Nina Yavicoli
D326 -PA

A. Summarize one real-world written business report that can be created from the DVD Dataset from the "Labs on Demand Assessment Environment and DVD Database" attachment.

This report is intended to analyze customer rental behavior by examining how frequently individual films are rented and how long they are typically checked out. Through summarizing average rental duration and average rental frequency per month for each film, this report identifies titles that are both popular and consistently checked out for long periods of time. These patterns indicate films that may require additional inventory to improve availability and ultimately increase rental revenue by ensuring high-demand titles are consistently accessible to customers.

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

Detailed Table:
rental_id
rental_date
return_date
actual_rental_days
rental_month
film_id
film_title

Summary Table:
film_id
film_title
avg_rental_days
avg_rentals_per_month

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

Detailed Table:
rental_id - **INT**

rental_date - **TIMESTAMP**
return_date - **TIMESTAMP**
actual_rental_days – **SMALLINT**
rental_month - **TEXT**
film_id - **INT**
film_title – **VARCHAR(50)**

Summary Table:
film_id - **INT**
film_title – **VARCHAR(50)**
avg_rental_days – **NUMERIC(10,2)**
avg_rentals_per_month – **NUMERIC(10.2)**

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.

- **rental table:** provides checkout and return dates for each film.
- **film table:** provides film title and ID.
- **inventory table:** links specific films to rental activity.

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 (e.g., you might translate a field with a value of N to No and Y to Yes).

- The *actual_rental_days* field requires a transformation using a function called *calc_actual_rental_days*. This function calculates the difference between the rental date and the return date to determine how many days a film was checked out for.
- The *rental_month* field also requires a function that converts the rental date timestamp into a standardized year-month format so that rentals can be grouped by month.

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

- The detailed table supports operational and analytical needs by providing rental data at the transaction level. It allows an analyst to verify calculations, review individual rentals, and explore patterns or outliers when necessary.

- The summary table supports strategic decision-making by aggregating rental activity at the film level. It allows stakeholders to quickly identify films that are rented frequently and kept for longer periods, helping support inventory planning, demand forecasting, and availability management.

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

This report should be refreshed monthly to remain relevant to stakeholders as it uses monthly averages.

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

```
-- function that changes the rental duration to a SMALLINT reflecting days instead of a timestamp --

CREATE OR REPLACE FUNCTION calc_actual_rental_days (

p_rental_date TIMESTAMP,

 p_return_date TIMESTAMP

)

RETURNS SMALLINT

LANGUAGE plpgsql

AS $$

BEGIN

 IF p_return_date IS NULL THEN

 RETURN NULL;

END IF;

RETURN (p_return_date::date - p_rental_date::date)::SMALLINT;

END;
```

```
$$;


-- function that formats rental month as YYYY-MM --
CREATE OR REPLACE FUNCTION rental_month(p_rental_date TIMESTAMP)
 RETURNS TEXT
 LANGUAGE plpgsql
 AS $$
 BEGIN
RETURN TO_CHAR(p_rental_date, 'YYYY-MM');
END;
 $$;
```

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

```
-- detailed table creation --
 CREATE TABLE detailed_duration (
rental_id INT,
rental_date TIMESTAMP NOT NULL,
 return_date TIMESTAMP NULL,
 actual_rental_days SMALLINT,
 rental_month TEXT NOT NULL,
film_id INT NOT NULL,
 film_title VARCHAR(50) NOT NULL
```

```
);
```

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.

```
-- load raw data into tables --

INSERT INTO detailed_duration
SELECT
 r.rental_id,
 r.rental_date,
 r.return_date,
 calc_actual_rental_days(r.rental_date, r.return_date) AS actual_rental_days,
 rental_month(r.rental_date) AS rental_month,
 f.film_id,
 f.title AS film_title
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id;
```

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 trigger to update summary with detailed report --

CREATE OR REPLACE FUNCTION duration_trigger_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
  DELETE FROM summary_duration;

  INSERT INTO summary_duration (
    film_id,
    film_title,
    avg_rental_days,
```

```
    avg_rentals_per_month
  )
  SELECT
    film_id,
    film_title,
    AVG(actual_rental_days)::numeric(10,2) AS avg_rental_days,
    (COUNT(*)::numeric / NULLIF(COUNT(DISTINCT rental_month), 0))::numeric(10,2)
      AS avg_rentals_per_month
  FROM detailed_duration
  WHERE actual_rental_days IS NOT NULL
  GROUP BY film_id, film_title
  ORDER BY avg_rentals_per_month DESC;

  RETURN NEW;
END;
$$;

DROP TRIGGER IF EXISTS updated_summary_duration ON detailed_duration;

CREATE TRIGGER updated_summary_duration
AFTER INSERT
ON detailed_duration
FOR EACH STATEMENT
EXECUTE FUNCTION duration_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.

```
-- refresh tables procedure --

CREATE OR REPLACE PROCEDURE refresh_tables()
LANGUAGE plpgsql
AS $$
BEGIN
DELETE FROM detailed_duration;
DELETE FROM summary_duration;
```

```
INSERT INTO detailed_duration(
rental_id,
rental_date,
return_date,
actual_rental_days,
rental_month,
film_id,
film_title
)
SELECT
r.rental_id,
r.rental_date,
r.return_date,
calc_actual_rental_days(r.rental_date, r.return_date),
rental_month(r.rental_date),
f.film_id,
f.title
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id;

END;
$$;

CALL refresh_tables();
```

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

A relevant job scheduling tool that can be used to automate this stored procedure is pgAgent. pgAgent can be configured to run stored procedures on a defined schedule, allowing the tables to refresh automatically each month without manual intervention. This ensures the report remains up to date while aligning with the recommended refresh frequency.