because all needed values are already in the database. We can just use a regular SELECT statement instead of building SQL dynamically.