Student Name: Jessica Short

Student ID: 010500487

Course: D326 – Advanced Data Management

Assignment: CSN1 — CSN1 Task 1: Data Analysis

October 17, 2024

<u>PART A</u>

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.

I chose the following business question for this report: Which top three film categories had the most rentals each month?

The answer to this question holds a considerable amount of value for a DVD rental business owner. The results can be used to choose which promotional posters to order for the store windows. For example, if the "Action" category is the top performing category, then promotional posters featuring new action films would likely attract customers to enter the store and browse. The results report would also be helpful in choosing which new release film titles and associated quantities to order for the stores' rental inventory. A DVD rental business could also use the results of the report to decide how to layout/organize their stores. By placing categories at the front of a store, customers are more likely to see the category, browse that section, and rent DVDs from that category. A high-demand category at the front of the store would likely result in an increase in overall revenue.

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

The fields that will be included the in the detailed table are:

- **rental_id** – from the table "rental" – A unique identifier for the rental transaction.
- **rental_month** (derived from rental_date from the table "rental")
- **rental_year** (derived from rental_date from the table "rental")
- **store_id** – from the table "store" – This is the store where the rental transaction occurred.

- **title** – from the table “film” – This is the title of the film.
- **category_name** – This is the “name” field from the table “category” and the detailed table – This is the name of the film category.

The fields that will be included in the summary table are:

- **rental_month** (derived from rental_date from the table “rental”)
- **rental_year** (derived from rental_date from the table “rental”)
- **category_name** - This is the “name” field from the table “category” and the detailed table – This is the name of the film category.
- **total_monthly_rentals** - This is the sum of all rental transactions for a given month and a given film category

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

The fields that will be included the in the detailed table are:

- **rental_id** – INT - from the table “rental” – A unique identifier for the rental transaction
- **rental_month** (derived from rental_date from the table “rental”) – INT
- **rental_year** (derived from rental_date from the table “rental”) - INT
- **store_id** – INT - from the table “store” – This is the store where the rental transaction occurred.
- **title** – VARCHAR(255) - from the table “film” – This is the title of the film.
- **category_name** – VARCHAR(25) – This is the “name” field from the table “category” and the detailed table – This is the name of the film category.

The fields that will be included in the summary table are:

- **rental_month** (derived from rental_date from the table “rental”) – INT
- **rental_year** (derived from rental_date from the table “rental”) - INT
- **category_name** - VARCHAR(25) – This is the “name” field from the table “category” and the detailed table – This is the name of the film category.
- **total_monthly_rentals** – INT – This is the sum of all rental transactions for a given month and a given film category

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 tables from the given dataset will provide the data necessary for this report: rental, store, inventory, film, film_category, and category.

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).

Two user defined functions will be utilized. One will extract the month from rental_date and return an integer value. The other will extract the year from rental_date and return an integer value. This transformation is needed to determine the time period for rental transaction aggregate values in the summary table. In addition, this transformation will make both tables easy to read and review.

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

By examining the summary table, stakeholders can easily determine the top performing film categories for each month. This knowledge can influence owners to implement informed decisions to grow their business. These decisions include ordering new promotional materials, selecting which new release films to order each month, and opting to decommission some films.

For example, if rentals of Foreign DVDs have been extremely low during the past 3 months, you may consider decommissioning the least popular titles in the category and reselling them as “pre-loved” DVDs to own. If after another 3 months, Foreign DVD sales are still low, you may consider getting rid of the section altogether.

The summary table also illustrates how rental trends change throughout the year. For example, one would expect that rentals in the categories Family, Classics, and Children would increase during the holiday months (November-December).

The detailed table contains individual rental transactions, described by rental ID, store ID, film title, and the category name. The detailed table can provide more insight into customer rental trends such as the store location that rented the most DVDs, the most popular DVD titles, and the top DVD categories/titles by store. It can also be used for auditing purposes as well.

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

This report should be refreshed once every month. In doing so, the business owners can make any necessary adjustments to adapt to changing customer demand. If you refreshed less frequently, the resulting data would likely be obsolete. If you refreshed more frequently, you may not have enough data points to accurately reflect rental trends. Also, refreshing more frequently than once per month could hinder productivity. This is due to the increase in the amount of time spent reviewing reports.

PART B

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

```
--PART B:  Return rental month as an integer --
CREATE OR REPLACE FUNCTION rental_month(rental_date TIMESTAMP)
RETURNS INT
LANGUAGE plpgsql
AS
$$
DECLARE rental_month INT;
BEGIN
        SELECT EXTRACT(MONTH FROM rental_date) INTO rental_month;
        RETURN rental_month;
END;
$$;
```

```
--PART B:  Return rental year as an integer --
CREATE OR REPLACE FUNCTION rental_year(rental_date TIMESTAMP)
RETURNS INT
LANGUAGE plpgsql
AS
$$
DECLARE rental_year INT;
BEGIN
        SELECT EXTRACT(YEAR FROM rental_date) INTO rental_year;
        RETURN rental_year;
END;
$$;
```

PART C

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

```
--PART C: Create Detailed Table--
CREATE TABLE detailed_rental_table (
rental_id INT,
rental_month INT,
rental_year INT,
store_id INT,
title VARCHAR(255),
category_name VARCHAR(25)
);
```

```
--PART C: Create Summary Table--
CREATE TABLE summary_rental_table (
rental_month INT,
rental_year INT,
category_name VARCHAR(25),
total_monthly_rentals INT
);
```

PART 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.

```
--PART D: Extract the raw data needed for the detailed table --
INSERT INTO detailed_rental_table(
rental_id,
rental_month,
rental_year,
store_id,
title,
category_name
)
SELECT
r.rental_id,
rental_month(rental_date) AS rental_month,
rental_year(rental_date) AS rental_year,
s.store_id,
f.title,
c.name AS category_name
FROM rental AS r
```

```
LEFT JOIN inventory AS i ON  r.inventory_id = i.inventory_id
LEFT JOIN store AS s ON i.store_id = s.store_id
LEFT JOIN film AS f ON i.film_id = f.film_id
LEFT JOIN film_category AS fc ON f.film_id = fc.film_id
LEFT JOIN category AS c ON fc.category_id = c.category_id;
```

<u>PART E</u>

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 update_summary_table()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
DELETE FROM summary_rental_table;
INSERT INTO summary_rental_table(
rental_month,
rental_year,
category_name,
total_monthly_rentals)
SELECT
rental_month,
rental_year,
category_name,
COUNT (rental_id) AS total_monthly_rentals
FROM detailed_rental_table
```

```
GROUP BY rental_year, rental_month, category_name
ORDER BY rental_year, rental_month, category_name;
RETURN NEW;
END;
$$;

CREATE TRIGGER update_summary_table_trigger
AFTER INSERT OR UPDATE OR DELETE on detailed_rental_table
FOR EACH STATEMENT
EXECUTE PROCEDURE update_summary_table();
```

PART 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.

```
CREATE OR REPLACE PROCEDURE refresh_rental_tables()
LANGUAGE plpgsql
AS $$
BEGIN
--Disable trigger--
ALTER TABLE detailed_rental_table DISABLE TRIGGER
update_summary_table_trigger;
--Clear contents of detailed and summary tables--
TRUNCATE TABLE detailed_rental_table;
TRUNCATE TABLE summary_rental_table;
--Extract the raw data needed for the detailed table--
INSERT INTO detailed_rental_table(
rental_id,
rental_month,
rental_year,
store_id,
title,
category_name
)
SELECT
r.rental_id,
rental_month(rental_date) AS rental_month,
rental_year(rental_date) AS rental_year,
s.store_id,
f.title,
c.name AS category_name
FROM rental AS r
LEFT JOIN inventory AS i ON  r.inventory_id = i.inventory_id
LEFT JOIN store AS s ON i.store_id = s.store_id
LEFT JOIN film AS f ON i.film_id = f.film_id
LEFT JOIN film_category AS fc ON f.film_id = fc.film_id
LEFT JOIN category AS c ON fc.category_id = c.category_id;
--Extract the data needed for the summary table--
INSERT INTO summary_rental_table(
rental_month,
rental_year,
category_name,
total_monthly_rentals)
SELECT
rental_month,
rental_year,
category_name,
COUNT (rental_id) AS total_monthly_rentals
FROM detailed_rental_table
GROUP BY rental_year, rental_month, category_name
ORDER BY rental_year, rental_month, category_name;
--Re-enable trigger--
ALTER TABLE detailed_rental_table ENABLE TRIGGER
update_summary_table_trigger;
RETURN;
END;
$$;


--Execute this statement to invoke the stored procedure--
CALL refresh_rental_tables();
```

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

I would use pgAgent to automate this stored procedure. I chose pgAgent because it is managed by pgAdmin4, the management application used in this project.

Assuming you have already installed pgAdmin4, follow the steps outlined below to automate this stored procedure with pgAgent (Source: Dias, 2020):

1. Download pgAgent.
2. Open pgAdmin4 and create the extension pgagent.
3. Start the pgAgent service.
4. Create a job in pgAdmin 4 and give it a name.
5. Define the job steps to for your stored procedure. The steps for this report are outlined in Part F above.

6. Select the tab “Schedules” to define the scheduling for your job. For this report, the job will be executed once per month.
7. Test your job to see that everything works correctly.

<u>PART G</u>

Provide a Panopto video recording that includes the presenter and a vocalized demonstration of the functionality of the code used for the analysis.

Panopto video link:
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=5ff80efd-1f8e-4cd3-9430-b2110175ee45

Note: For instructions on how to access and use Panopto, use the "Panopto How-To Videos" web link provided below. To access Panopto's website, navigate to the web link titled "Panopto Access," and then choose to log in using the “WGU” option. If prompted, log in using your WGU student portal credentials, and then it will forward you to Panopto’s website.

To submit your recording, upload it to the Panopto drop box titled “Advanced Data Management D191 | D326 (Student Creators) [assignments].” Once the recording has been uploaded and processed in Panopto's system, retrieve the URL of the recording from Panopto and copy and paste it into the Links option. Upload the remaining task requirements using the Attachments option.

<u>PART H</u>

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.

“An Overview of Job Scheduling Tools for PostgreSQL”

By Hugo Dias

Published February 3, 2020

https://severalnines.com/blog/overview-job-scheduling-tools-postgresql/