Brandon Council

SID: 009588480

D191 Advanced Data Management

Performance Assessment

Business Report

The problem selected for this business report is the Monthly DVD Rental Analysis (MDRA). With the given data sets and associated dictionaries, MDRA will provide valuable insights into the frequency of DVD rentals in a month. This information can be used for marketing, inventory management, and provide an understanding of customer trends.

A1. For the detailed section of the report the following specific fields will be used:

* Rental ID
* Rental date
* Return date
* Inventory ID
* Customer ID
* Staff ID

For the summary section of the report the following specific fields will be used:

* Month\_year
* Total\_rentals

A2. The data required for this report will present customer interactions and rental transactions.

The MDRA will contain information on each rental event, customer demographics, and employee involvement. MDRA will allow the stakeholders to assess customer activity and staff performance.

A3. The primary tables used from the dataset are rental, inventory, and customer. Rental is used to capture individual rental transactions. Inventory is used to track rentals to specific DVD titles. Customer is used for identification purposes if necessary for stakeholders.

A4. One field identified in the detailed section that will require transformation is the rental\_date and return\_date. These fields currently have timestamps and it will be changed into a more readable monthly format for stakeholders.

A5. The detailed section provides a complete view of transactions, showing rental activities by line view. It can also support operational decision making and customer service management by tracking late returns and the data can be used in dispute resolution.

The summary section presents a high-level overview of rental volume trends, which can aid in strategic decision-making. It can help in forecasting demand cycles, and guiding marketing initiatives.

A6. The refresh frequency should be high ensuring this report captures the most up-to-date trends effectively. Given the fast-moving environment of DVD rentals requires that a weekly refresh is recommended. Weekly refresh is preferred to capture the latest trends without overburdening the system for its function to stakeholders

F1. To schedule a stored procedure in PostgreSQL, we can use the pg\_cron extension. After setting the extension in place, and given the dynamic nature of the DVD business, we can automate nightly updates. This will allow stakeholders to adapt faster to emerging trends in daily operations.

-- B. CREATE TABLES

DROP TABLE IF EXISTS detailed\_rentals;

CREATE TABLE detailed\_rentals(

rental\_id integer PRIMARY KEY,

inventory\_id integer,

rental\_date timestamp,

customer\_id integer,

return\_date timestamp,

staff\_id integer

);

-- This displays the currently empty table

SELECT \* FROM detailed\_rentals;

-- CREATION OF THE EMPTY MONTHLY SUMMARY TABLE

DROP TABLE IF EXISTS monthly\_summary\_rentals;

CREATE TABLE monthly\_summary\_rentals(

month\_year VARCHAR(7),

total\_rentals integer

);

-- C. DATA INSERTION FOR DETAILED\_RENTALS TABLE

INSERT INTO detailed\_rentals(

rental\_id,

inventory\_id,

rental\_date,

customer\_id,

return\_date,

staff\_id

)

SELECT

rental\_id,

inventory\_id,

rental\_date,

customer\_id,

return\_date,

staff\_id

FROM rental;

-- This will show the now inserted data from the rental

-- table inside of the detailed\_rentals table

SELECT \* FROM detailed\_rentals;

-- D. FUNCTION CREATION

-- Summary table update

-- 1: Creates a function that will refresh monthly\_summary\_rentals table data

CREATE OR REPLACE FUNCTION refresh\_monthly\_summary()

RETURNS TRIGGER AS $$

BEGIN

-- 2: Refreshes the monthly\_summary\_rentals table with the data that is in detailed\_rentals table

DELETE FROM monthly\_summary\_rentals;

INSERT INTO monthly\_summary\_rentals(month\_year, total\_rentals)

SELECT

TO\_CHAR(DATE\_TRUNC('month', rental\_date), 'YYYY-MM') AS month\_year,

COUNT(\*) AS total\_rentals

FROM detailed\_rentals

GROUP BY TO\_CHAR(DATE\_TRUNC('month', rental\_date), 'YYYY-MM');

RETURN NEW;

END; $$

LANGUAGE plpgsql;

-- E. TRIGGER CREATION

CREATE TRIGGER trigger\_refresh\_monthly\_summary

AFTER INSERT ON detailed\_rentals

FOR EACH STATEMENT

EXECUTE FUNCTION refresh\_monthly\_summary();

INSERT INTO detailed\_rentals(rental\_id, inventory\_id, rental\_date, customer\_id, return\_date, staff\_id)

VALUES (20000, 20000, '2023-12-01 10:00:00', 8000, '2023-12-05 10:00:00', 301)

INSERT INTO detailed\_rentals(rental\_id, inventory\_id, rental\_date, customer\_id, return\_date, staff\_id)

VALUES (20002, 20001, '2023-12-03 10:00:00', 8003, '2023-12-05 10:00:30', 31)

SELECT \* FROM monthly\_summary\_rentals;

-- F. PROCEDURE CREATION

-- Procedure to refresh the detailed\_rentals and monthly\_summary\_rentals tables

CREATE OR REPLACE PROCEDURE table\_refresh()

LANGUAGE plpgsql

AS $$

BEGIN

      DELETE FROM detailed\_rentals;

      DELETE FROM monthly\_summary\_rentals;

      INSERT INTO detailed\_rentals(

            rental\_id,

            inventory\_id,

            rental\_date,

            customer\_id,

            return\_date,

            staff\_id

            )

      SELECT

            rental\_id,

            inventory\_id,

            rental\_date,

            customer\_id,

            return\_date,

            staff\_id

      FROM rental;

      INSERT INTO monthly\_summary\_rentals(month\_year, total\_rentals)

      SELECT

      TO\_CHAR(DATE\_TRUNC('month', rental\_date), 'YYYY-MM') AS month\_year,

      COUNT(\*) AS total\_rentals

      FROM detailed\_rentals

      GROUP BY TO\_CHAR(DATE\_TRUNC('month', rental\_date), 'YYYY-MM');

      RETURN;

END;

$$;

-- Procedure call

CALL table\_refresh();

-- Get results

SELECT \* FROM detailed\_rentals;

SELECT \* FROM monthly\_summary\_rentals;

Web Sources

I did not use any external web sources to acquire data or segments of third-party code.