1. Business Question: What are the revenue totals for each employee?

Rational

To understand each employee’s effectiveness and productivity as a team member, it may be useful to find out how many rentals they have sold. Using this information, sales strategies and employee training programs can be adjusted to recognize high achievers, identify problem areas, and identify top performers. Monitoring rental sales per employee can also help management make better decisions about employment and resource allocation, ensuring that the company is utilizing its resources to their fullest potential. In general, knowing how many rental sales each employee produces can help the business improve its financial performance and keep up its competitiveness in the market.

1. Data used

The database contains information about a DVD rental store. This database comprises 15 tables including information about customers, payments, rentals, films, and actors. The information used for the business report relates to sales and revenue made by the retailer.

1. Tables used in these reports, complete with the data fields used in each table

*Staff table:*

* staff\_id (integer) - unique identifier for each staff member
* first\_name (varchar) – first name of the staff member
* last\_name (varchar) - last name of the staff member

*Payment Table:*

* payment\_id (integer) – unique identifier for each payment
* rental\_id (integer) - identifier for the rental associated with the payment
* staff\_id (integer) - identifier for the staff member who processed the payment
* amount (numeric) – amount paid for the rental
* payment\_date (date) – date the payment was processed

*Rental table:*

* rental\_id (integer) – unique identifier for each rental
* inventory\_id (integer) – identifier for the DVD that was rented
* rental\_date (date) – date the rental was made

*Inventory table:*

* inventory\_id (integer) – unique identifier for each DVD in the store’s inventory
* film\_id (integer) – identifier for the film associated with the DVD

*Film table:*

film\_id (integer) – unique identifier for each film in the store’s

collection

1. Fields to be included in the detailed and the summary sections of the report

*The detailed table is created by combining data from the staff, payment, rental,*

*Inventory, and film tables. The detailed table includes the following fields:*

* amount (numeric) – amount paid for the rental
* payment\_date (date) – date the payment was processed
* rental\_id (integer) – identifier for the rental associated with the payment
* staff\_id (integer) – identifier for the staff member who processed the payment
* full\_name (varchar) – full name of the staff member who processed the payment
* inventory\_id (integer) – identifier for the DVD that as rented
* film\_id (integer) - identifier for the film associated with the DVD
* rental\_date (date) – date the rental was made

*The summary table is created by aggregating data from the detailed table. The*

*summary table includes the following fields:*

* staff\_id (integer) – identifier for the staff member who processed the payment
* full\_name (varchar) – full name of the staff member who processed the payment
* total\_revenue (numeric) – total revenue generated by the staff member

*The rationale for these tables is to have a detailed view of the sales revenue*

*generated by the store, as well as a summarized view of the same data, aggregated*

*by staff members. This information can be used to analyze sales trends, evaluate*

*staff performance and make informed business decisions.*

1. Custom transformation and justification

We are transforming the first and last names in the detailed table by concatenation into a full name. This will enable greater readability and use by the teams accessing these data. The database’s raw data, such as the payment amount, rental date, staff ID, inventory ID, film ID, etc., is not helpful to answer this question in its raw form. We can calculate the total income generated by each employee by aggregating the payment amount by staff id and full name after processing the raw data. When deciding on pay, promotions, and other incentives, management can utilize this information to identify which employees are making the biggest contributions to the company. This data can also be utilized to assess performance, track trends over time, and pinpoint areas that require improvement. Therefore, we are also transforming the amount filed from the detailed table through aggregation to determine the total amount of revenue generated by each employee.

5. Business uses of the detailed and the summary sections of the report

The detailed section of the report will be used to store the transactional data for each rental, payment, and staff member, as well as other relevant information, such as inventory and film data. The summary section, which will offer insights into the overall success of the rental business, will be built on the foundation of this section. The performance of each employee will be evaluated by the executive team in terms of revenue generated in the report’s summary section. By adding together the sales total from the detailed part, the total revenue for each employee will be determined. This data will give the executive and marketing teams important insights into the performance of certain employees, enabling them to spot great performers and strengthen their areas of weakness. Since the data is compiled and presented in a more comprehensive style in the summary part, analysis will likewise be more effective and streamlined.

6. Frequency for refreshing the report and justification

The stored procedure will be run monthly to match the decision-making time frames of the executive and marketing teams.

F. Stored procedure schedule and justification

The stored procedure will be run on a monthly schedule to ensure data freshness using the job scheduling tool pgAgent. This tool can be configured to run the refresh procedure on a monthly schedule. The monthly frequency of refreshing the data is necessary because the executive and marketing teams need to know the revenue totals for each employee to know which employees are performing well, or which employees may need additional training or support.

G. Programming environment

My programming environment consists of a Windows operating system, the Labs On-Demand virtual machine, the pgAdmin4 interface, and the Postgres Structured Query Language program.

H. Web sources

No sources used

1. Sources

No sources used

Code:

--Business Question: What are the revenue totals for each employee?

--B: CREATE detailed table

DROP TABLE IF EXISTS detailed\_concat;

CREATE TABLE detailed\_concat(

amount money,

payment\_date varchar(30),

rental\_id INT,

staff\_id INT,

full\_name varchar(50),

inventory\_id INT,

film\_id INT,

rental\_date TIMESTAMP WITHOUT TIME ZONE

);

SELECT \* FROM detailed\_concat;

--CREATE summary table

DROP TABLE IF EXISTS summary;

CREATE TABLE summary (

staff\_id INT,

full\_name varchar(50),

total\_revenue money

);

--TO view empty summary table

SELECT \* FROM summary;

DROP FUNCTION full\_name\_function(first\_name varchar(45), last\_name varchar (45));

CREATE OR REPLACE FUNCTION full\_name\_function(first\_name varchar(45), last\_name varchar (45))

RETURNS varchar AS $full\_name$

DECLARE full\_name varchar;

BEGIN

RETURN CONCAT(first\_name,' ',last\_name);

END; $full\_name$ LANGUAGE plpgsql;

--SELECT full\_name\_function('John', 'Smith');

-- C. Extract raw data from DVD Rental database into detailed table

DELETE FROM detailed\_concat;

INSERT INTO detailed\_concat(

amount,

payment\_date,

rental\_id,

staff\_id,

full\_name,

inventory\_id,

film\_id,

rental\_date)

SELECT

p.amount,

p.payment\_date,

r.rental\_id,

s.staff\_id,

full\_name\_function(s.first\_name,s.last\_name),

i.inventory\_id,

f.film\_id,

r.rental\_date

FROM staff AS s

INNER JOIN payment AS p ON p.staff\_id = s.staff\_id

INNER JOIN rental AS r ON r.rental\_id = p.rental\_id

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id;

--To view contents of detailed table

--SELECT \* FROM detailed;

--To verify accuracy of data, compare the aggregated sales amount in the summary table to the raw data

SELECT COUNT(payment\_id) FROM payment;

SELECT SUM(amount):: money FROM payment

SELECT SUM(amount) FROM detailed\_concat

--D. CREATE FUNCTION refreshing the summary table with a data transformation

--transforming sales\_amount from the detailed table with an aggregation

DROP FUNCTION refresh\_summary\_table();

CREATE OR REPLACE FUNCTION refresh\_summary\_table()

RETURNS trigger AS $$

BEGIN

DELETE FROM summary;

INSERT INTO summary (staff\_id, full\_name, total\_revenue)

SELECT staff\_id, full\_name, SUM(amount)

FROM detailed\_concat

GROUP BY staff\_id, full\_name;

RETURN NEW;

END; $$ LANGUAGE plpgsql;

--E. CREATE TRIGGER

--DROP TRIGGER refresh\_summary\_trigger ON detailed\_concat;

CREATE TRIGGER refresh\_summary\_trigger

AFTER INSERT ON detailed\_concat

FOR EACH STATEMENT

EXECUTE PROCEDURE refresh\_summary\_table();

--F. CREATE STORED PROCEDURE

--To be automated to run on a monthly basis, the last day of every month

--Use the external pfAgent application as a job scheduling tool

CREATE OR REPLACE PROCEDURE refresh\_reports()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM detailed\_concat;

INSERT INTO detailed\_concat(

amount,

payment\_date,

rental\_id,

staff\_id,

full\_name,

inventory\_id,

film\_id,

rental\_date)

SELECT

p.amount,

p.payment\_date,

r.rental\_id,

s.staff\_id,

full\_name\_function(s.first\_name,s.last\_name),

i.inventory\_id,

f.film\_id,

r.rental\_date

FROM staff AS s

INNER JOIN payment AS p ON p.staff\_id = s.staff\_id

INNER JOIN rental AS r ON r.rental\_id = p.rental\_id

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id;

END; $$;

-- Verify code

DELETE FROM detailed\_concat;

DELETE FROM summary;

SELECT \* FROM detailed\_concat;

SELECT \* FROM summary;

--To call stored procedure

CALL refresh\_reports();

--To view results

SELECT \* FROM detailed\_concat;

SELECT \* FROM summary;