Harrison Plummer
D191 Performance Assessment
10/2/2023

A. *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.*

A common report I've created at various jobs is a month over month and year over year report. Since they only have about 4 full months of rental data, I decided to do a month over month rental report for July and August 2005 separated by staff. The summary report will be total rentals (by count) by month and staff. The detailed report will be rentals per film by month and staff.

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:

   staff_id (int primary/foreign key), month (varchar(9) primary key), total_rentals (int)

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

   film_id (int primary/foreign key), month (varchar(9) primary/foreign key), staff_id (int primary/foreign key), film_title (varchar(255)), count_rentals (int)

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

   Data types included will be VARCHAR, and INTEGER

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 summary table will pull from the detail table.
   The detail table will pull from the inventory, rental, staff, and film tables.

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*).*

   The field that will be transformed is the timestamp payment_date field to varchar(9) month field. This will be done for better readability by stakeholders.

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

   The summary table can be used as a brief look to make see the number of rentals a staff member is processing. This could be used to make sure all staff members are meeting their SLA if there is one. It could also be used as one indicator of high or low performance (though other factors should be taken into consideration. A shelf stocker won't have that many rentals, if any, compared to a cashier).

   The detailed table can be used to determine the top rented DVDs and could be compared to inventory data to ensure that there are enough copies to rent out.

Inventory data could also be compared to this report (or maybe a year-long version of this report) to determine DVDs that don't get rented out and can be removed from inventory, or at least reduce the number of copies in inventory.

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

   These reports should be refreshed monthly (most likely on the first of the month for the previous 2 months). Preferably similar reports containing year-over-year data would be available as well.

Code starts here *(assignment section descriptions are included in comments):*

```
--These were some common queries I ran while testing
--DELETE FROM rental_details;
--SELECT * FROM rental_details;

--DELETE FROM rental_summary;
--SELECT * FROM rental_summary;

--DROP TABLE rental_summary;
--DROP TABLE rental_details;

/* This query was used to verify the total count from the created tables matched the original
    count
SELECT COUNT(*)
FROM rental
WHERE rental_date BETWEEN '07/01/2005 00:00:00' AND '08/31/2005 23:59:59';
*/

/*B.  Provide original code for function(s) in text format that perform the transformation(s) you
    identified in part A4.
At first I overcomplicated this in my head and tried to create a case statement for each month. It
    wasn't working well and then I remembered that there are conversion functions for time.*/
CREATE OR REPLACE FUNCTION month_string(payment_date TIMESTAMP)
RETURNS VARCHAR(9)
LANGUAGE plpgsql
AS
$$
DECLARE month_return VARCHAR(9);
BEGIN
SELECT to_char(payment_date, 'Month') INTO month_return;
RETURN month_return;
END;
$$
```

```
/*C.  Provide original SQL code in a text format that creates the detailed and summary tables to
     hold your report table sections.
*/
CREATE TABLE rental_summary (
staff_id INT,
staff_name VARCHAR(45),
month VARCHAR(9),
total_rentals INT,
PRIMARY KEY(staff_id, month),
FOREIGN KEY(staff_id)
REFERENCES staff (staff_id)
);

CREATE TABLE rental_details (
film_id INT,
month VARCHAR(9),
staff_id INT,
staff_name VARCHAR(45),
film_title VARCHAR(255),
count_rentals INT,
PRIMARY KEY (film_id, month, staff_id),
FOREIGN KEY (film_id)
REFERENCES film (film_id),
FOREIGN KEY (staff_id)
REFERENCES staff (staff_id)
);


/*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.
*/
INSERT INTO rental_details
SELECT      i.film_id, month_string(r.rental_date), r.staff_id, s.last_name, f.title,
     COUNT(r.inventory_id)
FROM rental AS r
LEFT 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 staff AS s
ON r.staff_id = s.staff_id
WHERE r.rental_date BETWEEN '07/01/2005 00:00:00' AND '08/31/2005 23:59:59'
GROUP BY month_string(r.rental_date), r.staff_id, s.last_name, i.film_id, f.title
ORDER BY month_string(r.rental_date), r.staff_id, COUNT(r.inventory_id) DESC;
```

```
/*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.
*/
CREATE OR REPLACE FUNCTION insert_trigger_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
DELETE FROM rental_summary;
INSERT INTO rental_summary
SELECT staff_id, staff_name, month, SUM(count_rentals)
FROM rental_details
GROUP BY month, staff_id, staff_name
ORDER BY month, staff_id;
RETURN NEW;
END;
$$


CREATE TRIGGER new_summary
AFTER INSERT
ON rental_details
FOR EACH STATEMENT
EXECUTE PROCEDURE insert_trigger_function();

/*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_tables()
LANGUAGE plpgsql
AS $$
BEGIN
DELETE FROM rental_details;
INSERT INTO rental_details
SELECT      i.film_id, month_string(r.rental_date), r.staff_id, s.last_name, f.title,
     COUNT(r.inventory_id)
FROM rental AS r
LEFT 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 staff AS s
ON r.staff_id = s.staff_id
WHERE r.rental_date BETWEEN '07/01/2005 00:00:00' AND '08/31/2005 23:59:59'
GROUP BY month_string(r.rental_date), r.staff_id, s.last_name, i.film_id, f.title
```

```
ORDER BY month_string(r.rental_date), r.staff_id, COUNT(r.inventory_id) DESC;
--The rental_summary table will be cleared and and refreshed automatically with the trigger.
RETURN;
END;
$$

CALL refresh_tables();

SELECT * FROM rental_summary;
SELECT * FROM rental_details;
```

End of code.

F1. Identify a relevant job scheduling tool that can be used to automate the stored procedure.

Since this was done in PostgresSQL, I would install and use the pgAgent tool. I would most likely have it refresh on the first of the month around 4:00 AM so the updates are complete once employees start their day. This timing makes sense for these reports since they are a month over month report, and a full month has to be complete for accurate comparisons.

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=8ee7fad4-4d97-485b-ab06-b08f012fa150#

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 for third-party code.