D191- Advanced Data Management

Eduardo Ramirez

Student ID: 009769427

A. **Summarize one real-world written business report that can be created from the DVD Dataset and DVD Database.**

One real-world business case that can be extracted from the available data sets, is what film categories are rented most frequently and the total sales each category sold. With this data, a business can identify what film categories are in high demand. It will also help in making strategic pricing decisions when it comes to categories based on their popularity.

A1. Identify the specific fields that will be included in the detailed table and the summary table of the report.

Detailed table: category_id, film_id, rental_id, payment_id, film, film_category, rented_films, amount, rent_rate

Summary table: film_category, total_rents_by_category, total_sales_by_category, and rent_rate_by_category

A2. Describe the types of data fields used in the report.

Detailed table: SMALLINT, SMALLINT, SMALLINT, SMALLINT, VARCHAR, VARCHAR, INT, NUMERIC, NUMERIC

Summary table: VARCHAR, BIGINT, NUMERIC, NUMERIC

A3. 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 category and rental table will be two of the needed tables to create both the detail and summary report.

A4. 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.

The amount field on the detailed table will be called total_sales_by_category in the summary table and it will contain the total amount of sales by category. For example, the “Action” category will contain the sum of all the sales it has produced.

1. **A5. Explain the different business uses of the detailed table section and the summary table section of the report.**

The detailed table will provide a detailed breakdown of the categories associated with the films. With the information provided in the detailed section, the DVD rental business will be able to make inventory management decisions based on analytical data and facts. Allowing the business to stock more on-demand films featuring popular categories and set prices.

The summary table will provide an overview of what film categories have been rented most frequently, as well as the total sales per category, and the average price of a film associated with that category.

A6. Explain how frequently your report should be refreshed to remain relevant to stakeholders.

Since the business is continually changing, the report should be refreshed every month to maintain its relevancy to stakeholders.

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

--This function will convert the sum of all the payments received per category as well as the count of how many films have been rented and the average rent rate per category.

```
CREATE OR REPLACE FUNCTION summary_refresh_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN
DELETE FROM summary;
INSERT INTO summary(
                SELECT
                film_category AS film_category,
                COUNT(rental_id) AS total_rents_by_category,
                SUM(payment_id) AS total_sales_by_category,
                AVG(rent_rate) AS rent_rate_by_category
        FROM detailed
                GROUP BY film_category

);
RETURN NEW;
END;
$$
```

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

Detail table:

```
CREATE TABLE detailed (
   category_id SMALLINT,
   film_id SMALLINT,
   rental_id SMALLINT,
   payment_id SMALLINT,
   film VARCHAR(100),
   film_category VARCHAR(100),
   rented_films INT,
   amount NUMERIC,
   rent_rate NUMERIC
);
```

Summary table:

```
CREATE TABLE summary(
        film_category VARCHAR(100),
        total_rents_by_category BIGINT,
        total_sales_by_category NUMERIC,
        rent_rate_by_catergory NUMERIC
);
```

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.

--This will extract raw data from the source database into the detailed table report.

```
INSERT INTO detailed (category_id, film_id, rental_id, payment_id, film, film_category,
rented_films, amount, rent_rate)
```

```
SELECT
   category.category_id,
   film.film_id,
   rental.rental_id,
   payment.payment_id,
   film.title AS film,
   category.name AS film_category,
   COUNT(rental.rental_id) AS rented_films,
   payment.amount AS amount,
   film.rental_rate AS rent_rate
FROM category
JOIN film_category ON category.category_id = film_category.category_id
JOIN film ON film.film_id = film_category.film_id
JOIN inventory ON inventory.film_id = film.film_id
JOIN rental ON rental.inventory_id = inventory.inventory_id
JOIN customer ON customer.customer_id = rental.customer_id
JOIN payment ON rental.rental_id = payment.rental_id
GROUP BY category.category_id, film.film_id, rental.rental_id,
payment.payment_id,
category.name, payment.amount;
```

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.

--This trigger will execute after the is an insert on the detailed table.

```
CREATE TRIGGER summary_refresh
AFTER INSERT ON detailed
FOR EACH STATEMENT
EXECUTE PROCEDURE summary_refresh_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.

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

PgAgent would be a great job scheduling tool to use with PostgreSQL. The advantages of PgAgent is that it’s lightweight and Postgre native that can be used for automating and scheduling tasks within the database.

--Procedure to reload tables. To use this procedure use the CALL reload_tables().

```
CREATE PROCEDURE reload_tables()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM detailed;

INSERT INTO detailed(

        category_id, film_id, rental_id, payment_id, film, film_category, rented_films, amount, rent_rate)

SELECT

    category.category_id,

    film.film_id,

    rental.rental_id,

    payment.payment_id,

    film.title AS film,

    category.name AS film_category,

    COUNT(rental.rental_id) AS rented_films,

    payment.amount AS amount,

    film.rental_rate AS rent_rate

FROM category
```

```
JOIN film_category ON category.category_id = film_category.category_id
JOIN film ON film.film_id = film_category.film_id
JOIN inventory ON inventory.film_id = film.film_id
JOIN rental ON rental.inventory_id = inventory.inventory_id
JOIN customer ON customer.customer_id = rental.customer_id
JOIN payment ON rental.rental_id = payment.rental_id
GROUP BY category.category_id, film.film_id, rental.rental_id,
payment.payment_id,
category.name, payment.amount;
END$$
```