1. A report that could be generated by the attached data sets would be an income report. The purpose for an income report is to allow stakeholders and board members to see the number of sales their operations are generating. This allows these stakeholders and board member to make informed decisions based on the amount of income coming into the operation.

A1. The data that will be used in this report will be data that will allow us to cross reference the amount of inventory that is leaving the store and currently in the store, current rental rates, and the IDs of the stores that these sales are being made at.

A2. The tables that will be used to generate the detailed report will be the customer table, inventory table, payment table, and the rental table. The tables that will be used to generate the summary report will be the inventory table, the payment table, and the rental table.

A3. The specific fields that will be included in the detailed report will be customer ids, first name, last name, and the information needed for the sale such as the store id, rental id, inventory id, and the amount of the sale/rental. The specific field that will be included in the summary report will be the information only needed to conduct the sale, such as the store id, rental id, inventory id, and the amount of the sale/rental.

A4. One field that would require custom transformation would be the payment amount from each sale at each location. By using aggregation, more importantly the SUM() transformation, it will allow you to see the total amount of sales done by the operation. This is important to allow stakeholders and board members to see a simplified report versus seeing every single transaction processed by a given location.

A5. The different business uses for the detailed report would consist of targeting customer sales metrics, allowing you to keep track of what consumer has what rental, overall income of the store, and allows you to keep track of what rentals are leaving your inventory. The different business uses of the summary report would only consist of the sales metrics, overall income, as well as the rental and inventory information.

A6. These reports should be updated monthly and then can be summed again at the end of the year to generate an annual income report. This will allow stakeholders and board members to see how the operation does month to month. It’ll show how busy a store is and allow you to forecast potential sales for the coming months based on location and popularity.

1. Detailed Sales Report Table Creation Code:

Create table InfoForSales (

customer\_id INT NOT NULL,

first\_name CHARACTER varying(45) NOT NULL,

last\_name CHARACTER varying(45) NOT NULL,

store\_id INT NOT NULL,

rental\_id INT NOT NULL,

inventory\_id INT NOT NULL,

amount NUMERIC (5,2)

);

Summary Report Table Creation Code:

Create table TotalSales(

store\_id INT NOT NULL,

rental\_id INT NOT NULL,

inventory\_id INT NOT NULL,

amount NUMERIC (5,2) NOT NULL

);

1. Data Extraction For Detailed Report:

INSERT INTO InfoForSales (

customer\_id,

first\_name,

last\_name,

store\_id,

rental\_id,

inventory\_id,

amount

)

Select c.customer\_id, c.first\_name, c.last\_name,

i.store\_id, i.inventory\_id,

r.rental\_id,

p.amount

FROM payment AS p

INNER JOIN rental AS r ON r.rental\_id = p.rental\_id

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN store AS s ON s.store\_id = i.store\_id

INNER JOIN customer AS c ON c.customer\_id = r.customer\_id;

Verification of Data Accuracy:

Select SUM(amount) from payment;

Select SUM(amount) FROM infoforsales;

1. Code for Function that provides transformation (A4) :

CREATE FUNCTION model\_totalsales ()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM totalsales;

INSERT INTO totalsales (

SELECT

store\_id, rental\_id, inventory\_id, SUM(amount)

FROM infoforsales

GROUP BY store\_id, rental\_id, inventory\_id

ORDER BY store\_id ASC);

RETURN NEW;

END; $$ LANGUAGE PLPGSQL;

1. Code for Trigger Creation:

CREATE TRIGGER totalsales\_refresh

AFTER INSERT ON infoforsales

FOR EACH STATEMENT

EXECUTE PROCEDURE model\_totalsales();

1. Code for Stored Procedure:

-- To be run on a monthly basis, last day of every month

CREATE PROCEDURE refreshed\_salesreport()

LANGUAGE PLPGSQL

AS $$

BEGIN

DELETE FROM infoforsales;

INSERT INTO infoforsales (

customer\_id,

first\_name,

last\_name,

store\_id,

rental\_id,

inventory\_id,

amount

)

SELECT

c.customer\_id, c.first\_name, c.last\_name,

i.store\_id, i.inventory\_id,

r.rental\_id,

p.amount

FROM payment as p

INNER JOIN rental AS r ON r.rental\_id= p.rental\_id

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN store AS s ON s.store\_id = i.store\_id

INNER JOIN customer AS c ON c.customer\_id = r.customer\_id;

END; $$;

F1. In order to automate the stored procedure, you would need an external tool to help schedule the procedure to run. The way I would automate this process would be to install Agent pgAgent. What this allows you to do is task this external tool with a job and control the frequency in which it does this job. Meaning that once you install the agent, task it with the job of running the stored procedure, schedule the frequency in which it runs the procedure the only thing left to do is enable it to run in the background. To do so you would need to enter in this code “/usr/bin/pgagent host=localhost dbname=postgres user=postgres port=5432 -l 1 (Dias)”. Once that is done then the process will be fully automated.

H. Web sources used for assessment:

<https://www.postgresqltutorial.com/>

<https://severalnines.com/blog/overview-job-scheduling-tools-postgresql/>

<https://www.geeksforgeeks.org/postgresql-tutorial/?ref=lbp>

1. Dias, Hugo. “An Overview of Job Scheduling Tools for Postgresql.” Severalnines, 4 May 2022, https://severalnines.com/blog/overview-job-scheduling-tools-postgresql/.