<u>**Data Management – DVD Rentals**</u>

John McGinnes

A. Using the DVD Rental Datasets provided, I would like to answer the simple question: how many DVD rentals have been sold by each employee within a given month? This report can be used by management to determine each employee's efficiency and sales skills to potentially improve the lowest performers and reward the highest performers. Rewarding the top employees will give an incentive for everyone to strive for more sales, thus increasing overall revenue. The report can also be easily modified to show the results of any month to track the sales on a regular basis.

A1. The detailed table will include the staff_id, and the employee_name (using the first_name and last_name fields) fields from the Staff dataset as well as the rental_date and the customer_id from the Rental dataset. The summary table will include the employee_name and total_rentals (using the rental_date) fields needed to determine the results for each employee. Both summary table columns use information from the detailed table rather than the original datasets.

A2. The datatypes for each of these fields used in the detailed table are-

staff_id (integer): A unique numeric identifier given to each staff member.

first_name (variable length string): The first name of the employee who sold each movie rental. This is selected by the employee_name function and transformed to part of the employee_name column.

last_name(variable length string): The last name of the employee who sold each movie rental. This is also selected by the employee_name function and transformed to part of the employee_name column.

employee_name (text): Combines the first and last names of each employee based on the information in the detailed table. Displayed as one complete column.

rental_date (date): Used to mark the month, day, and year of each transaction for each rental sold. I will be using the month from this field to select a specific set of data.

customer_id(integer): A unique identifier given to each customer to connect them to each of their rental transactions.

And for the data types in the summary table-

employee_name (text): Combines the first and last names of each employee based on the information in the detailed table.

total_rentals (text): Counts the total number of rentals per each employee using the rows within the rental_date column from the detailed table.

A3. I will be using the existing tables Staff (fields staff_id, first_name, and last_name) and Rental (fields rental_date, customer_id) combined to gather the data to insert into the detailed table. As mentioned above, each column will retain the same data as the datasets except first_name and last_name, which will be combined to employee_name. I will be using this same data to populate my summary table, though that will be taken from the detailed table rather than the individual datasets (Staff and Rental) and changed to custom columns employee_name and total_rentals.

A4. I will be using a function to transform the data from the datasets and combine the first and last for the names of the employees to the employee_name column to improve the readability in the detailed table and make the overall results more concise.

A5. The detailed table can be used to track each sale that occurred and verify each transaction for the month in question (May of 2006 in my example). This data could be used to track the employee's productivity to more specific timeframe if needed. The customer id is also included, which could be helpful to determine which customers or demographic frequently purchase from a given employee. The summary table gives a monthly account of each employee's sales to reward top performers and find areas of improvement for the employees that have fewer sales. It may also assist with scheduling each employee to be sure the top performers are available during peak sales hours.

A6. I would recommend refreshing this report monthly to ensure accurate tracking for employee reviews and use the information to create improvement plans for the employees with sales lower than the preferred amount.

B. Below is the code used for the transformation mentioned in A4. It will combine the first and last name columns to condense the data for better readability-

```
CREATE OR REPLACE FUNCTION employee_name(first_name VARCHAR(45),
last_name VARCHAR(45))
        RETURNS text
        LANGUAGE plpgsql
AS
$$
DECLARE employee_name text;
```

```
        BEGIN
        SELECT CONCAT(first_name,' ',last_name) INTO employee_name;
        RETURN employee_name;
        END;
        $$
        ;
```

C. Code used to create both the detailed and summary tables-

<u>Detailed Table</u>

```
        DROP TABLE IF EXISTS rentals_per_employee_detailed;
        CREATE TABLE rentals_per_employee_detailed (
        staff_id integer,
        employee_name text,
        rental_date date,
        customer_id int
        );
SELECT * FROM rentals_per_employee_detailed;
```

<u>Summary Table</u>

```
DROP TABLE IF EXISTS rentals_per_employee_summary;

CREATE TABLE rentals_per_employee_summary (

employee_name text,

total_rentals text

);
```

D. Code used to Insert the data into the detailed table from the dataset-

```
        INSERT INTO rentals_per_employee_detailed (
        SELECT s.staff_id,employee_name(s.first_name,s.last_name),
        r.rental_date,r.customer_id
        FROM staff s
        JOIN rental r
        ON s.staff_id=r.staff_id
        WHERE rental_date BETWEEN '2005-05-01' AND '2005-05-31'
);
```

E. The trigger below (with function) will automatically update the summary table when any additions are made to the detailed table-

```
        CREATE OR REPLACE FUNCTION update_summary_table() RETURNS
        TRIGGER AS
        $$
        BEGIN
                DELETE FROM rentals_per_employee_summary;
                INSERT INTO rentals_per_employee_summary (
        SELECT d.employee_name,COUNT(d.rental_date) AS total_rentals
        FROM rentals_per_employee_detailed d
        GROUP BY employee_name
        );
                RETURN new;
        END;
        $$
        LANGUAGE plpgsql;

        CREATE TRIGGER trig_update
                AFTER INSERT ON rentals_per_employee_detailed
                FOR EACH ROW
                EXECUTE PROCEDURE update_summary_table()
        ;
```

F. The following code will clear both the detailed and summary tables and extract the data from the dataset again to refresh both tables-

```
CREATE OR REPLACE PROCEDURE refresh_tables()
LANGUAGE plpgsql
AS $$
BEGIN

DELETE FROM rentals_per_employee_detailed;
DELETE FROM rentals_per_employee_summary;

INSERT INTO rentals_per_employee_detailed (
SELECT s.staff_id,employee_name(s.first_name,s.last_name),
r.rental_date,r.customer_id
FROM staff s
JOIN rental r
ON s.staff_id=r.staff_id
WHERE rental_date BETWEEN '2005-05-01' AND '2005-05-31'
);

RETURN;
END
$$
```

F1. A relevant Job Scheduling Tool to automate this procedure every month would be pgagent. I would recommend this over similar scheduling tools since it is a more complete tool for stored procedures and other SQL functions, though it does require a separate install, I believe the functionality would be worth the extra steps. This tool would also be more user friendly to change the month range to reflect the required data.

H. SOURCES

No sources were used when creating this report.