Christa Newing
D326 Performance Assessment
3/23/25

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.

Often, revenue reports need to be summarized for a specific amount of time so that managers can determine which products are selling the most. The DVD rental data has a few months' worth of data, so my report will be on which film category has made the most revenue within this timeframe. The summary report will show the film category and the total amount of revenue. The detailed report will show the payment amounts and date information for each film category.

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

The following fields will be in the *summary table*:
film_category (VARCHAR(255))
total_revenue (INTEGER)

The following fields will be in the *detailed table*:
rental_id (PRIMARY KEY/ FOREIGN KEY INTEGER)
inventory_id (FOREIGN KEY, INTEGER)
filmi (FOREIGN KEY, INTEGER)
category_id (FOREIGN KEY, INTEGER)
category_name (FOREIGN KEY, VARCHAR(255))
payment_id (PRIMARY KEY / FOREIGN KEY, INTEGER)
amount (NUMERIC (5,2))
rental_date (VARCHAR)

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

The types of data fields in this report will be VARCHAR and INTEGER.

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 detailed table will consist of data pulled from the rental, inventory, film, film_category, category, and payment tables.

The summary table will consist of data pulled from the detailed table.

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).**

   The TIMESTAMP in the rental_date column of the detailed table will be transformed to a VARCHAR that reads 'Month DD, YYYY' (EX: March 17, 2025) to clearly show the date without the time for easier readability.

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

   The detailed table has a business use of tracking the dates for payment on each film category rental. This could help determine what time of year brought in the most and least sales for each category.

   The summary table is a quick view of which film category generated the most revenue. This can be important when determining promotions and managing stock levels at each store.

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

   This report should be refreshed every quarter so that new goals for the following quarter can be planned and implemented to show growth based on customer demand.

CODE STARTS BELOW ("-- " represents comments, assignment tasks are in bold)

These queries were used often during the development process:

```
DELETE FROM detailed_table;
    DROP TABLE IF EXISTS detailed_table;
    DROP TABLE IF EXISTS summary_table;

    SELECT * FROM detailed_table;
    -- returns sum of payments for each category
    SELECT SUM(amount), category_name
    FROM detailed_table
    GROUP BY category_name;
```

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

```
CREATE OR REPLACE FUNCTION timestamp_to_string_date (rental_date TIMESTAMP)
RETURNS VARCHAR
LANGUAGE plpgsql
AS $$
BEGIN
-- Function will transform the TIMESTAMP to VARCHAR ex: May 19,2006
RETURN TO_CHAR(rental_date, 'Month DD, YYYY');
END;
$$;

--Test to see if function works for rental date in database
SELECT timestamp_to_string_date('2005-06-15 23:57:20');
-- Successful call reads: June 15, 2005
```

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 INTEGER,
inventory_id INTEGER,
film_id INTEGER,
category_id INTEGER,
category_name VARCHAR(255),
payment_id INTEGER,
amount NUMERIC(5,2),
rental_date VARCHAR,
-- primary key & foreign key constraints
CONSTRAINT detailed_table_pk PRIMARY KEY(rental_id, payment_id),
CONSTRAINT detailed_table_inventory_id_fk FOREIGN KEY(inventory_id) REFERENCES
inventory(inventory_id),
CONSTRAINT detailed_table_film_id_fk FOREIGN KEY(film_id) REFERENCES film(film_id),
CONSTRAINT detailed_table_category_id_fk FOREIGN KEY(category_id) REFERENCES
category(category_id),
CONSTRAINT detailed_table_payment_id_fk FOREIGN KEY(payment_id) REFERENCES
payment(payment_id)
```

```
);


CREATE TABLE summary_table (
film_category VARCHAR(255),
total_revenue INTEGER
);
```

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,
        category_id,
        category_name,
        payment_id,
        amount,
        rental_date
)
SELECT
        r.rental_id,
        r.inventory_id,
        i.film_id,
        fc.category_id,
        c.name AS category_name,
        p.payment_id,
        p.amount,
        timestamp_to_string_date(r.rental_date)

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
JOIN
        payment p ON r.rental_id = p.rental_id;


-- check to show populated tables
SELECT * FROM detailed_table;
SELECT * FROM summary_table;
```

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 max_sum_category()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN

DELETE FROM summary_table;
INSERT INTO summary_table (film_category, total_revenue)
SELECT category_name AS film_category, SUM(amount) AS total_revenue
FROM detailed_table
GROUP BY category_name
ORDER BY total_revenue DESC
LIMIT 1;
RETURN NULL;

END;
$$;

-- trigger statement to call the function on insert
CREATE TRIGGER populate_summary
AFTER INSERT OR UPDATE OR DELETE
ON detailed_table
FOR EACH STATEMENT
EXECUTE PROCEDURE max_sum_category();
```

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 table_refresh()
LANGUAGE plpgsql
AS $$
BEGIN
DELETE FROM detailed_table;
DELETE FROM summary_table;

INSERT INTO detailed_table (
rental_id,
inventory_id,
film_id,
category_id,
category_name,
payment_id,
amount,
rental_date
)

SELECT
r.rental_id,
r.inventory_id,
i.film_id,
fc.category_id,
c.name AS category_name,
p.payment_id,
p.amount,
timestamp_to_string_date(r.rental_date)

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
JOIN
payment p ON r.rental_id = p.rental_id;
```

```
RETURN;
END;
$$;

CALL table_refresh();

-- check to show refreshed tables
SELECT * FROM detailed_table;
-- now has 14596 rows again
SELECT * FROM summary_table;
-- now has 4892 as total_revenue again
```

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

   The best tool to use for job scheduling would be the pgAgent. This report was generated in pgAdmin and is most compatible with the product's agent installation. To keep the most current data for the report, the refresh should happen around 5:00am every first day of the month. This will ensure that the data is ready to pull when a typical workday starts. This frequency also makes sense for quarterly updates because there will be less runtime needed on the day of reporting and a more efficient updating process that keeps the most current information within the database.

G. Provide a Panopto video recording that includes the presenter and a vocalized demonstration of the functionality of the code used for the analysis.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=5763cde0-a11a-4de3-8164-b2a9016e4b5a

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 sources were used for third-party code.