Cade Fisher
D191: Task 1, Data Analysis
8/18/2024

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 real-world business report using the DVD dataset could analyze which film categories generate the most revenue and compare this performance between different stores. This report would help us understand revenue trends and differences across locations, informing decisions on inventory and marketing strategies.

A1 and A2. Identify the specific fields that will be included in the detailed table and the summary table of the report. Describe the types of data fields used for the report.

Detailed Table		
rental_id	INT	identifier for the rental
inventory_id	INT	identifier for the inventory item
film_id	INT	identifier for the film
store_id	INT	identifier for the store
rental_rate	NUMERIC	Rate charged for renting the film
category_id	INT	identifier for the film category
category_name	TEXT	Name of the film category
rental_duration	INT	Duration of the rental in days
total_rental_amount	NUMERIC	Total rental amount

Summary Table		
store_id	INT	identifier for the store
category_name	TEXT	Name of the film category
rental_total_revenue	NUMERIC	Total revenue generated per store and category

A3. 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.

Rental table: provides data for rental_id, inventory_id, rental_date, and return_date
Inventory table: provides data for film_id and store_id
Film table: Provides data for rental_rate
Film_category table: links the film_id with the category_id
Category table: Provides the category name

A4. 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).

Rental duration: Requires a custom function to calculate the difference between rental start and end dates to determine how long each film was rented.

Total Rental Amount: Requires a custom function to multiply the rental duration by the rental rate to compute the total charge for each rental.

A5. Explain the different business uses of the detailed table section and the summary table section of the report.

The detailed table provides a breakdown of each rental transaction, including rental amounts and film details, which helps with managing inventory, answering customer questions, and tracking financial performance. The summary table aggregates total revenue by store and film category, giving a high-level view that supports financial reporting, strategic planning, and performance analysis.

A6. Explain how frequently your report should be refreshed to remain relevant to stakeholders.

The report should be refreshed monthly to ensure stakeholders have up-to-date information on rentals and revenue, enabling timely and informed decision-making based on recent trends.

B. Provide original code for function(s) in text format that perform the transformation(s) you identified in part A4.

```
-- Function to calculate rental duration
CREATE OR REPLACE FUNCTION calculate_duration(rental_date TIMESTAMP, return_date
TIMESTAMP)
RETURNS INTEGER AS $$
BEGIN
  RETURN EXTRACT(DAY FROM (return_date - rental_date));
END;
$$ LANGUAGE plpgsql;

--SELECT calculate_duration('2024-08-08','2024-08-18');

-- Function to calculate total rental amount
CREATE OR REPLACE FUNCTION calculate_total_rental_amount(rental_duration INTEGER,
rental_rate NUMERIC)
RETURNS NUMERIC AS $$
BEGIN
  RETURN rental_duration * rental_rate;
END;
$$ LANGUAGE plpgsql;

--SELECT calculate_total_rental_amount(10,3.50);
```

C. Provide original SQL code in a text format that creates the detailed and summary tables to hold your report table sections.

```
CREATE TABLE detailed_table (
  rental_id INT,
  inventory_id INT,
  film_id INT,
  store_id INT,
  rental_rate NUMERIC,
  category_id INT,
  category_name TEXT,
  rental_duration INT,
  total_rental_amount NUMERIC
```

```
);
CREATE TABLE summary_table (
  store_id INT,
  category_name TEXT,
  rental_total_revenue NUMERIC,
  PRIMARY KEY (store_id, category_name)
);
```

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 detailed_table (
  rental_id,
  inventory_id,
  film_id,
  store_id,
  rental_rate,
  category_id,
  category_name,
  rental_duration,
  total_rental_amount
)
SELECT
  rental.rental_id,
  rental.inventory_id,
  inventory.film_id,
  inventory.store_id,
  film.rental_rate,
  film_category.category_id,
  category.name,
  calculate_duration(rental.rental_date, rental.return_date),
  calculate_total_rental_amount(calculate_duration(rental.rental_date,
rental.return_date), film.rental_rate)
FROM rental
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id;

INSERT INTO summary_table (store_id, category_name, rental_total_revenue)
SELECT
  store_id,
```

```
  category_name,
  SUM(total_rental_amount) AS rental_total_revenue
FROM
  detailed_table
GROUP BY
  store_id,
  category_name;
```

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 update_summary_table()
RETURNS TRIGGER AS $$
BEGIN

  DELETE FROM summary_table
  WHERE store_id = NEW.store_id
   AND category_name = NEW.category_name;


  INSERT INTO summary_table (store_id, category_name, rental_total_revenue)
  SELECT
    NEW.store_id,
    NEW.category_name,
    SUM(total_rental_amount) AS rental_total_revenue
  FROM detailed_table
  WHERE store_id = NEW.store_id
   AND category_name = NEW.category_name
  GROUP BY store_id, category_name;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Create the trigger
CREATE TRIGGER trigger_update_summary
AFTER INSERT ON detailed_table
FOR EACH ROW
EXECUTE FUNCTION update_summary_table();
```

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_data()
AS $$
BEGIN
        TRUNCATE TABLE detailed_table;

        INSERT INTO detailed_table (
    rental_id,
    inventory_id,
    film_id,
    store_id,
    rental_rate,
    category_id,
    category_name,
    rental_duration,
    total_rental_amount
  )
  SELECT
    rental.rental_id,
    rental.inventory_id,
    inventory.film_id,
    inventory.store_id,
    film.rental_rate,
    film_category.category_id,
    category.name AS category_name,
    calculate_duration(rental.rental_date, rental.return_date) AS rental_duration,
    calculate_total_rental_amount(
      calculate_duration(rental.rental_date, rental.return_date),
      film.rental_rate
    ) AS total_rental_amount
```

```
    FROM rental
    INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
    INNER JOIN film ON inventory.film_id = film.film_id
    INNER JOIN film_category ON film.film_id = film_category.film_id
    INNER JOIN category ON film_category.category_id = category.category_id;

    -- Note: The summary_table will be updated by triggers
END;
$$ LANGUAGE plpgsql;
```

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

You can use cron to automate tasks on Unix-like systems by scheduling scripts to run at specific times, such as executing your stored procedure regularly. It is easy to set up and use but lacks advanced features for handling complex tasks and error management.

Alternatively, Apache Airflow allows you to manage and automate workflows with more flexibility, including scheduling and monitoring tasks like running stored procedures. It offers powerful scheduling and monitoring tools but can be complex to set up and maintain compared to cron.

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 additional resources were used.