DVD Rental Report

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.

- This business report will use the DVD Dataset from the "Labs on Demand Assessment Environment and DVD Database" to calculate the total rentals in each film category each month. The detailed table and summary table of this report will use various fields to show the relevant data. The data for these fields will be pulled from four tables in the DVD source database. Altogether, the data in this report will provide a detailed and summarized table to demonstrate the total monthly rentals of each film category.

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

Detailed Table Specific Fields	Summary Table Specific Fields
● rental_id ● month_and_year (transformation from rental_date in Part B) ● category_id ● category_name ● film_id ● film_title ● film_description ● inventory_id ● total_category_rentals	● month_and_year (transformation from rental_date in Part B) ● category_id ● category_name ● total_category_rentals

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

- The types of data fields that will be used for the report are INTEGER, TEXT, VARCHAR, and TIMESTAMP.

INTEGER	TEXT	VARCHAR	TIMESTAMP
• rental_id • category_id • film_id • inventory_id • total_category_rentals	• month_and_year • film_description	• category_name • film_title	• rental_date (transformed to month_and_year in Part B)

3. **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.**
    - The rental table, inventory table, film table, and the category table will provide the data necessary for the detailed table and summary table.
        - Rental Table: The rental table will provide data for the rental_date and the rental_id fields.
        - Inventory Table: The inventory table will provide data for the inventory_id field.
        - Film Table: The film table will provide data for the film_id, film_title, and film_description fields.
        - Category Table: The category table will provide the data for the category_id and category_name fields.

4. **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).**
    - One field in the detailed section that will require a custom transformation with a user defined function is month_and_year. This function will be transformed from the rental_date field from the rental table. The rental_date is a timestamp. This should be transformed because the rental_date timestamp can be difficult to read. To improve readability, this data will be transformed to a text data type and it will show the month's name and the year. For example, the rental_date ("2005-05-24", "22:54:33") will be transformed to show the month_and_year field as "May 2005". This transformation is important because it improves the readability of the tables' data.

5. **Explain the different business uses of the detailed table section and the summary table section of the report.**
    - The detailed table section of the report can be used to provide the business with an in-depth analysis of the number of rentals for each category per month. This table not only calculates the amount of rentals for each category during the month, but it also provides the business with additional information about each film. It includes the rental id, month / year the film was rented, category id, category name, film id, title of the film, a description of the film, and the inventory id. This specific information can be used to show trends between the films. For instance, the categories with lower amounts of rentals throughout the month, may be low because the inventory for that category of films is limited. This could suggest that the business should increase the film inventory of a certain category. Another example would be looking at the films' description. This could suggest that the business increases or decreases the number of films with similar storylines. The business can improve by taking these patterns into consideration and making business decisions based on them. This detailed section is useful for the business if they are looking for an in depth analysis of the monthly total rentals for each film category.

    - The summary table section of the report can be used to provide the business with a quick overview of the total monthly rentals for each category. The summary table includes the month / year the film was rented, the category name and id, and the total rentals for that category. It is ordered by total category rentals descending. This means the category with the most rentals will be at the top. For instance, the summary table could show the category of "Horror" being the most popular during September and October. This could suggest that the business should increase their inventory of horror films for these months. The use of the summary table can be used to improve the business because they will be able to quickly find the most popular film categories for each month. From here, they can go to the detailed table to find a more in-depth analysis to make their business decisions. This summarized section is useful for the business if they are just looking for a simple overview of the total number of category rentals in each month.

6. **Explain how frequently your report should be refreshed to remain relevant to stakeholders.**
    - The report should be refreshed monthly to remain relevant to stakeholders. This report calculates the total number of rentals in each category per month. By refreshing the report monthly, stakeholders will be able to view relevant monthly reports on the popularity of each film category. This information can then be

used to make future business decisions. For instance, the data could begin to show a trend that animation films are the most popular in the months of June, July, and August. This could suggest that the business should have more animation films available during the summer season. Overall, the report should be refreshed monthly to remain up-to-date on the total rentals in each category per month so that stakeholders can make informed business decisions.

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

```sql
-- CREATE A FUNCTION TO PERFORM THE TRANSFORMATION IDENTIFIED IN PART A4 --
CREATE OR REPLACE FUNCTION get_month_and_year(rental_date TIMESTAMP)
RETURNS TEXT AS $$
BEGIN
    RETURN TO_CHAR(rental_date, 'Month YYYY');
END;
$$ LANGUAGE plpgsql;
```

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

```sql
-- CREATE DETAILED TABLE --
CREATE TABLE detailed_category_rentals (
    rental_id INTEGER,
    month_and_year TEXT,
    category_id INTEGER,
    category_name VARCHAR(25),
    film_id INTEGER,
    film_title VARCHAR(255),
    film_description TEXT,
    inventory_id INTEGER,
    total_category_rentals INTEGER
);

-- CREATE SUMMARY TABLE --
CREATE TABLE summary_category_rentals (
    month_and_year TEXT,
    category_id INTEGER,
    category_name VARCHAR(25),
    total_category_rentals INTEGER,
    CONSTRAINT summary_pk PRIMARY KEY (month_and_year, category_id)
);
```

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.

```
-- EXTRACT RAW DATA FROM THE SOURCE DATABASE AND INSERT INTO DETAILED TABLE --
INSERT INTO detailed_category_rentals (
    rental_id,
    month_and_year,
    category_id,
    category_name,
    film_id,
    film_title,
    film_description,
    inventory_id,
    total_category_rentals
)
SELECT
    r.rental_id,
    get_month_and_year(r.rental_date) AS month_and_year,
    c.category_id,
    c.name AS category_name,
    f.film_id,
    f.title AS film_title,
    f.description AS film_description,
    i.inventory_id,
    SUM(COUNT(*)) OVER (PARTITION BY get_month_and_year(r.rental_date), c.category_id) AS total_category_rentals

FROM
    rental r
JOIN
    inventory i ON r.inventory_id = i.inventory_id
JOIN
    film f ON i.film_id = f.film_id
JOIN
    film_category fc ON f.film_id = fc.film_id
JOIN
    category c ON fc.category_id = c.category_id
GROUP BY
    r.rental_id,
    get_month_and_year(r.rental_date),
    c.category_id,
    c.name,
    f.film_id,
    f.title,
    f.description,
    i.inventory_id
ORDER BY
    get_month_and_year(r.rental_date) ASC,
    total_category_rentals DESC,
    f.film_id ASC;
```

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 FUNCTION FOR THE TRIGGER ON THE DETAILED TABLE TO CONTINUALLY UPDATE THE SUMMARY TABLE
CREATE OR REPLACE FUNCTION update_summary_table_function()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO summary_category_rentals (
        month_and_year,
        category_id,
        category_name,
        total_category_rentals
    )
    SELECT
        NEW.month_and_year,
        NEW.category_id,
        NEW.category_name,
        COUNT(*) AS total_category_rentals
    FROM
        detailed_category_rentals
    WHERE
        month_and_year = NEW.month_and_year
        AND category_id = NEW.category_id
    GROUP BY
        month_and_year,
        category_id,
        category_name

    ON CONFLICT (month_and_year, category_id) DO UPDATE
    SET
        total_category_rentals = EXCLUDED.total_category_rentals;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


-- CREATE TRIGGER ON DETAILED TABLE TO UPDATE SUMMARY TABLE --
CREATE TRIGGER update_summary_table_trigger
AFTER INSERT ON detailed_category_rentals
FOR EACH ROW
EXECUTE FUNCTION update_summary_table_function();
```

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 A STORED PROCEDURE THAT CAN BE USED TO REFRESH THE DATA IN BOTH TABLES
CREATE OR REPLACE PROCEDURE refresh_tables()
LANGUAGE plpgsql
AS $$
BEGIN

-- Clear the contents (existing data) of the detailed table and summary table:
TRUNCATE TABLE detailed_category_rentals;
TRUNCATE TABLE summary_category_rentals;

-- Perform the raw data extraction from Part D:
INSERT INTO detailed_category_rentals (
    rental_id,
    month_and_year,
    category_id,
    category_name,
    film_id,
    film_title,
    film_description,
    inventory_id,
    total_category_rentals
)

SELECT
    r.rental_id,
    get_month_and_year(r.rental_date) AS month_and_year,
    c.category_id,
    c.name AS category_name,
    f.film_id,
    f.title AS film_title,
    f.description AS film_description,
    i.inventory_id,
    SUM(COUNT(*)) OVER (PARTITION BY get_month_and_year(r.rental_date), c.category_id) AS total_category_rentals
FROM
    rental r
JOIN
    inventory i ON r.inventory_id = i.inventory_id
JOIN
    film f ON i.film_id = f.film_id
JOIN
    film_category fc ON f.film_id = fc.film_id
JOIN
    category c ON fc.category_id = c.category_id
```

```
GROUP BY
    r.rental_id,
    get_month_and_year(r.rental_date),
    c.category_id,
    c.name,
    f.film_id,
    f.title,
    f.description,
    i.inventory_id
ORDER BY
    get_month_and_year(r.rental_date) ASC,
    total_category_rentals DESC,
    f.film_id ASC;

END;
$$;


CALL refresh_tables();
```

1. **Identify a relevant job scheduling tool that can be used to automate the stored procedure.**
    - A relevant job scheduling tool that can be used to automate the stored procedure is pgAgent. This software is a job scheduling tool used for PostgreSQL databases. Since this database is in pgAdmin 4, pgAgent would be able to be used. By using pgAgent, the business can schedule for the stored procedure to be automatically run periodically.