Trevor England

D326 Performance Assessment

11/25/2024

1. *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.*

In this project, I will be giving a Seasonal Trend Analysis report. One real world business question that would be useful to answer is “During the summer months of 2005, what were the top three genres of movies that customers rented the most?” Please note that I have defined the term “summer months” as being June 1st – August 31st.

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

The following fields will be included in the **summary** table:

**genre** VARCHAR(25) (Note that this is category.name renamed as genre)

**total\_rentals** INTEGER

The following fields will be included in the **detailed** table:

**rental\_id** SERIAL PRIMARY KEY,

**rental\_date** TIMESTAMP

**customer\_id** INTEGER,

**customer\_name** VARCHAR(60),

**movie\_title** VARCHAR(255),

**genre** VARCHAR(25),

**film\_id** INTEGER,

**category\_id** INTEGER

1. *Describe the types of data fields used for the report.*

The data types used for this report include VARCHAR for all string data, INTEGER for all number data, and TIMESTAMP for the date of the rentals.

1. *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 **summary** table will be populated by joining the rental\_details table.

The **detailed** table will be populated by the joining rental, inventory, film, film\_category, category, and customer tables.

1. *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).*

The fields that I have created a user-defined function for are the “first\_name” and last\_name” fields. I chose to do a transformation on these fields for better readability of the data, as well as condensing the detailed table down by one column.   
  
The other user-defined function I created was on the name column in the category table. This function parses through the rental summary table to find the top three most rented genres as an array. This allows the output (the array of genres) to be used in other queries upon the function call. This approach allows for reusability of this function for all seasons, since the calculation of the top three genres is built into the function and renders the list dynamically.

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

The **summary** table provides a superficial insight as to which movie genres are rented the most during summer months of a particular year. This information can be used to analyze the data against previous years to determine if there is a trend of genres being popular during the summer months. If there does seem to be a trend, it will help the store come up with a business plan for the number of movies in the most popular genres they should have available to support customer demand for the upcoming summer season to boost their number of rentals.

The **detailed** table provides a granular look into the data from the summary table. Here, the business can see individual transactions from customers of the top three genres that were most frequently rented and try to determine correlations between the specific movies (such as movie duration, actors, etc.) and the number of rentals. This can be helpful as well to a business because they can use the data for personalized marketing to customers. For example, if you use this same table structure and modify the months for each season, you can then see which customers are most active during each season. This helps the business to target customers and potentially use loyalty rewards program more efficiently by offering deals at calculated times to keep customers coming in.

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

Since this specific report is a seasonal trend analysis report, the report should be refreshed at the end of each season. This will identify how the store performed in a given season for a given year, which can drive business decisions. If the stakeholders wanted to see the progress of the business decision based on the season trend analysis report from the year prior, you could alter the query dates and refresh the report on a weekly basis. This will show stakeholders whether the DVD Rental business is headed in the right direction with their marketing campaigns and DVD inventory given the current weekly data.

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

-- User Defined Function for dynamically populating detailed table with top three genres from the summary table

CREATE OR REPLACE FUNCTION get\_top\_three\_genres()

RETURNS VARCHAR[] AS $$

DECLARE top\_three\_genres\_list VARCHAR[];

BEGIN

SELECT array\_agg(genre)

INTO top\_three\_genres\_list

FROM (

SELECT cat.name AS genre, COUNT(r.rental\_id) AS rental\_count

FROM rental AS r

INNER JOIN inventory AS i ON r.inventory\_id = i.inventory\_id

INNER JOIN film AS f ON i.film\_id = f.film\_id

INNER JOIN film\_category AS fc ON f.film\_id = fc.film\_id

INNER JOIN category AS cat ON fc.category\_id = cat.category\_id

WHERE r.rental\_date BETWEEN '2005-06-01' AND '2005-09-01'

GROUP BY cat.name

ORDER BY rental\_count DESC

LIMIT 3

) AS top\_genres;

RETURN top\_three\_genres\_list;

END;

$$ LANGUAGE plpgsql;

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

-- Create summary table

CREATE TABLE rental\_summary (

genre VARCHAR(25),

total\_rentals INTEGER

);  
  
-- Create detailed table

CREATE TABLE rental\_details(

rental\_id SERIAL PRIMARY KEY,

rental\_date TIMESTAMP,

customer\_id INTEGER,

customer\_name VARCHAR(60),

movie\_title VARCHAR(255),

genre VARCHAR(25),

film\_id INTEGER,

category\_id INTEGER,

FOREIGN KEY(customer\_id) REFERENCES customer(customer\_id),

FOREIGN KEY(film\_id) REFERENCES film(film\_id),

FOREIGN KEY(category\_id) REFERENCES category(category\_id)

);

1. *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 INTO rental\_details

SELECT r.rental\_id, r.rental\_date, cust.customer\_id, get\_customer\_name(cust.first\_name, cust.last\_name) AS customer\_name, f.title AS movie\_title, cat.name AS genre, f.film\_id, cat.category\_id

FROM rental AS r

INNER JOIN inventory AS i ON r.inventory\_id = i.inventory\_id

INNER JOIN film AS f ON i.film\_id = f.film\_id

INNER JOIN film\_category AS fc ON f.film\_id = fc.film\_id

INNER JOIN category AS cat ON fc.category\_id = cat.category\_id

INNER JOIN customer as cust ON r.customer\_id = cust.customer\_id

WHERE r.rental\_date BETWEEN '2005-06-01' AND '2005-09-01'

AND cat.name IN (SELECT unnest(get\_top\_three\_genres()))

ORDER BY r.rental\_date DESC;

1. *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.*

-- Update the summary table when the detail table is updated with new data

CREATE OR REPLACE FUNCTION update\_summary()

RETURNS TRIGGER

AS $$

BEGIN

DELETE FROM rental\_summary;

INSERT INTO rental\_summary

SELECT genre, COUNT(rental\_id) AS rental\_count

FROM rental\_details

GROUP BY genre

ORDER BY rental\_count DESC;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create the trigger

CREATE TRIGGER auto\_summary

AFTER INSERT

ON rental\_details

FOR EACH STATEMENT

EXECUTE PROCEDURE update\_summary();

1. *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 procedure to refresh table data

CREATE OR REPLACE PROCEDURE refresh\_summary\_and\_details()

AS $$

BEGIN

DELETE FROM rental\_details;

INSERT INTO rental\_details

SELECT r.rental\_id, r.rental\_date, cust.customer\_id, get\_customer\_name(cust.first\_name, cust.last\_name) AS customer\_name, f.title AS movie\_title, cat.name AS genre, f.film\_id, cat.category\_id

FROM rental AS r

INNER JOIN inventory AS i ON r.inventory\_id = i.inventory\_id

INNER JOIN film AS f ON i.film\_id = f.film\_id

INNER JOIN film\_category AS fc ON f.film\_id = fc.film\_id

INNER JOIN category AS cat ON fc.category\_id = cat.category\_id

INNER JOIN customer as cust ON r.customer\_id = cust.customer\_id

WHERE r.rental\_date BETWEEN '2005-06-01' AND '2005-09-01'

AND cat.name IN (SELECT unnest(get\_top\_three\_genres()))

ORDER BY r.rental\_date DESC;

RETURN;

END;

$$ LANGUAGE plpgsql;

-- Call the procedure

CALL refresh\_summary\_and\_details();

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

There are a few tools for scheduling the stored procedure, however I would choose pg\_cron. Once you install the package, you can enable pg\_cron in your database by creating an extension. From there, you should add pg\_cron to your shared preload libraries. Once the procedure has been written, you can use pg\_cron to schedule a job by calling cron.schedule and passing in the job name, the schedule for which you want the job to run (similar to the way you would set up a job using crontab in Linux), and the command to call the procedure. From here, the task of calling the stored procedure would be scheduled on a user-defined interval to refresh the report as needed.

1. *Provide a Panopto video recording that includes the presenter and a vocalized demonstration of the functionality of the code used for the analysis.*
2. *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.*

<https://github.com/citusdata/pg_cron>

* README.md section

<https://www.postgresql.org/docs/current/sql-createtrigger.html>

<https://stackoverflow.com/questions/2987433/how-to-import-csv-file-data-into-a-postgresql-table>

<https://www.geeksforgeeks.org/postgresql-array_agg-function/>,

-\copy zip\_codes(ZIP,CITY,STATE) FROM '/path/to/csv/ZIP\_CODES.txt' DELIMITER ',' CSV

1. *Demonstrate professional communication in the content and presentation of your submission.*