POSTGRESQL LAB:

ETL, FUNCTIONS, & TRIGGERS

OBJECTIVES

Using a DVD Rental database in Postgres, address the following to determine “Which genre of movie generated the least revenue per quarter?”:

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

B. Write SQL code to…

   I. **create the tables** to hold your report sections.

   II. **extract the raw data** needed for the Detailed section of the report from the source database and verify the data’s accuracy.

   III. create **functions that perform the data transformations**.

   IV. **create a trigger** on the detailed table of the report that will continually update the summary table as data is added to the detailed table.

   V. **create a stored procedure** that can be used to refresh the data in both the detailed and summary tables.

DATA POINTS AND DETAILS

Figure 1: Database tables and details

Table Name	Original Field Name	Detailed Field Name	Summary Field Name	Report(s) of Inclusion
Rental	rental_id	rental_id	–	Detailed
	rental_date*	rental_date (timestamp)	quarter,year (double precision)	Detailed, Summary
	inventory_id**	–	–	Detailed
Payment	amount	amount	amount	Detailed, Summary
	rental_id**	–	–	Detailed
Inventory	inventory_id**	–	–	Detailed
	film_id**	–	–	Detailed
Film_Category	film_id**	–	–	Detailed
	category_id**	–	–	Detailed
Category	name	genre	genre	Detailed, Summary
	category_id**	–	–	Detailed

* Field used to transform the date into the corresponding quarter and year to associate genre sales with rental dates. This transformation will identify the quarter in which rentals took place, and thus associate it with the genre sales.
** Field only used in a join statement.

Data
The data used in this lab is a subset of data pulled from a large DVD rental company.

Data Points
To determine the movie genre that generated the least revenue each quarter for the DVD Rental Company, three key data points are used:

1. movie genre,
2. sales amount, and
3. rental date

Data Point Details
Data was collected from the tables displayed in Figure 1, which contain the specific fields and their associated reports.

SQL CODE & QUERIES

I. SQL TABLE FOR CREATING & HOUSING DETAILED & SUMMARY SECTIONS

```
--CREATE: Detailed table
DROP TABLE IF EXISTS revenue_detailed;

CREATE TABLE revenue_detailed(
   rental_id INTEGER PRIMARY KEY,
   genre VARCHAR(25),
   rental_date TIMESTAMP,
   amount NUMERIC(12,2)
)
```

```
--CREATE: Summary table
DROP TABLE IF EXISTS revenue_summary;

CREATE TABLE revenue_summary(
   genre VARCHAR(25),
   quarter DOUBLE PRECISION,
   year DOUBLE PRECISION,
   amount NUMERIC(12,2)
)
```

II. SQL EXTRACTION QUERY FOR DETAILED SECTION

```
--EXTRACT: Insert raw data into Detailed table

INSERT INTO revenue_detailed(
   rental_id,
   genre,
   rental_date,
   amount
)

SELECT
   r.rental_id,
   cat.name,
   r.rental_date,
   p.amount

FROM
   rental r
JOIN payment p ON r.rental_id = p.rental_id
JOIN inventory I ON r.inventory_id = i.inventory_id
JOIN film_category fact ON i.film_id = fcat.film_id
JOIN category cat ON fcat.category_id = cat.category_id;
```

```
--VERIFY: Data accuracy

SELECT COUNT(*) FROM revenue_detailed;
SELECT COUNT(*)

FROM
   rental r
JOIN payment p ON r.rental_id = p.rental_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film_category fcat ON i.film_id = fcat.film_id
JOIN category cat ON fcat.category_id = cat.category_id;
```

III. SQL FUNCTION TO TRANSFORM DATA

```
/*CREATE: Function to transform rental date (Detailed) to
  Quarter & Year (Summary)*/

CREATE OR REPLACE FUNCTION revenue_summary_xform()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN

   CREATE TEMP TABLE ttb (
      genre VARCHAR(25),
      quarter DOUBLE PRECISION,
      year DOUBLE PRECISION,
      amount NUMERIC(12,2)
      );

--Transform data from Detailed section and aggregate amount

   INSERT into ttb(
      SELECT
         genre,
      EXTRACT(quarter FROM rental_date) as quarter,
      EXTRACT(year FROM rental_date)as year,
      SUM(amount) as amount

      FROM
         revenue_detailed

      GROUP BY
          genre,
      EXTRACT(quarter FROM rental_date)
      EXTRACT(year FROM rental_date)
      );

/*Insert transformed data and gather each genre per quarter
with the least revenue*/

   INSERT INTO revenue_summary(
      SELECT ttb.genre, ttb.quarter, ttb.year, r2.low_sales
from ttb
      INNER JOIN (
         SELECT quarter, min(amount) as low_sales
         FROM ttb
         GROUP BY quarter) r2 ON ttb.quarter = r2.quarter
and ttb.amount=r2.low_sales);
   DROP TABLE ttb;

   RETURN NEW;

END;
$$
```

IV. SQL TRIGGER ON DETAILED TABLE TO UPDATE SUMMARY TABLE

```
/*CREATE: Trigger to Update Summary Table when Detailed
Table is modified*/

CREATE TRIGGER revenue_summary_update
AFTER INSERT
ON revenue_detailed
FOR EACH STATEMENT
EXECUTE PROCEDURE revenue_summary_xform();
```

V. SQL STORED PROCEDURE TO UPDATE DETAILED & SUMMARY SECTIONS

```
/*
The procedure would be run quarterly because of the nature of the business question, "Which genre of movie generated the least revenue per
quarter?" is based on the quarter of the year. A data refresh can be achieved by creating and configuring a Postgres pgAgent Job in the pgAdmin
extension pgAgent.
*/

CREATE PROCEDURE revenue_combo_update()
   LANGUAGE PLPGSQL
   AS
$$
BEGIN

   --CLEAR: Detailed  & Summary tables
   TRUNCATE revenue_detailed;
   TRUNCATE revenue_summary;

   --INSERT: Raw data into Detailed section
   INSERT INTO revenue_detailed(
      rental_id,
      genre,
      rental_date,
      amount
    )

   SELECT
      r.rental_id,
      cat.name,
      r.rental_date,
      p.amount
   FROM
      rental r
   JOIN payment p ON r.rental_id = p.rental_id
   JOIN inventory i ON r.inventory_id = i.inventory_id
   JOIN film_category fcat ON i.film_id = fcat.film_id
   JOIN category cat ON fcat.category_id =
      cat.category_id;


END;
$$

--VALIDATE: Stored procedure
SELECT * FROM revenue_detailed;
SELECT * FROM revenue_summary;
CALL revenue_combo_update();
```

USE CASE SUMMARY

The report and SQL coding could be executed quarterly to evaluate current performance, make necessary adjustments for the upcoming quarter, and review previous quarters. The resulting summary section can help increase marketing efforts for struggling genres, restructure the inventory by decreasing low-performing genres and increasing in-demand genres, or identify patterns over longer periods for further analysis of possible underlying causes of low sales.

Alternatively, the detailed section can be used to determine whether specific rental prices need to be adjusted.

THANKS FOR VIEWING

S.J. Richardson

Github: SQLJamz