Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business 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.*

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

   The primary inquiry for the report revolves around determining the top 10 revenue-generating movies in the fictional DVD rental service company. Key data for the report encompasses movie titles, corresponding database movie IDs, rental rates, rental frequency, and total revenue generated.

2. *Describe the types of data fields used for the report. The report integrates data fields such as film_id, title, rental_rate, times_rented, and revenue.*

   These fields are instrumental in providing a comprehensive analysis of the DVD rental service's performance.

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.*

   Two crucial tables, the "film" table and the "rental" table, are employed to extract necessary information. The "film" table contributes data on movie titles and film IDs, while the "rental" table provides insights through the rental_id field.

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 times_rented field in the detailed table is created using a count function, tracking the total number of times a movie has been rented. The revenue field is derived through a custom transformation, multiplying the count of times_rented by the respective rental rate to showcase the financial contribution of each film.

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

   The detailed table serves various business purposes, offering a comprehensive view of the highest-performing movies, aiding in strategic promotion, and facilitating inventory management decisions. The summary table, with condensed information on title and revenue, provides a quick overview for efficient decision-making.

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

   To ensure relevance, the report should be updated at least monthly. The dynamic nature of movie popularity, releases, and DVD availability necessitates frequent updates. The implemented code facilitates regular updates, ensuring the report remains current and valuable to stakeholders.

Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business Report--

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

```
-- FUNCTIONS FOR TABLE UPDATES
CREATE OR REPLACE FUNCTION update_rental_info()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
   -- Truncate the rental_info table to remove existing data
   TRUNCATE TABLE rental_info;

   -- Insert data into the rental_info table
   INSERT INTO rental_info (rental_id, film_id, title, rental_rate)
   SELECT
      rental.rental_id,
      inventory.film_id,
      film.title,
      film.rental_rate
   FROM
      rental
   INNER JOIN
      inventory ON rental.inventory_id = inventory.inventory_id
   INNER JOIN
      film ON film.film_id = inventory.film_id
   GROUP BY
      rental.rental_id, film.title, film.rental_rate, inventory.film_id
   ORDER BY
      rental.rental_id ASC;

   RETURN NEW;
END;
$$;

-- Drop the function if needed
-- DROP FUNCTION update_rental_info();
```

```
-- Function to update detailed_table
CREATE OR REPLACE FUNCTION update_detailed()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
   -- Truncate the detailed_table to remove existing data
   TRUNCATE TABLE detailed_table;

   -- Insert data into the detailed_table
   INSERT INTO detailed_table (film_id, title, rental_rate,
rented_times, earnings)
   SELECT
      film_id,
      title,
      rental_rate::money,
      COUNT(film_id) AS rented_times,
      (COUNT(film_id) * rental_rate)::money AS earnings
   FROM rental_info
   GROUP BY film_id, title, rental_rate
   ORDER BY earnings DESC, title
   LIMIT 10;

   RETURN NEW;
END;
$$;

-- Drop the function if needed
-- DROP FUNCTION update_detailed();;
```

Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business Report--

```
-- Function to update summary_table
CREATE OR REPLACE FUNCTION update_summary()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    -- Truncate the summary_table to remove existing
data
    TRUNCATE TABLE summary_table;

    -- Insert data into the summary_table
    INSERT INTO summary_table (title, earnings)
    SELECT
        title,
        earnings
    FROM detailed_table
    ORDER BY earnings DESC, title
    LIMIT 10;

    RETURN NEW;
END;
$$;
```

Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business Report--

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

```
-- DETAILED TABLE
CREATE TABLE detailed_table AS
WITH detailed_data AS (
    SELECT
        rental_info.film_id,
        rental_info.title,
        rental_info.rental_rate::money,
        COUNT(rental_info.film_id) AS times_rented,
        (COUNT(rental_info.film_id) *
rental_info.rental_rate)::money AS revenue
    FROM
        rental_info
    GROUP BY
        rental_info.film_id, rental_info.title,
rental_info.rental_rate
    ORDER BY
        revenue DESC, title
    LIMIT 10
)
SELECT * FROM detailed_data;

-- Display the detailed table
SELECT * FROM detailed_table;

-- Drop the detailed table (if needed)
DROP TABLE detailed_table;
```

```
-- SUMMARY TABLE
CREATE TABLE summary_table AS
SELECT
    detailed_table.title,
    detailed_table.revenue
FROM
    detailed_table
ORDER BY
    detailed_table.revenue DESC, detailed_table.title
LIMIT 10;

-- Display the summary table
SELECT * FROM summary_table;

-- Drop the summary table (if needed)
-- DROP TABLE summary_table;
```

Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business Report--

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.

```
-- Create rental_info table
CREATE TABLE rental_info AS
SELECT
    rental.rental_id,
    inventory.film_id,
    film.title,
    film.rental_rate
FROM
    rental
INNER JOIN
    inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN
    film ON film.film_id = inventory.film_id
GROUP BY
    rental.rental_id, film.title, film.rental_rate, inventory.film_id
ORDER BY
    rental.rental_id ASC;

-- Display the rental_info table
SELECT * FROM rental_info;

-- Drop the rental_info table (if needed)
-- DROP TABLE rental_info;
```

Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business Report--

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.

```
-- Trigger for updating the summary table when data is added
to the detailed table
CREATE OR REPLACE FUNCTION update_summary_trigger()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
   -- Truncate the summary_table to remove existing data
   TRUNCATE TABLE summary_table;

   -- Insert data from detailed_table into summary_table
   INSERT INTO summary_table (title, earnings)
   SELECT
      title,
      earnings
   FROM detailed_table
   ORDER BY earnings DESC, title
   LIMIT 10;

   RETURN NEW;
END;
$$;

-- Create the trigger
CREATE TRIGGER update_summary_trigger
AFTER INSERT ON detailed_table
FOR EACH STATEMENT
EXECUTE FUNCTION update_summary_trigger();
```

Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business Report--

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_tables()
LANGUAGE plpgsql
AS $$
BEGIN
    -- Clear contents of rental_info table
    DELETE FROM rental_info;

    -- Extract raw data for rental_info section
    INSERT INTO rental_info
    SELECT
        rental.rental_id,
        inventory.film_id,
        film.title,
        film.rental_rate
    FROM
        rental
    INNER JOIN
        inventory ON rental.inventory_id =
inventory.inventory_id
    INNER JOIN
        film ON film.film_id = inventory.film_id
    GROUP BY

        rental.rental_id, film.title, film.rental_rate,
inventory.film_id

    ORDER BY

        rental.rental_id ASC;
```

```
  -- Clear contents of detailed table
    DELETE FROM detailed_table;
   -- Extract raw data for detailed section from rental_info
    INSERT INTO detailed_table
    SELECT
        film_id,
        title,
        rental_rate::money,
        COUNT(film_id) AS rented_times,
        (COUNT(film_id) * rental_rate)::money AS earnings
    FROM
        rental_info
    GROUP BY
        film_id, title, rental_rate
    ORDER BY
        earnings DESC, title
    LIMIT 10;

    -- Clear contents of summary table
    DELETE FROM summary_table;

    -- Extract raw data for summary section from detailed_ta
as test ble
    INSERT INTO summary_table
    SELECT
        title,
        earnings
    FROM
        detailed_table
    ORDER BY
        earnings DESC, title
    LIMIT 10;
END;
$$.
```

Okunta Braide
Student ID: #002450037
ADVANCED DATA MANAGEMENT — D191

--Business Report--

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

There are several tools you can use to automate the execution of this stored procedure. Some popular options include:

1. cron: If you're using a Unix-like operating system, you can schedule the execution of the stored procedure using cron jobs.
2. pgAgent: It's a job scheduling agent for PostgreSQL. You can use pgAgent to schedule and automate the execution of database tasks, including stored procedures.
3. Airflow: Apache Airflow is a powerful open-source platform to programmatically author, schedule, and monitor workflows. It supports PostgreSQL and can be used to schedule the execution of stored procedures.

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

- Video provided.

H. Sources.

- No source where used.