**Advanced Data Management**

**SECTION A**

The sales department has asked me to find out which member of their sales team currently has the highest total sales. This information will be used to hand out the bonus that is associated with the achievement. The sales department could also use this information to find areas of improvement for their team members.

1. Describe the data used for the report:
   1. The data used for this report will be:
      1. detailed\_report\_id - The unique ID number for each row of the detailed report
      2. payment\_id - The unique ID number for each payment
      3. customer\_id - The unique ID number for each customer
      4. staff\_id - The unique ID number for each staff member
      5. rental\_id - The unique ID number for each rental in the database
      6. amount - The dollar amount of each rental in the database
      7. payment\_date - The date for each payment transaction
      8. first\_name - The first name of the staff member
      9. last\_name - The last name of the staff member
      10. rental\_date - The date for the initial rental
      11. return\_date - The date the rentals were returned
      12. summary\_report\_id - The unique ID number for each row of the summary report
      13. total\_sales – The sum amount of sales per each salesperson (transformed from amount)
      14. month\_of\_the\_report - The month that the report is generated for
2. Identify two or more specific tables from the given dataset that will provide the data necessary for the detailed and summary sections of the report:
   1. The three tables necessary in this report are the staff table, the rental table, and the payment table.
3. Identify the specific fields that will be included in the detailed and summary sections of the report:
   1. The detailed table of the report will contain the fields detailed\_report\_id, payment\_id, customer\_id, staff\_id, rental\_id, amount, payment\_date, first\_name, last\_name, rental\_date, and return\_date.
   2. The summary table of the report will contain the fields summary\_report\_id, staff\_id, first\_name, last\_name, total\_sales, and month\_of\_report.
4. Explain the different business uses of the detailed and summary sections of the report.
   1. Detailed Report
      1. The detailed\_report table is designed to store fine-grained information about each transaction, including details about payments, customers, staff involved, rentals, and associated dates. Here's how it can be used in a business context:
      2. Transaction Analysis: This table allows for a deep dive into each transaction, making it possible to analyze customer behavior, payment patterns, and rental trends at an individual level.
      3. Customer Service and Dispute Resolution: With detailed records including customer IDs, staff IDs, rental IDs, and payment details, this table can be instrumental in resolving disputes, processing refunds, or addressing customer queries about specific transactions.
      4. Operational Insights: By examining rental and return dates, businesses can gauge inventory turnover, identify peak rental periods, and optimize inventory management.
      5. Staff Performance Monitoring: The inclusion of staff IDs allows for monitoring of staff performance in terms of customer interaction, efficiency in handling rentals, and processing payments.
   2. Summary Report
      1. The summary\_report table, on the other hand, aggregates data to present a high-level view of sales performance, specifically focusing on staff performance over a given month. Uses in a business setting include:
      2. Performance Tracking: By summarizing total sales by staff on a monthly basis, it provides a clear picture of individual performance, aiding in performance reviews, incentives, and identifying training needs.
      3. Trend Analysis: Comparing total sales across different months can help identify trends, seasonality, and the effectiveness of sales strategies or marketing campaigns.
      4. Resource Allocation: Insights into which staff members are generating more sales can guide decisions on resource allocation, staffing, and scheduling to optimize sales efforts.
      5. Strategic Planning: The summarized data can inform higher-level strategic decisions, such as expanding the team, setting future sales targets, or reallocating resources to different areas of the business.
5. Explain how frequently your report should be refreshed to remain relevant to stakeholders.
   * 1. This report should be run monthly to allow the sales department to come up with new strategies for the next month, and to let them see who their top salesperson is every month. The report can also be run at the discretion of the sales department.

**SECTION B**

The code for the detailed\_summary is as follows:

CREATE TABLE IF NOT EXISTS detailed\_report

(

detailed\_report\_id SERIAL PRIMARY KEY,

payment\_id INT NOT NULL,

customer\_id SMALLINT NOT NULL,

staff\_id SMALLINT NOT NULL,

rental\_id INT NOT NULL,

amount NUMERIC(5,2) NOT NULL,

payment\_date TIMESTAMP WITHOUT TIME ZONE NOT NULL,

first\_name VARCHAR(45),

last\_name VARCHAR(45),

rental\_date TIMESTAMP WITHOUT TIME ZONE NOT NULL,

return\_date TIMESTAMP WITHOUT TIME ZONE,

FOREIGN KEY (staff\_id) REFERENCES staff(staff\_id),

FOREIGN KEY (payment\_id) REFERENCES payment(payment\_id),

FOREIGN KEY (rental\_id) REFERENCES rental(rental\_id)

);

The summary\_report code is as follows:

CREATE TABLE IF NOT EXISTS summary\_report

(

staff\_id SMALLINT NOT NULL,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL,

total\_sales NUMERIC(10,2) NOT NULL,

report\_month DATE NOT NULL,

CONSTRAINT summary\_report\_pkey PRIMARY KEY (staff\_id, report\_month)

);

**SECTION C**

The code to extract the information from the database and load it into my detailed\_report is as follows:

INSERT INTO detailed\_report (payment\_id, customer\_id, staff\_id, rental\_id, amount, payment\_date, first\_name, last\_name, rental\_date, return\_date)

SELECT

p.payment\_id,

r.customer\_id,

s.staff\_id,

r.rental\_id,

p.amount,

p.payment\_date,

s.first\_name,

s.last\_name,

r.rental\_date,

r.return\_date

FROM

payment p

JOIN

rental r ON r.rental\_id = p.rental\_id

JOIN

staff s ON s.staff\_id = p.staff\_id;

The code to generate my summary report is as follows:

**SECTION D**

1. The code to create a trigger to continually update the summary\_report when the detailed report is updated is as follows:

CREATE OR REPLACE TRIGGER detailed\_report\_after\_change

AFTER INSERT OR UPDATE OR DELETE ON detailed\_report

FOR EACH ROW EXECUTE FUNCTION update\_summary\_report();

1. The code for the trigger function is as follows:

CREATE OR REPLACE FUNCTION update\_summary\_report()

RETURNS TRIGGER AS

$$

BEGIN

-- Determine the month of the affected row

-- Use NEW for insert/update operations and OLD for delete

-- COALESCE is used to handle both insert/delete cases

DECLARE

affected\_month DATE;

BEGIN

affected\_month := DATE\_TRUNC('month', COALESCE(NEW.return\_date, OLD.return\_date));

-- Delete the existing summary for the affected month

DELETE FROM summary\_report

WHERE staff\_id = COALESCE(NEW.staff\_id, OLD.staff\_id)

AND report\_month = affected\_month;

-- Recalculate and insert the new summary for the affected month

INSERT INTO summary\_report (staff\_id, first\_name, last\_name, total\_sales, report\_month)

SELECT

staff\_id,

first\_name,

last\_name,

SUM(amount) AS total\_sales,

affected\_month AS report\_month

FROM

detailed\_report

WHERE

DATE\_TRUNC('month', return\_date) = affected\_month

AND

staff\_id = COALESCE(NEW.staff\_id, OLD.staff\_id)

GROUP BY

staff\_id, first\_name, last\_name;

RETURN NULL; -- Since this is an AFTER trigger

END;

END;

$$

**SECTION F**

The following procedure will refresh data in both tables by dropping, recreating, and repopulating both tables. This procedure should be run monthly in order to find the top salesperson. We can run this automatically on the 1st of every month. The following is the code for the procedure:

CREATE OR REPLACE PROCEDURE refresh\_reports\_and\_find\_salespersons()

LANGUAGE plpgsql

AS $$

BEGIN

-- Drop the tables if they exist

DROP TABLE IF EXISTS detailed\_report, summary\_report;

-- Recreate the detailed\_report table

CREATE TABLE detailed\_report (

detailed\_report\_id SERIAL PRIMARY KEY,

payment\_id INTEGER NOT NULL,

customer\_id SMALLINT NOT NULL,

staff\_id SMALLINT NOT NULL,

rental\_id INTEGER NOT NULL,

amount NUMERIC(5,2) NOT NULL,

payment\_date TIMESTAMP WITHOUT TIME ZONE NOT NULL,

first\_name VARCHAR(45),

last\_name VARCHAR(45),

rental\_date TIMESTAMP WITHOUT TIME ZONE NOT NULL,

return\_date TIMESTAMP WITHOUT TIME ZONE

);

-- Recreate the summary\_report table

CREATE TABLE summary\_report (

staff\_id SMALLINT NOT NULL,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL,

total\_sales NUMERIC(10,2) NOT NULL,

report\_month DATE NOT NULL,

PRIMARY KEY (staff\_id, report\_month)

);

INSERT INTO detailed\_report (payment\_id, customer\_id, staff\_id, rental\_id, amount, payment\_date, first\_name, last\_name, rental\_date, return\_date)

SELECT

p.payment\_id,

r.customer\_id,

s.staff\_id,

r.rental\_id,

p.amount,

p.payment\_date,

s.first\_name,

s.last\_name,

r.rental\_date,

r.return\_date

FROM

payment p

JOIN

rental r ON r.rental\_id = p.rental\_id

JOIN

staff s ON s.staff\_id = p.staff\_id;

-- Call this function for the most recent complete month

PERFORM generate\_summary\_report((DATE\_TRUNC('month', CURRENT\_DATE) - INTERVAL '1 month')::DATE);

END;

$$;