Max Sealey

Section A: Business Report

Due to the prevalence of online rentals and streaming services, video rental stores have struggled to stay alive. This business report was commissioned by one of the last remaining video rental franchises to show which film categories have had the most rentals each month. This will allow them to more efficiently stock their inventory according to demand and efficiently allocate advertising resources toward promoting their most popular films, leading to an increase in total revenue. After this data has been recorded over an extended period, the business decision makers will be able to use trends in the data to predict when certain movies will be more popular and adjust their inventory orders and advertising materials accordingly.

A1: Describe the data used for the report &
A3: Identify the specific fields that will be included in the detailed and the summary sections of the report.

The summary table contains each genre (string), which is the primary key for the summary table since all values are distinct and not null. The table also contains the total number of rentals for each genre (int).

The detailed table contains the the rental ID (int, also the primary key), the genre name for each rental (string), the rental date (date), rhe return date (date), and the store ID of the rental location (int).

Variable Name	Detailed/Summary	Database Table	Datatype	Description
genre	BOTH	category	VARCHAR	This column contains distinct values (genre names) in the summary table. The number of instances of each genre populates the total_rentals column. PK
total_rentals	SUMMARY	rental	INT	This corresponds with the number of rentals for each genre.
rental_id	DETAILED	rental	INT	This refers to the specific rental instance and is the primary key for the detailed table,

				since all values are distinct.
rental_date	DETAILED	rental	DATE	This can be used to filter by time period, which is useful when mapping trends over time.
return_date	DETAILED	rental	DATE	This shows when the rental was returned, which helps provide a more complete picture of the data in the detailed table.
store_id	DETAILED	inventory	INT	This is included to show where each film was rented from, which, depending on the demographics of the locals near eacj store, may show variations in the popularity of categories.

A2: 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.

The report takes data from the rental, category, and inventory tables.

Both the summary and detailed tables use the name attribute from the category table to populate the genre field. The summary table also contains the total number of rentals, which is found by counting the rental IDs found in the detailed table for each genre. The detailed table gets those rental IDs from the rental table in the main database and uses them as a primary key.

In addition to the genre and rental ID fields, the detailed table contains the rental date and return date for each rental, also acquired from the rental table, as well as the store ID of the sale from the inventory table.

A4: 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 needed to use a custom transformation to aggregate the number of rentals for each category. I used the COUNT function and grouped by genre, which was necessary to populate the total_rentals fields of the summary table.

A5: Explain the different business uses of the detailed and the summary sections of the report.

The summary table provides an overview of which film categories have been most popular, showing each category and the corresponding total number of rentals. This gives the owners and/or managers a general idea of which categories of films to prioritize when stocking inventory and which categories to make the center focus of their advertising materials.

The detailed table also includes (as well as the genre field in the summary table) the rental ID and date for each individual rental, and the store ID where each rental took place. This is to give the owners and/or managers more freedom in looking at various periods of time, and whether there is a difference in needs between the different stores. If the different stores have different local demographics, that could impact the popularity of categories, so it's important to have a record of where each rental took place.

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

The report should be refreshed once a month, at least for the first couple years, to better map the trends of which categories are more popular during specific seasons or time periods. For example, romantic comedies may be more popular earlier in the year, around January to February, while action movies may be more popular during the summer. In the short term this will help the stores stock their inventories with more popular movies, and eventually they'll be able to anticipate trends regarding the fluctuations in popularity of these categories down to the month. Then they will be better able to make accurate decisions based on historical trends and decrease the refresh frequency.

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

```
CREATE TABLE detailed (
      rental_id INT PRIMARY KEY, -- rental table
      genre VARCHAR(30), -- category table
      rental_date DATE, -- rental table
      return_date DATE, -- rental_table
      store_id INT -- inventory table
);

CREATE TABLE summary (
      genre VARCHAR(30),
      total_rentals INT
);
```

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.

Extracts rental_id, genre, rental_date, return_date, and store_id, then populates detailed table:

```
INSERT INTO detailed (
      rental_id,
      genre,
      rental_date,
      return_date,
      store_id
)
SELECT r.rental_id, c.name, r.rental_date, r.return_date, i.store_id
FROM category AS c
INNER JOIN film_category AS fc
ON c.category_id = fc.category_id
INNER JOIN film AS f
ON fc.film_id = f.film_id
INNER JOIN inventory AS i
ON f.film_id = i.film_id
INNER JOIN rental AS r
ON i.inventory_id = r.inventory_id;
```

Query to verify data's accuracy:

```
SELECT * FROM detailed;
```

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

```
CREATE OR REPLACE FUNCTION update_summary_trigger()
      RETURNS TRIGGER
      LANGUAGE plpgsql
AS $$
BEGIN
      DELETE FROM summary;
      INSERT INTO summary
      SELECT genre, COUNT(rental_id)
      FROM detailed
      GROUP BY genre;
      RETURN New;
END;
$$;
```

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 TRIGGER update_summary
      AFTER INSERT
      ON detailed
```

```
      FOR EACH ROW
      EXECUTE PROCEDURE update_summary_trigger();
```

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 OR REPLACE PROCEDURE data_refresh()
LANGUAGE plpgsql
AS $$
BEGIN
      DROP TABLE IF EXISTS detailed;
      DROP TABLE IF EXISTS summary;

      CREATE TABLE detailed (
      rental_id INT PRIMARY KEY, -- rental table
      genre VARCHAR(30), -- category table
      rental_date DATE, -- rental table
      return_date DATE, -- rental_table
      store_id INT -- inventory table
      );

      CREATE TABLE summary (
            genre VARCHAR(30),
            total_rentals INT
      );

      -- Section C: Extracts raw data from DB and inserts into detailed
table
      INSERT INTO detailed (
            rental_id,
            genre,
            rental_date,
            return_date,
            store_id
      )
      SELECT r.rental_id, c.name, r.rental_date, r.return_date, i.store_id
      FROM category AS c
      INNER JOIN film_category AS fc
      ON c.category_id = fc.category_id
      INNER JOIN film AS f
      ON fc.film_id = f.film_id
      INNER JOIN inventory AS i
      ON f.film_id = i.film_id
      INNER JOIN rental AS r
      ON i.inventory_id = r.inventory_id;

      INSERT INTO summary
      SELECT genre, COUNT(rental_id)
      FROM detailed
      GROUP BY genre;
      RETURN;
END;
$$;
```

```
-- This procedure should be refreshed once a month for the forseeable future
to
-- accurately map trends in the popularity of specific genres of movies
-- over the course of a year. Once the most popular genres and trends have
been identified,
-- decision-makers will be able to tailor how they stock their inventory and
which movies to advertise.
```

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

In order to run a stored procedure on a schedule, you need to create a "job" through an SQL Server Agent and follow the steps to create and attach a schedule (summarized/paraphrased from the Database FAQs article listed below).

H & I. References

Bijay. (2021, October 26). *SQL Server scheduled stored procedure*. DatabaseFAQs.com. Retrieved December 22, 2022, from https://databasefaqs.com/sql-server-scheduled-stored-procedure/