Brandon Jones - bjon675

D191 Advanced Data Management

A: BUSINESS REPORT:

A business report that can be derived from this database is one that shows an accurate list of repeating customers and how often they utilize the dvd rental service. This report would contain a list of customers with their first and last name, email address, physical address, customer number, and the amount of rentals they've purchased. This report would be critical for stakeholders to understand who returns for service and could be used to implement a rewards program for repeating customers.

A1: DATA DESCRIPTION:

The data used to generate this report will be data that uniquely identifies each repeating customer and tallies the amount of rentals they've purchased. The company could use this data once a loyalty program is implemented so they can decide which customers to award.

A2: IDENTIFYING SPECIFIC TABLES:

The tables used to generate this report will be the customer table and rental table. We will use the customer table to extract information about the customer in conjunction with the rental table to discover how many times the customer has purchased a rental.

A3: IDENTIFYING SPECIFIC FIELDS:

The fields for the detailed report report will be customer.id, customer.first_name, customer.last_name, customer.email, customer.address_id, and rental.rental_id. I've also opted to include a rental and return date in the detailed report section, which could be used in the future to determine a timeline for loyalty rewards. This will generate a complete list of rentals made by

customers and we will derive a summary table from this table that will only contain the customer's name, email, and amount of rentals they have purchased.

A4: FIELD TRANSFORMATION:

For the detailed report I transformed the first_name and last_name field into a singular customer_name field that concatenates the first_name and last_name fields. This will improve report readability.

A5: BUSINESS USES:

A business could use this report to get an idea of the amount of repeating customers they have as well as using this data to help implement a rewards program in which customers may receive points or discounts based on their repeating activity.

A6: REPORT FRESHNESS:

In order to keep this report accurate, it is important that it refreshes every three months in order for the report to stay up to date and also allow customers to build up reward incentive by renting DVDs.

F1: DATA FRESHNESS:

Since I have created a stored procedure named 'refresh_detailedreport()' you can simply call this function and it will refresh the detailed report by deleting all of the rows that exist in it and inserting new rows from the database. Once this table inserts new data, it will trigger our update_detailedreport trigger which will call our trigger function that will refresh our summaryreport table. This can be done manually or through an SQL procedure scheduler. I would recommend that this procedure is ran every three months based on returning customer data.

I have added my complete SQL query to the last page of this report:

```
--Delete the table if it already exists, if it does not exist create the detailed table where we will
store all the detailed information
DELETE FROM detailedreport;
DROP TABLE IF EXISTS detailedreport;
CREATE TABLE detailedreport(
        customer_id int,
        customer_name varchar(180),
        email varchar(90),
        address_id int,
        rental_id int,
        rental_date timestamp,
        return_date timestamp
);

--Insert the data into the detailed report from the customer and rental tables
INSERT INTO detailedreport(
        customer_id,
        customer_name,
        email,
        address_id,
        rental_id,
        rental_date,
        return_date
```

```
    )
SELECT customer.customer_id, concat(customer.first_name, ' ', customer.last_name) AS
customer_name, customer.email, customer.address_id, rental.rental_id, rental.rental_date,
rental.return_date
FROM rental
INNER JOIN customer ON rental.customer_id = customer.customer_id;

--Display detailed report
SELECT * FROM detailedreport;

--Delete the summary table if it already exists, then create the table that will hold the summary
information
DROP TABLE IF EXISTS summaryreport;
CREATE TABLE summaryreport(
    customer_name varchar(180),
    email varchar(90),
    rental_count int
);

CREATE OR REPLACE FUNCTION refresh_summaryreport()
RETURNS TRIGGER
AS $$
BEGIN
```

```
    --Clear the table before insertion
    DELETE FROM summaryreport;
    --Populate the summary table from detailed table
    INSERT INTO summaryreport(
        SELECT customer_name, email, COUNT(customer_id) AS rental_count
        FROM detailedreport
        GROUP BY customer_name, email
        ORDER BY rental_count DESC
    );

    RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;
--Create the trigger that will detect updates from the detailedreport and call the refresh trigger function
CREATE TRIGGER update_detailedreport
AFTER INSERT
ON detailedreport
FOR EACH STATEMENT
EXECUTE PROCEDURE refresh_summaryreport();

--the stored procedure that will refresh the detailedreport, effectively refreshing the summmaryreport as well
```

```
--This stored procedure should be ran routinely once every three months to ensure data freshness
CREATE OR REPLACE PROCEDURE refresh_detailedreport()
AS $$
BEGIN
DELETE FROM detailedreport;
INSERT INTO detailedreport(
        customer_id,
        customer_name,
        email,
        address_id,
        rental_id,
        rental_date,
        return_date
        )
SELECT customer.customer_id, concat(customer.first_name, ' ', customer.last_name) AS
customer_name, customer.email, customer.address_id, rental.rental_id, rental.rental_date,
rental.return_date
FROM rental
INNER JOIN customer ON rental.customer_id = customer.customer_id;
END;
$$ LANGUAGE PLPGSQL;

CALL refresh_detailedreport();
```

```
--SELECT * FROM summaryreport;

--SELECT * FROM detailedreport;
```