**A: Summarize one real-world written business report:**

How many rentals were done by each employee and store?

Details Table: Create a table that allow the user to pull the number of rentals, sorted by date, employee ID, store, and/or film name or film ID. This can help track business performance.

Summary Table: Display the total rentals, full employee name, and the store ID. This can help track employee performance.

**1. Identify the specific fields that will be included in the detailed table and the summary table of the report:**

rental\_summary table:

staff\_id = smallint *ForeignKey*,

staff\_name = varchar(100) *PrimaryKey*,

month = varchar(10),

total\_rentals = integer

rental\_details table:

month = varchar(10) *PrimaryKey*,

staff\_id = smallint *PrimaryKey ForeignKey*,

first\_name = varchar(45),

last\_name = varchar(45),

count\_rentals = integer,

film\_id = integer *PrimaryKey ForeignKey*,

film\_title = varchar(100)

**2. Describe the types of data fields used for the report:**

varchar, integer, smallint

**3. Identify*at least* two specific tables from the given dataset that will provide the data necessary for the detailed table section and the summary table section of the report:**

The rental\_details table pulls from internal tables: rental, staff, inventory, and film.

The rental\_summary table pulls from the rental\_details table.

**4. Identify *at least* one field in the detailed table section that will require a custom transformation with a user-defined function and explain why it should be transformed:**

For the staff\_name field, first\_name and last\_name fields will be concatenated with a space for improved legibility.

**5. Explain the different business uses of the detailed table section and the summary table section of the report:**

a) rental\_summary table -> **Performance Reviews:** Provides concrete data for evaluating staff performance and setting goals.

b) rental\_details table -> **Rental Volume:** The count\_rentals column indicates how many times each film was rented by each staff member in a given month, helping with demand forecasting and marketing strategies.

**6.  Explain how frequently your report should be refreshed to remain relevant to stakeholders**

1. When tracking rentals for business intelligence, report should be re-run every month to create a baseline of monthly rentals for each fiscal year.

2. Alternatively, when tracking employee performance, the report could be re-run weekly to determine an Employee of the Month award recipient.

**Start of queries section:**

**B. Provide original code for function(s) in text format that perform the transformation(s) you identified in part A4:**

**-- User-defined function to concatenate first and last name**

CREATE OR REPLACE FUNCTION get\_staff\_name(first\_name VARCHAR, last\_name VARCHAR)

RETURNS VARCHAR(100) AS $$

BEGIN

    RETURN first\_name || ' ' || last\_name;

END;

$$ LANGUAGE plpgsql;

**C.  Provide original SQL code in a text format that creates the detailed and summary tables to hold your report table sections:**

**-- Create the rental\_summary table**

CREATE TABLE rental\_summary(

staff\_id SMALLINT,

staff\_name VARCHAR(100),

month VARCHAR(10),

total\_rentals INT,

PRIMARY KEY (staff\_id, month),

FOREIGN KEY(staff\_id)

REFERENCES staff(staff\_id)

);

**-- Create the rental\_details table**

CREATE TABLE rental\_details(

month VARCHAR(10),

staff\_id SMALLINT,

first\_name VARCHAR(45),

last\_name VARCHAR(45),

count\_rentals INT,

film\_id INT,

film\_title VARCHAR(100),

PRIMARY KEY (staff\_id, month, film\_id),

FOREIGN KEY (staff\_id)

REFERENCES staff (staff\_id),

FOREIGN KEY (film\_id)

REFERENCES film (film\_id)

);

**D.  Provide an original SQL query in a text format that will extract the raw data needed for the detailed section of your report from the source database:**

**-- Insert data into rental\_details table**

INSERT INTO rental\_details (month, staff\_id, first\_name, last\_name, count\_rentals, film\_id, film\_title)

SELECT

    TO\_CHAR(r.rental\_date, 'YYYY-MM') AS month,

    s.staff\_id,

    s.first\_name,

    s.last\_name,

    COUNT(r.rental\_id) AS count\_rentals,

    f.film\_id,

    f.title AS film\_title

FROM rental AS r

INNER JOIN staff AS s ON r.staff\_id = s.staff\_id

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

INNER JOIN film AS f ON i.film\_id = f.film\_id

GROUP BY month, s.staff\_id, s.first\_name, s.last\_name, f.film\_id, f.title;

**-- Insert data into rental\_summary table**

INSERT INTO rental\_summary (staff\_id, staff\_name, month, total\_rentals)

SELECT

    staff\_id,

    get\_staff\_name(first\_name, last\_name) AS staff\_name,

    month,

    SUM(count\_rentals) AS total\_rentals

FROM rental\_details

GROUP BY staff\_id, month, first\_name, last\_name;

**E.  Provide original SQL code in a text format that creates a trigger on the detailed table of the report that will continually update the summary table as data is added to the detailed table:**

CREATE OR REPLACE FUNCTION insert\_trigger\_function()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN DELETE FROM rental\_summary;

INSERT INTO rental\_summary (staff\_id, staff\_name, month, total\_rentals)

SELECT

    staff\_id,

    get\_staff\_name(first\_name, last\_name) AS staff\_name,

    month,

    SUM(count\_rentals) AS total\_rentals

FROM rental\_details

GROUP BY staff\_id, month, first\_name, last\_name;

RETURN NEW;

END;

$$

CREATE TRIGGER new\_summary

AFTER INSERT ON rental\_details

FOR EACH STATEMENT

EXECUTE PROCEDURE insert\_trigger\_function();

**F.  Provide an original stored procedure in a text format that can be used to refresh the data in *both*the detailed table and summary table. The procedure should clear the contents of the detailed table and summary table and perform the raw data extraction from part D:**

**--Following procedure refreshes data in the details table, which triggers the refresh of the summary table from the defined trigger function.**

CREATE OR REPLACE PROCEDURE refresh\_tables()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM rental\_details;

INSERT INTO rental\_details (month, staff\_id, first\_name, last\_name, count\_rentals, film\_id, film\_title)

SELECT

    TO\_CHAR(r.rental\_date, 'YYYY-MM') AS month,

    s.staff\_id,

    s.first\_name,

    s.last\_name,

    COUNT(r.rental\_id) AS count\_rentals,

    f.film\_id,

    f.title AS film\_title

FROM rental AS r

INNER JOIN staff AS s ON r.staff\_id = s.staff\_id

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

INNER JOIN film AS f ON i.film\_id = f.film\_id

GROUP BY month, s.staff\_id, s.first\_name, s.last\_name, f.film\_id, f.title;

RETURN;

END;

$$

--Call the function

CALL refresh\_tables();

1. **Identify a relevant job scheduling tool that can be used to automate the stored procedure.**

The pgAgent tool will work very well for this task. We can set this tool to run either weekly or monthly based on the type of reports we are looking to pull. Most commonly, a monthly report would suffice for business intelligence and can be set to run at midnight on the last day of the month, so reports are ready on each 1st of the month.

**G. Panopto video submission:**

**G. Panoptop video submission link:** [Panopto Video Link - Click Here](Panopto%20Video%20Link%20-%20Click%20Here)

**H.  Acknowledge all utilized sources, including any sources of third-party code, using in-text citations and references. If no sources are used, clearly declare that no sources were used to support your submission.**

No outside sources were used for this submission.