Jessica Borsellino

Performance Assessment Advanced Data Management

D326

7/10/2023

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.**

The rental business has two locations that sell a similar number of rentals each month, one real-world business report that can be extracted from their data set is a comparison to see what store is selling the highest number of rentals per month. This will provide insight to the managers and owners that can drive a number of revenue-increasing activities such as the potential for friendly competition between stores and employees, or focused advertisements based on location. This report consists of two tables, the detailed report contains a comprehensive listing of rentals, film names, genres, rental dates, and store id. Additionally, a summary report is included and shows total rentals by store by month.

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

The detailed table will include:

* Rental\_id
* Store\_id
* Film\_title
* Film\_genre
* Rental\_date

The summary table will include:

* Rental\_month
* Total\_rentals
* Store\_id

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

The detailed table will include:

* rental\_id INT – This column holds individual rental IDs from the rental table so they can be counted by date and store for the summary report.
* store\_id SMALLINT - This column is a single digit, either 1 or 2, no need for larger numerical data types.
* film\_title VARCHAR(255) – The title of the film provided in the event subsequent reports will be run on this table. The longest film title is 255 characters.
* film\_genre VARCHAR(50) – The genre of film provided in the event subsequent reports will be run on this table. The longest genre is 50 characters.
* rental\_date DATE – Illustrates when an individual rental transaction took place, this data will be transformed to get rentals by month for the summary table and business report.

The summary table will include:

* rental\_month VARCHAR – This column is the product of a transformation on rental\_date which extracts the month from a date as a string.
* rental\_year INT – This column is the product of a transformation on rental\_date which extracts the year from the date as an integer.
* total\_rentals BIGINT – This aggregate column is the count of all individual rental\_ids for transactions in the rental table.
* store\_id SMALLINT – This column is a single digit, either 1 or 2, no need for larger numerical data types.

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.**

For the detailed table, data will be provided from the rental, inventory, film, film category, and category tables. For the summary report, data will be provided from the rental date and inventory 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 rental\_date field in the detailed table will require a custom transformation through the use of a user-defined function so that the month and year can be extracted from the date data type. The business report focused on rentals by month between the two stores, the most relevant information is provided by extracting the name of the month as a string from this field, the year is also relevant to determine exactly which time period the data are coming from.

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

The most applicable data that the detailed report table provides is the individual rental transaction id, the store id, and the rental date. These will be used to create the summary report, allowing business owners to compare total sales per month by store. There are multiple improvements that the rental business can make based on this such as adjusting inventory, running advertisements, and fostering friendly competition among the staff at the locations as management can decide to reward staff for renting the most that month. This summary report provides insight into which location is performing better and certainly many conclusions can be drawn from that; is there more foot traffic in one location, are the customer demographics a factor, are the staff performing better, and so on and so forth. The detailed report also includes the film title and genre so that additional reports can be run, perhaps a specific genre is more popular at one store or is rented more often in certain months. That is out of the scope of this project but providing those columns in the business report makes it easy to gain these insights.

The detailed table provides additional columns to derive further analysis from and is a comprehensive report from the full rental database, the summary table is a broad overview that can serve as a starting point for the business to begin to optimize their operations.

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

This report should be refreshed every month as its most pertinent use case is the comparison of sales by store per month. Once the data from the prior month is received it is a reasonable expectation that the report be generated as soon as possible so that staff and management can provide feedback or adjust procedures for the month ahead. Additionally, this monthly refresh allows for inventory adjustment – is store 1 consistently out of an action movie that store 2 never rents out? A monthly refresh is a pertinent factor for these reports to remain relevant.

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

(For these sections, I will supply both plain text code along with screenshots as the format is easier to read on a screenshot but the project requirements specify plain text. I have performed some additional work above and beyond project and rubric requirements, these sections are noted as “ADDITIONAL WORK” and are underlined to easily separate them from the core components).

For the transformation, I created two functions that automate similar extraction tasks. The first function returns the month as a string, extracted from a date data type while the second returns the year as an integer.

A computer screen shot of a program

Description automatically generated

--RETURN MONTH AS STRING--

CREATE OR REPLACE FUNCTION get\_month\_string(rental\_date DATE)

RETURNS TEXT

LANGUAGE plpgsql

AS $$

DECLARE month\_string TEXT;

BEGIN

SELECT TO\_CHAR (rental\_date, 'Month')

INTO month\_string;

RETURN month\_string;

END; $$;

--RETURN YEAR AS INT--

CREATE OR REPLACE FUNCTION get\_year(rental\_date DATE)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE sales\_year INT;

BEGIN

SELECT EXTRACT (YEAR from rental\_date)

INTO sales\_year;

RETURN sales\_year;

END; $$;

ADDITIONAL WORK: I created a table with the output or return values from these functions to verify data that is input into the summary table. The extra table is named rentals\_by\_month, it includes a comprehensive view of total rentals as illustrated by the included query. It will be the sum of both rental stores’ sales for any given month. The screenshot and plain text code are included below.

A screenshot of a computer program

Description automatically generated

--CREATE RENTALS BY MONTH TABLE--

--This table serves as data verification for summary report table--

CREATE TABLE rentals\_by\_month(

rental\_month VARCHAR(25),

rental\_year INT,

store\_id SMALLINT,

rental\_id INT

);

INSERT INTO rentals\_by\_month (

rental\_month,

rental\_year,

store\_id,

rental\_id

)

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

store\_id,

rental\_id

FROM detailed\_report;

--DATA VERIFICATION QUERY TOTALS SHOULD BE SUM OF BOTH STORES IN SUMMARY REPORT--

SELECT rental\_month,

COUNT(rental\_id) AS total\_rentals

FROM rentals\_by\_month

GROUP BY rental\_month

ORDER BY total\_rentals;

--RESULTS FROM VERIFICATION QUERY—

A screenshot of a computer

Description automatically generated

--SELECTING ALL FROM SUMMARY\_REPORT—

SELECT \* FROM summary\_report;

A screenshot of a computer

Description automatically generated

For example, adding the amounts from store 1 and store 2 in August produces 5,686 which is shown in the total\_rentals column in the rentals\_by\_month table.

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

A screenshot of a computer screen

Description automatically generated A screen shot of a computer

Description automatically generated

--CREATE DETAILED REPORT TABLE--

CREATE TABLE detailed\_report (

rental\_id INT,

store\_id SMALLINT,

film\_title VARCHAR(255),

film\_genre VARCHAR(50),

rental\_date DATE

);

--CREATE SUMMARY REPORT TABLE--

CREATE TABLE summary\_report (

rental\_month VARCHAR(25),

rental\_year INT,

total\_rentals BIGINT,

store\_id SMALLINT

);

**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.**

A screen shot of a computer

Description automatically generated

--POPULATE DETAILED REPORT TABLE--

INSERT INTO detailed\_report (

rental\_id,

store\_id,

film\_title,

film\_genre,

rental\_date

)

SELECT

r.rental\_id,

i.store\_id,

f.title as film\_title,

cat.name as film\_genre,

r.rental\_date

FROM rental AS r

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

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

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id;

ADDITIONAL WORK: I have also taken the time to include the data population of the summary report here, with screenshot and raw code included.

A screen shot of a computer program

Description automatically generated

INSERT INTO summary\_report(

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

**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.**

A screenshot of a computer program

Description automatically generated

--TRIGGER CREATION BETWEEN 2 TABLES--

--this updates the summary table when data is added to the detailed table--

CREATE OR REPLACE FUNCTION summary\_updater\_function()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM summary\_report;

INSERT INTO summary\_report

(SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN NEW;

END; $$;

CREATE TRIGGER summary\_updater\_trigger

AFTER INSERT OR UPDATE OR DELETE ON detailed\_report

FOR EACH STATEMENT

EXECUTE PROCEDURE summary\_updater\_function();

ADDITIONAL WORK: I have also taken the time to create the following steps on verifying that the trigger does work as intended, screenshot and plain text code provided.

A screenshot of a computer program

Description automatically generated

--VERIFYING TRIGGER ON DETAILED\_REPORT--

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044—

A screenshot of a computer

Description automatically generated

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10—

A screenshot of a computer

Description automatically generated

INSERT INTO detailed\_report VALUES (20000, 2, 'FAKE FILM', 'FAKE GENRE', '01-01-1901')

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16045—

A screenshot of a computer

Description automatically generated

SELECT COUNT(\*) FROM summary\_report

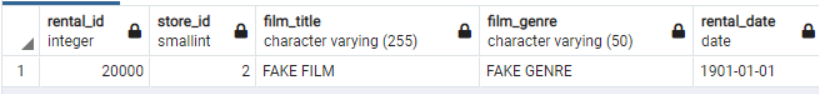
--RESULT IS 11—

A screenshot of a computer

Description automatically generated

DELETE FROM detailed\_report WHERE rental\_id = 20000 RETURNING \*;

--this will show the exact row that is deleted--



SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044—

A screenshot of a computer

Description automatically generated

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10—

A screenshot of a computer

Description automatically generated

--back to normal, trigger works--

**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.**

A screenshot of a computer program

Description automatically generated

--STORED PROCEDURE TO REFRESH REPORTS--

CREATE OR REPLACE PROCEDURE report\_refresher()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM detailed\_report;

DELETE FROM summary\_report;

INSERT INTO detailed\_report (

SELECT

r.rental\_id,

i.store\_id,

f.title as film\_title,

cat.name as film\_genre,

r.rental\_date

FROM rental AS r

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

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

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id);

INSERT INTO summary\_report (

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN;

END; $$;

--TO CALL PROCEDURE--

CALL report\_refresher();

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

The lab application itself, pgAgent is a job scheduling tool that can automate stored procedures for Postgres databases.

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

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7ed14a21-677f-4239-9872-b03c0060485c>

**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 were used.