Justonna Naing
Advanced Data Management - D191
001467971

Business Report from the DVD Rental Database

A. Summarize one real-world business report that can be created from the attached Data Sets and Associated Dictionaries.

As the DVD rental business is growing, business owners may want to keep track of which categories of genres are generating the highest revenue. Knowing this information will help the owner where to stock more inventory based on the total amount of quantity rented and total sales generated by each genre. This will help the business owner reduce the cost of inventory used for unpopular genre categories.

1. Describe the data used for the report.

Following data will be used for the reports.

- **category** - contain the store film's categories data
- **film_category** - contain data relationships between films and categories
- **film** - contain film data such as title, release year, etc.
- **inventory** - contain inventory data
- **rental** - contain rental information
- **payment** - contain customer's payments for the rented film

Category information is needed to collect a list of different genres. Category will then be joined with film_category and film information to know which rented film has what categories. Rental table will look at the inventory table to see requested movie film has available quantity in stock. Finally, rental information will need to join with the payment table to get the cost of the amount for renting the movie.

2. Identify two or more specific tables from the given dataset that will provide the data necessary for the detailed and the summary sections of the report.

Tables that will be using for the detailed and the summary sections of the reports:

- **category** - to acquire category_id and category name (genre) and join to film_category table using category_id
- **film_category** - to join to film table using film_id
- **film** - to acquire film_id and join to inventory table using film_id
- **inventory** - to acquire inventory_id and join to rental table using inventory_id

- **rental** - to acquire rental_id and join to payment table using rental_id
- **payment** - to acquire payment_id and amount

3. Identify the specific fields that will be included in the detailed and the summary sections of the report.

Fields that will be included in the detailed section of the report:

Field	Type	Description / Definition
name	varchar (20)	The film category name (genre)
category_id	integer	A unique identifier for each film category
film_id	integer	A unique identifier for each film
inventory_id	integer	A unique identifier for each inventory
rental_id	integer	A unique identifier for each rental information
payment_id	integer	A unique identifier for customer payment information
amount	numeric (7,2)	The cost of renting the film

Fields that will be included in the summary section of the report:

Field	Type	Description / Definition
genre	varchar (20)	The film category name (genre)
total_rents	integer	Total rent count by film category (genre)
total_sales	numeric (7,2)	Total sales for each film category (genre)

4. Identify one field in the detailed section that will require a custom transformation and explain why it should be transformed. For example, you might translate a field with a value of 'N' to 'No' and 'Y' to 'Yes'.

Rental_id and amount fields will require a custom transformation. Rental_id will be counted as total_rents and amount will be summed as toatl_sales. It will help the business owner to easily identify total sales and total rents by each genre.

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

Detail section of the report:

- The detail section of the report will hold all the information necessary to the rented film based on specific genre and their total sales. This report will help business owners to assess inventory quantities for films in each genre to ensure appropriate copies in inventory to match expected demand.

Summary section of the report:

- The summary section will provide a quick look at Genre, Total Rents, and Total Sales. This report will help visualize the business owner which film genres are generating most revenue for the company.

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

The Report should be refreshed monthly to remain relevant to business owners and to align with monthly marketing communications and monthly decisions for purchases of new films and purging of films in unpopular categories to reduce demand.

B. Write a SQL code that creates the tables to hold your report sections.

```
-- B. CREATE detailed table-
      DROP TABLE IF EXISTS detailed;
      CREATE TABLE detailed (
         name varchar(20),
         category_id integer,
         film_id integer,
         inventory_id integer,
         rental_id integer,
         payment_id integer,
         amount numeric (7,2)
         );
-- To view empty detailed table
         -- SELECT * FROM detailed;
 -- CREATE summary table
         DROP TABLE IF EXISTS summary;
         CREATE TABLE summary (
               genre varchar(20),
               total_rents integer,
               total_sales numeric (7,2)
         );
   -- To view empty summary table
   -- SELECT * FROM summary;
```

C. Write a SQL query that will extract the raw data needed for the Detailed section of your report from the source database and verify the data's accuracy.

```
-- C. Extract raw data needed for the detailed section of the
report from the DVD database

INSERT INTO detailed (
   name,
   category_id,
   film_id,
   inventory_id,
   rental_id,
   payment_id,
   amount)
SELECT
   a.name, a.category_id, c.film_id, d.inventory_id,
   e.rental_id,p.payment_id, p.amount
FROM category AS a
INNER JOIN film_category b ON a.category_id = b.category_id
INNER JOIN film c ON b.film_id = c.film_id
INNER JOIN inventory d ON c.film_id = d.film_id
INNER JOIN rental e ON d.inventory_id = e.inventory_id
LEFT JOIN payment p ON e.rental_id = p.rental_id;

-- To view contents of detailed table
-- SELECT * FROM detailed;
```

D. Write code for function(s) that perform the transformation(s) you identified in part A4.

```
-- D. CREATE FUNCTION refreshing the summary table with a data
transformation
-- Transforming amount from the detailed table with an aggregation
(summing amount, counting rental_id and grouping name)

CREATE OR REPLACE FUNCTION summary_refresh_function()
RETURNS TRIGGER AS $$
BEGIN
DELETE FROM summary;
INSERT INTO summary (
      SELECT
            name, count(rental_id), sum(amount)
      FROM detailed
      GROUP BY 1
      ORDER BY 2 DESC, 3);
RETURN NEW;
END; $$ LANGUAGE PLPGSQL;
```

E. Write a SQL code 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.

```
-- E. CREATE TRIGGER

--DROP TRIGGER summary_refresh ON detailed

CREATE TRIGGER summary_refresh
AFTER INSERT ON detailed
FOR EACH STATEMENT
EXECUTE PROCEDURE summary_refresh_function();
```

F. Create a stored procedure that can be used to refresh the data in *both* your detailed and summary tables. The procedure should clear the contents of the detailed and summary tables and perform the ETL load process from part C and include comments that identify how often the stored procedure should be executed.

```
-- F. CREATE STORED PROCEDURE
-- To be automated to run on a monthly basis, the last day of
every month
-- Use the external pgAgent application as a job scheduling tool
      CREATE OR REPLACE PROCEDURE refresh_reports()
      LANGUAGE PLPGSQL
      AS $$
      BEGIN
      DELETE FROM detailed;
      INSERT INTO detailed (
         name,
         category_id,
         film_id,
         inventory_id,
         rental_id,
         payment_id,
         amount)
      SELECT
         a.name, a.category_id, c.film_id, d.inventory_id,
         e.rental_id,p.payment_id, p.amount
      FROM category AS a
            INNER JOIN film_category b ON a.category_id =
            b.category_id
            INNER JOIN film c ON b.film_id = c.film_id
            INNER JOIN inventory d ON c.film_id = d.film_id
            INNER JOIN rental e ON d.inventory_id = e.inventory_id
            LEFT JOIN payment p ON e.rental_id = p.rental_id;
      END; $$;
```

```
-- To call stored procedure
        -- CALL refresh_reports();

-- To view results
      -- SELECT * FROM detailed;
      -- SELECT * FROM summary;
```

1. Explain how the stored procedure can be run on a schedule to ensure data freshness.

To ensure data freshness, the stored procedure for this data should be run at least once a month.

Since PostgreSQL does not have a built-in automation function, we will use pgAgent as our job scheduling application.

First, we will install pgAgent. Then, we will create pgagent extension, which will create all the tables and functions for the pgAgent operation.

A new job can be created using "pgAgent Jobs", and defining the steps to execute it.

Web Sources:

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