Rachel Pepe
August 2024

ADVANCED DATA MANAGEMENT D326
PERFORMANCE ASSESMENT

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.

The following business report is created from the DVD Rental Database and will summarize sales volume categorized by movie genre to help stakeholders answer the question of which genres of movies are the most profitable. The detailed table will provide extensive information on films, genres, rental identification and payment. The summary table will summarize the total sales volume per genre of movie. This report will address which film genres bring in the most revenue which will help the business to focus on where it can further expand by purchasing more rentals in high grossing genres and lowering spending on genres that don't bring in much income. This will help the business grow financially and attract more customers with the most in demand videos.

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

The specific fields I will be including in the detailed table and summary table of the report include; title, category_id, rental_id, payment_date_month, payment_date_year, rental_rate, amount, name, and total_sales_revenue. The following fields are not used as direct columns in the detailed and summary tables but are used as foreign and primary keys to join tables; film_id, and inventory_id.

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

Types of data fields used for the detailed and summary table of the report include:

- Title - VARCHAR(225) - A variable length string that represents the name of the title of each film from the film table of the DVD database.
- Category_id - INT (Primary Key) - An integer number used to categorize and label the 16 unique category/genre types from the category table of the database.
- Name - VARCHAR(25) - A variable length string that represents the genre name for each of the 16 categories in the DVD database from the category table.

- Rental_id – INT (Primary Key) – An integer number that represents the id number for each rental transaction in the rental table of the DVD database.
- Rental_rate - NUMERIC (4,2) - A decimal number that represents the listed price that each film is rented for. Rental_rate is from the film table.
- Payment_date_month - VARCHAR(10) - A variable length string that represents the month that a movie was rented for a specific transaction referenced from the payment table. This is a user defined function.
- Payment_date_year – INT – An integer number that represents the year that a movie was rented for a specific transaction referenced from the payment table. This is a user defined function.
- Amount – NUMERIC (5,2) - A decimal number that signifies the amount that a film was rented out for. This can differ from the actual rental rate. Amount is a field in the payment table of the DVD rental database.
- Total_sales_revenue – NUMERIC (10,2) - A decimal number that is calculated by summing the total number of payments for all rentals of films in each category of movie. This uses the amount column from the payment table as a parameter in an aggregate function.

Types of data fields used as primary and foreign keys to join tables, or as a parameter for a user defined function that is used in the detailed table of the report include:

- Inventory_id – INT (Primary Key) - An integer number used to inventory and track films in the DVD Rental database. The inventory_id field is found in the inventory table.
- Film_id – INT (Primary Key) - An integer number used to label and identify each film title in the film table of the DVD database.
- Payment_date TIMESTAMP – Integer numbers that consist of the date and time a payment was made. Payment_date is found in the payment table.

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 following specific tables will be used directly in the detailed and summary tables of my report.

- Film Table –
    - The title column from the Film table will be used directly in the detailed table of my report
- Film_category Table -
    - The category_id column from the Film_category table will be used directly in the detailed and summary table of my report.

- Category Table
    - The name column of the category table will be used in the detailed and summary tables of my report
- Rental Table -
    - The rental_id column from the rental table will be used directly in the detailed table of my report
- Payment Table -
    - The payment_date column will be used in the detailed table of my report as a parameter of the user defined functions payment_date_month and payment_date_year.
    - The amount column will be used in the detailed table and as a parameter for an aggregate function in the summary 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.

The payment_date field from the payment table that is used in the detailed table of my report requires custom transformations with user-defined functions. The user defined function named payment_date_month will extract the month from the payment date field and transform it to the specified name of the month in which the transaction was made as a variable length string. The user defined function payment_date_year will transform the TIMESTAMP to an integer number that reflects just the year that the transaction was made.

The payment_date field should be transformed so it is easier to read for stakeholders who are utilizing this business report. The transformations will take the long format of the TIMESTAMP data type and transform it into two easy to read columns specifying the month and year a transaction was made. This will allow them to view the data from the table for relevance of when the transaction was made and how up to date the information is.

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

Stakeholders can use the detailed table of the report to verify how recent certain movies have been rented out and which genre category they belong to. They can also use it to compare the rental rate of movies compared to how much revenue they actually brought in for a certain movie title. They can see if certain genres of top titles bring in a higher amount than the base rental rate of those titles. This information can be used to make purchasing decisions for more

film titles. The table also contains rental ids so that they can easily track inventory and make sure they have enough stock of on demand titles.

Stakeholders can use the summary table of the report to see how much money each movie genre has brought in. This can help them make important financial decisions to increase business and profitability. They can use the information to purchase more movies from high grossing categories and limit purchasing of genres that don't bring in much profit. This will also attract more customers as they purchase movies from popular genres.

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

The report should be refreshed monthly to remain relevant to stakeholders. By refreshing the report every month with new data, the stakeholders will be able to stay on top of trending genres and make purchase decisions accordingly.

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

```
-- Function to transform payment date to month string
    CREATE OR REPLACE FUNCTION payment_date_month (payment_date
        TIMESTAMP)
    RETURNS VARCHAR(10)
    LANGUAGE plpgsql
    AS
    $$
    DECLARE simple_month VARCHAR (10);
    BEGIN
    SELECT to_char(payment_date, 'Month') INTO simple_month;
    RETURN simple_month;
    END;
    $$;

-- Function to transform payment date to year
    CREATE OR REPLACE FUNCTION payment_date_year (payment_date
    TIMESTAMP)
    RETURNS INT
    LANGUAGE plpgsql
    AS
    $$
    DECLARE simple_year INT;
```

```
BEGIN
SELECT EXTRACT(YEAR FROM payment_date) INTO simple_year;
RETURN simple_year;
END;
$$;
```

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

```
-- Create table statement for detailed rental table
    CREATE TABLE detailed_rental (
        title VARCHAR(225),
        name VARCHAR(25),
        category_id INT,
        rental_id INT,
        payment_date_month VARCHAR(10),
        payment_date_year INT,
        rental_rate NUMERIC (5,2),
        amount NUMERIC(5,2)
    );

-- Create table statement for summary rental table
    CREATE TABLE summary_rental (
        name VARCHAR(25),
        category_id INT,
        total_sales_revenue NUMERIC(10,2)
    );
```

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 raw data into detailed rental table
    INSERT INTO detailed_rental
    SELECT f.title, c.name, fc.category_id, r.rental_id,
    payment_date_month(p.payment_date), payment_date_year(p.payment_date),
    f.rental_rate, p.amount
    FROM payment p
    JOIN rental r ON p.rental_id = r.rental_id
    JOIN inventory i ON i.inventory_id = r.inventory_id
```

```
JOIN film f ON f.film_id = i.film_id
JOIN film_category fc ON fc.film_id = f.film_id
JOIN category c ON c.category_id = fc.category_id
ORDER BY f.title, c.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.

```
-- Trigger statement to insert into summary table
    CREATE TRIGGER summary_insert_trigger
    AFTER INSERT
    ON detailed_rental
    FOR EACH STATEMENT
    EXECUTE PROCEDURE trigger_function();

    CREATE OR REPLACE FUNCTION trigger_function()
    RETURNS TRIGGER
    LANGUAGE plpgsql
    AS $$
    BEGIN
    DELETE FROM summary_rental;
    INSERT INTO summary_rental
    SELECT name, category_id, SUM(amount)
    FROM detailed_rental
    GROUP BY name, category_id
    ORDER BY SUM(amount) DESC;
    RETURN NEW;
    END;
    $$;
```

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.

```
-- Stored procedure to refresh data in detailed and summary tables
    CREATE PROCEDURE table_refresh_procedure()
```

```
LANGUAGE plpgsql
AS $$
BEGIN
DELETE FROM detailed_rental;
DELETE FROM summary_rental;
INSERT INTO detailed_rental
SELECT f.title, c.name, fc.category_id, r.rental_id,
        payment_date_month(p.payment_date), payment_date_year(p.payment_date),
f.rental_rate, p.amount
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON i.inventory_id = r.inventory_id
JOIN film f ON f.film_id = i.film_id
JOIN film_category fc ON fc.film_id = f.film_id
JOIN category c ON c.category_id = fc.category_id
ORDER BY f.title, c.name;
RETURN;
END;
$$;
```

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

A relevant job scheduling too that can be used to automate the stored procedure would be pgAgent. This is a scheduling agent designed to run on PostgreSQL and it runs and manages jobs like stored procedures.

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 sources or third-party code were used for my report.