Aidan Polm

Advanced Data Management

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.

The purpose of this project is to create a business report with a detailed and summary table. The report will answer the question "Which film category is rented most often?". This report will allow stakeholders to optimize their inventory and improve revenue based on changing customer trends and needs. By identifying high and low performing categories, stakeholders can target specific categories for promotional material and increase customer traffic.

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

Summary table: category_id, category_name, category_rentals
Detailed table: rental_id, rental_date, store_id, category_id, category_name

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

category_id – int – This is the primary key and is the id for the type of category.
category_name – varchar (255) – This is the name of the category.
category_rentals – int – The total number rentals for that category across all stores.

rental_id – int – This is the primary key and is the id for each rental transaction.
rental_date – varchar (5) – This is the date of each rental transaction
store_id – int – This is the id for a specific store.
category_id – int – This is the id for the type of category.
category_name – varchar (255) – This is the name of the category.

A.3 Identify at least two specific tables that will provide the necessary data.

The summary table will use the information in the detail table.

Detailed table will use data from:
rental table: rental_id, rental_date, inventory_id, and store_id.
category table: category_id, category_name.

film_category table: Links films to their respective categories via the film_id and category_id.
inventory table: Provides inventory_id and links to store_id, connecting rentals to specific stores.

A.4 Identify at least one field in the detailed table that will require a custom transformation with a user-defined function.

The rental_date field in the rental table will require a custom transformation into a MM/dd format for the rental_date field in the detailed table. This allows easier readability for stakeholders.

A.5 Explain the different business uses of the detailed and summary table.

The summary table is intended to be a brief overview of the category film performance. This will allow stakeholders to quickly see which category films are rented and how often, allowing them to purchase inventory geared towards the customer needs, or run promotional advertisements or sales to increase sales for either high or low performing. For example, if “Comedy” is a low performing category, then stakeholders could make the strategic decision to no longer rent those films and save money on that inventory. However, if “Comedy” is a high performing category, stakeholders can anticipate more inventory needs, or run a promo that encourages more customers to come in and rent that category.

The detailed table allows stakeholders to get a granular look of every rental transaction and the associated category. This will help them identify trends in dates with categories rented and allow them to make strategic decisions for inventory and promotional material for the upcoming season. For example, if it’s a noticeable trend that “Comedy” is rented in May, then the following year, stakeholders can increase “Comedy” inventory and make decisions to bring more customers in for “Comedy” in May.

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

The report should be refreshed monthly. This allows time for clear trends and changes to take place, but it’s not too much time that prevents stakeholders from adjusting their strategy or inventory should something change month over month.

B. Provide original code for the function that performs the transformation identified in A.4.

```
CREATE OR REPLACE FUNCTION date_transform(rental_date timestamp)
RETURNS VARCHAR(5)
LANGUAGE plpgsql
AS
$$
BEGIN
        RETURN TO_CHAR(rental_date, ‘MM/DD’);
        END;
        $$;
```

C. Provide original code that creates the detailed and summary tables to hold your report tables.

```
CREATE TABLE summary (
        category_id int NOT NULL PRIMARY KEY,
        category_name varchar(255),
        category_rentals int
);

CREATE TABLE detailed (
        rental_id int NOT NULL PRIMARY KEY,
        rental_date varchar(5),
        store_id int,
        category_id int,
        category_name varchar(255)
);
```

D. Provide an original query that will extract the raw data needed for the detailed section of your report from the source database.

```
INSERT INTO detailed(rental_id, rental_date, store_id, category_id, category_name)
SELECT
        r.rental_id,
        date_transform(r.rental_date) AS rent_date,
        i.store_id,
        c.category_id,
        c.name AS category_name
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film_category fc ON i.film_id = fc.film_id
JOIN category c ON fc.category_id = c. category_id
ORDER BY rent_date;
```

E. Provide original code that creates a trigger on the detailed table that will continually update the summary table as data is added to the detailed table.

```
CREATE OR REPLACE FUNCTION update_summary()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN
  IF EXISTS (SELECT 1 FROM summary WHERE category_id = NEW.category_id) THEN
    UPDATE summary
    SET category_rentals = category_rentals + 1
    WHERE category_id = NEW.category_id;
  ELSE
    INSERT INTO summary (category_id, category_name, category_rentals)
    VALUES (NEW.category_id, NEW.category_name, 1);
  END IF;

  RETURN NEW;
END;
$$;
```

```
CREATE TRIGGER update_summary_after_insert
AFTER INSERT ON detailed
FOR EACH ROW
EXECUTE FUNCTION update_summary();
```

F. Provide original code that creates a trigger on the detailed table that will continually update the summary table as data is added to the detailed table.

```
CREATE OR REPLACE PROCEDURE refresh()
LANGUAGE plpgsql
AS
$$
BEGIN
DELETE FROM detailed;
INSERT INTO detailed(rental_id, rental_date, store_id, category_id, category_name)
SELECT
        r.rental_id,
        date_transform(r.rental_date) AS rent_date,
        i.store_id,
        c.category_id,
        c.name AS category_name
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film_category fc ON i.film_id = fc.film_id
JOIN category c ON fc.category_id = c. category_id
ORDER BY rent_date;
RETURN;
END;
$$;

CALL refresh();
```

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

To automate the stored procedure, pgAgent would be the best for simplicity and connectivity with PostgreSQL. To maximize efficiency, the procedure should be scheduled to run on the 1st of the month, prior to the start of the working day. This allows time for the tables to be updated and analyze the results for the previous month.

G. Provide a Panopto link.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=678e377e-f2b8-4d4d-a56f-b20b014644d4

H. Acknowledge all utilized sources.

No sources were used.