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:**

The function is designed to check the availability of a specific film in a particular store. It identifies the inventory items (copies of the film) that are currently in stock.

**film\_not\_in\_stock:**

The function film\_not\_in\_stock is similar to the film\_in\_stock function but focuses on retrieving inventory items that are not currently available

**inventory\_in\_stock:**

The function determines whether a specific inventory item (copy of a film) is currently available in stock. It checks the rental status of the item using the rental table.

**get\_customer\_balance:**

The function calculates the balance of a customer (based on rental fees, late fees, and payments) as of a specific effective\_date. The calculation considers:

* Rental Fees for all previous rentals up to the specified date.
* Late Fees for all overdue rentals.
* Replacement Cost if a film is more than twice the rental duration overdue.
* Payments made by the customer before the specified effective\_date.

**inventory\_held\_by\_customer:**

The function designed to return the customer\_id of the customer currently holding a specific inventory item (movie) that has not been returned yet.

**rewards\_report:**

The function is designed to generate a report of customers who qualify for rewards based on two criteria:

* The minimum number of monthly purchases they must have made in the last three months.
* The minimum dollar amount they must have spent in those purchases.

**last\_day:**

The last\_day function is designed to return the last day of the month for a given date with time zone (timestamp with time zone). The function works by checking the month of the input date, and then calculating the last day of that month.

1. **Why does ‘rewards\_report’ function return 0 rows? Correct and recreate the function, so that it's able to return rows properly. Is there any function that can potentially be removed from the dvd\_rental codebase? If so, which one and why?**

The rewards\_report function is returning 0 rows likely due to issues in the way the dynamic SQL query is executed or the way the conditions are being set for qualifying customers. For example the function raises exceptions if the min\_monthly\_purchases or min\_dollar\_amount\_purchased parameters are set to invalid values (0), which could prevent the function from executing, resulting 0 rows. Also, the function calculates last\_month\_start and last\_month\_end based on a "3 month" interval from the current date. This could be incorrect because it's hardcoded to always look 3 months back, and this may not match the desired reporting period.

The dynamic SQL (tmpSQL) is constructed and executed, inserting qualifying customers into a temporary table. There might be issues regarding the SUM(p.amount) and COUNT(customer\_id) conditions not properly qualifying customers, resulting in no customers being inserted into tmpCustomer. The GROUP BY clause uses customer\_id, but the criteria in HAVING involve both the sum of amounts and count of purchases. If no customers meet both conditions at the same time, the result will be empty.

The function inventory\_held\_by\_customer can be potentially be removed, as this function only returns the customer ID holding a particular inventory item if it's not returned, also its functionality can be replaced with a simple SELECT query directly in application code or other functions.

Changed the function by correcting the calculation of last\_month\_start to focus on the previous month, aligning it with the expected logic and used date\_trunc('month', CURRENT\_DATE - INTERVAL '1 month') to get the start of the previous month. Updated HAVING conditions to ensure both the number of purchases and the total amount spent are met. Removed the dynamic SQL execution (EXECUTE tmpSQL), simplifying the function. Added ON COMMIT DROP to ensure the temporary table is automatically removed at the end of the session, preventing conflicts on reruns.

**\* 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.**

Issues in the function: The current function does not charge the replacement cost when a film is overdue by more than twice its rental duration. The late fee is calculated based only on (return\_date - rental\_date) > rental\_duration, but it does not account for rentals that are still overdue (i.e., return\_date is NULL). It does not handle cases where the rental has not been returned (return\_date IS NULL), which should be considered overdue based on the current date. Replacement Cost is not included in the calculation for rentals overdue by more than rental\_duration \* 2.

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

group\_concat and \_group\_concat are used in cases where there is a need to concatenate multiple text values from grouped data into a single text string. It can be used in queries where STRING\_AGG might be unavailable or for custom behavior (like different separators or handling NULL values differently). This is typically helpful in reporting, data aggregation, and generating lists or summaries from detailed data. \_group\_concat is a helper function that concatenates two text values with a comma separator, handling NULL values.

group\_concat is an aggregate function built on top of \_group\_concat, applying it across multiple rows to concatenate a list of values.

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

Automatically updates the last\_update column with the current timestamp when a row is modified, ensuring that timestamp is always accurate and does not require manual intervention during updates. It is used in tables where modification tracking is important, such as film, inventory, or customer. It requires a trigger setup on the desired table to invoke this function during UPDATE operations.

**\* What is tmpSQL variable for in ‘rewards\_report’ function? Can this function be recreated without EXECUTE statement and dynamic SQL? Why?**

In the rewards\_report function, the variable tmpSQL is used to construct a dynamic SQL query as a text string. This dynamic SQL is then executed using the EXECUTE statement. It builds a SQL query dynamically based on the input parameters (min\_monthly\_purchases and min\_dollar\_amount\_purchased). Specifically, it inserts customer IDs into a temporary table (tmpCustomer) based on payment records that meet the specified criteria.

Rewriting the function without dynamic SQL can make it simpler and more secure, provided the requirements do not demand dynamically constructed queries.