D191 PA Documentation

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=14651e5f-73f3-4468-a8e9-adf200064e9a>

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

In this report I will be creating a summary and detailed table that will contain data that shows the number of rentals and sales made. The summary table will show the total number of rentals and total sales made by each store, ordered by the total sales. The detailed table will show each staff member at each store and every rental and sale that they have made. The summary table can be used to see which stores are meeting rental and sale quotas, and the detailed table can be used to identify which staff members are contributing the most or the least to each store’s totals. The tables also only hold data for rentals that have been returned by customers as the payment is processed upon return.

**1. Describe the data used for the report.**

In the summarized section of this report, I am using the store ID to display total rentals and sales per each store. In the detailed section, I am using the store ID and staff ID to display the rentals and sales made by each active staff member of each store.

**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.**

I’m pulling data from the store, staff, rental, and payment tables.

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

I’m using the store table’s store\_id, staff table’s staff\_id, first\_name, and last\_name fields, the rental table’s rental\_id field, and the payment table’s amount field.

**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’.**

I’m transforming the first\_name and last\_name fields by concatenating them into a single field to reduce the number of columns in the table and also reduce the number of fields that will be in queries on the detailed table.

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

The summary section can be used to determine each store’s total rentals and sales. This is beneficial for determining the overall productivity of each store. The detailed section can be used to determine which staff members are meeting or exceeding sales quotas and allow for management to acknowledge successful staff members or identify staff members that are not making enough sales.

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

The report should be updated at least once per month to display each stores monthly earnings and which staff members are meeting or exceeding sale quotas.

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

    CREATE TABLE summary\_table

    (

        store\_id int NOT NULL,

        total\_rentals int NOT NULL,

        total\_sales decimal(12,2) NOT NULL

    );

    CREATE TABLE detailed\_table

    (

        store\_id int NOT NULL,

        staff\_id int NOT NULL,

        staff\_name varchar(90) NOT NULL,

        rental\_id int NOT NULL,

        rental\_price decimal(10,2) NOT NULL

    );

Text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, website

Description automatically generated

**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.**

INSERT INTO detailed\_table

        SELECT s.store\_id, e.staff\_id, concat\_name(e.first\_name, e.last\_name), r.rental\_id, p.amount

            FROM store AS s

                JOIN staff AS e ON s.store\_id=e.store\_id

                    JOIN rental AS r ON e.staff\_id=r.staff\_id

                        JOIN payment AS p ON r.rental\_id=p.rental\_id

            WHERE e.active = true

            ORDER BY e.staff\_id, rental\_id desc;

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

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

CREATE FUNCTION concat\_name(first\_name varchar(45), last\_name varchar(45))

        RETURNS varchar(90)

        LANGUAGE plpgsql

        AS

        $$

            DECLARE staff\_name varchar(90);

                BEGIN

                SELECT first\_name || ' ' || last\_name INTO staff\_name;

                RETURN staff\_name;

            END;

        $$

Text

Description automatically generated

\* proof of this function working can be seen in the above section’s screenshot of the detailed table

**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.**

    CREATE PROCEDURE update\_sum\_table()

        LANGUAGE plpgsql

        AS

        $$

            BEGIN

                TRUNCATE summary\_table;

                INSERT INTO summary\_table

                    SELECT store\_id, count(rental\_id), sum(rental\_price)

                        FROM detailed\_table

                        GROUP BY store\_id

                        ORDER BY store\_id;

            END;

        $$;

    CREATE FUNCTION insert\_trigger\_function()

        RETURNS TRIGGER

        LANGUAGE plpgsql

        AS

        $$

            BEGIN

                CALL update\_sum\_table();

                RETURN NEW;

            END;

        $$;

    CREATE TRIGGER detailed\_inserted

        AFTER INSERT ON detailed\_table

        FOR EACH ROW

        EXECUTE PROCEDURE insert\_trigger\_function();

Text

Description automatically generated

Summary Table before data inserted into detailed table

Graphical user interface, text, application, website

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Summary table after data inserted into detailed table

Graphical user interface, text, application, chat or text message

Description automatically generated

**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.**

CREATE PROCEDURE reload\_tables\_data() -- Should be executed monthly

        LANGUAGE plpgsql

        AS

        $$

            BEGIN

                TRUNCATE detailed\_table;

                INSERT INTO detailed\_table

                    SELECT s.store\_id, e.staff\_id, concat\_name(e.first\_name, e.last\_name), r.rental\_id, p.amount

                        FROM store AS s

                            JOIN staff AS e ON s.store\_id=e.store\_id

                                JOIN rental AS r ON e.staff\_id=r.staff\_id

                                    JOIN payment AS p ON r.rental\_id=p.rental\_id

                        WHERE e.active = true

                        ORDER BY e.staff\_id, r.rental\_id desc;

                TRUNCATE summary\_table;

                INSERT INTO summary\_table

                    SELECT store\_id, count(rental\_id), sum(rental\_price)

                        FROM detailed\_table

                        GROUP BY store\_id;

            END;

        $$

Text

Description automatically generated

A picture containing background pattern

Description automatically generated

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

Since PostgreSQL doesn’t have a built-in task scheduler that can be used to run this stored procedure on a schedule, the use of an external tool is necessary. As of PostgreSQL 9.5, an extension known as pg\_cron can be loaded as a shared library into PostgreSQL and can be used to schedule the stored procedure. The documentation and files for pg\_cron can be found here <https://github.com/citusdata/pg_cron>