Joe Bonfanti Jr.

Movie Genres & Yearly Profits

D191 Advanced Data Management

Performance Assessment

**A**:

In what I found to be a captivating investigation of the dynamic crossing between the film industry and financial success, I have produced a well-researched and data-driven business report. This report delves deep into the nuanced relationship between movie genres and profitability. The findings of this report will shed light on the fascinating patterns and correlations that govern this realm, offering valuable insights for the company so that profits will increase.

Our analytical journey will be powered by PostgreSQL as the robust backend architecture, complemented by the user-friendly interface of pgAdmin4 on the frontend, paving the way for insightful data-driven decisions that can propel the company towards greater success.

With our database infrastructure in place, the next step in this venture is to identify and prioritize the top 10 movie genres that have demonstrated the highest profitability every year. We will sift through the data within the “dvdrental” database to extract meaningful insights such as the total profit for every genre. This approach will allow us to pinpoint the genres that have consistently delivered the most significant financial returns over time, supplying value for decision-making within the company. The goal: to unearth the gems of the movie industry that have improved company profits the most, hopefully ensuring a data-backed strategy for more favorable outcomes.

**A1**: columns in detailed and summary tables

Detailed Table

|  |
| --- |
| name |
| amount |
| payment\_date |
| rental\_id |
| inventory\_id |
| film\_id |
| category\_id |

# Summary Table

|  |
| --- |
| name |
| amount |
| payment\_date |

**A2**: Column names, types, and use-case

# Detailed Table

|  |  |  |
| --- | --- | --- |
| name | CHARACTER VARYING (25) | The genre name that will be used to identify the top-selling genre |
| amount | NUMERIC (5,2) | Will show how much money was spent on each rental. Will be used alongside payment\_date |
| payment\_date | TIMESTAMP without time zone | Will grab the year from the date and use it with the sum of “amount” to see how much a given genre made that year |
| rental\_id | INTEGER | Will be used to track what movies were rented in conjunction with payment\_id. |
| inventory\_id | INTEGER | Describes the amount of stock for a given movie. Used to connect the rental table with the inventory table |
| film\_id | SMALLINT | Used in the film, film\_category, and inventory tables to associate category\_id with the film |
| category\_id | SMALLINT | Used in the film\_category and category table to associate film\_id with the genre |

# Summary Table

|  |  |  |
| --- | --- | --- |
| genre | CHARACTER VARYING (25) | Names of genres |
| money\_earned | NUMERIC (7, 2) | Will be used to quantify the total profit amount per genre type |
| year | TIMESTAMP without time zone | Will be used to break down profits per genre every year, looking for trends |

**A3**:

I will need to utilize 6 tables for my analysis: payment, rental, inventory, film, film\_category, and category. With these tables, I should be able to categorize and identify the genres with the highest amount of profit per year.

**A4**:

After digging through the data, I will need to create a custom transformation: money\_earned. This will calculate the total amount of profit for every genre.

Then I will produce another custom transformation from the payment\_date and name it year, which will be used with the genre and money\_earned column showing the profits for the given year for each genre.

**A5**:

The detailed table section of my report will be used for more in-depth analysis. With access to the raw data, it allows me to perform detailed examinations and identify any patterns there may be within that data. With that detailed analysis, I will be able to identify specialized audiences and cater more towards them, resulting in more sales and profit. The detailed portion will provide much more information than required for the summary.

The summary table section of my report will drive decision-making. For example, if we find out that horror movies are the number 1 selling product, then we can make the decision to keep a higher stock of horror films. The other way around, our lowest-selling genre might not be worth investing money into for purchasing more inventory. The summary portion, unlike the detailed, will provide a more condensed version of what we are looking for: yearly profits per genre.

**A6**:

Given that this dataset is based on yearly profits, I believe the report could be refreshed bi-yearly. Every 6 months the data should be updated and investigated into to see if trends among genres have changed or stayed consistent. If the trends begin changing, the company may want to start putting more money into the areas with the most sales.

**B**: Transformations PD & CM

-- Create a trigger that runs the provided code when new data is inserted into a table

CREATE OR REPLACE FUNCTION summary\_function()

RETURNS TRIGGER

AS

$$

BEGIN

DELETE FROM summary;

INSERT INTO summary (

SELECT EXTRACT(YEAR FROM payment\_date) AS year,

name AS genre,

sum(amount) AS money\_earned

FROM detailed

GROUP BY name, (EXTRACT(YEAR FROM payment\_date))

ORDER BY (sum(amount)) DESC

);

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

**C**: SQL code to generate detailed and summary tables UC

-- Delete current table if it exists and create detailed table

DROP TABLE IF EXISTS detailed;

CREATE TABLE detailed (

name CHARACTER VARYING (25),

amount NUMERIC (8,2),

payment\_date TIMESTAMP WITHOUT TIME ZONE,

rental\_id INTEGER,

inventory\_id INTEGER,

film\_id SMALLINT,

category\_id SMALLINT

);

-- Delete current table if exists and create the summary table

DROP TABLE IF EXISTS summary;

CREATE TABLE summary (

year NUMERIC (4),

genre CHARACTER VARYING (25),

money\_earned NUMERIC (8,2)

);

**D**: Inserting data into detailed table UC

INSERT INTO detailed (

name,

amount,

payment\_date,

rental\_id,

inventory\_id,

film\_id,

category\_id

)

select c.name, p.amount, p.payment\_date, r.rental\_id, i.inventory\_id, f.film\_id, c.category\_id

FROM payment p

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

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

**E**: Trigger function PD & CM

-- Create a trigger associated with the detailed table

CREATE TRIGGER refresh\_summary\_trigger

AFTER INSERT ON detailed

FOR EACH ROW

EXECUTE FUNCTION summary\_function();

CREATE TRIGGER refresh\_detailed\_trigger

AFTER INSERT ON payment

FOR EACH ROW

EXECUTE FUNCTION detailed\_function();

**F**: Stored procedure to refresh summary and detailed tables

-- Create a procedure that refreshes the summary tables

CREATE OR REPLACE PROCEDURE summary\_procedure()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM summary;

INSERT INTO summary (

SELECT EXTRACT(YEAR FROM payment\_date) AS year,

name AS genre,

SUM(amount) AS money\_earned

FROM detailed

GROUP BY name, EXTRACT(YEAR FROM payment\_date)

ORDER BY SUM(amount) DESC

);

END;

$$;

-- Create a procedure that refreshes the detailed tables

CREATE OR REPLACE PROCEDURE detailed\_procedure()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM detailed;

INSERT INTO detailed

SELECT c.name, p.amount, p.payment\_date, r.rental\_id, i.inventory\_id, f.film\_id, c.category\_id

FROM payment p

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

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;

END;

$$;

**F1**: A relevant job scheduling tool PG

After spending just a few minutes online, I found that there is one specifically designed for PostgreSQL called pgAgent. It was said to be suitable for organizations using PostgreSQL as their database and pgAdmin4 as their database management tool. It seems to be a great choice for scheduling tasks in this environment.

**G**: Panopto Video

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=90784e59-43cc-4b36-8429-b08800f69462>

**H**: Sources

[PostgreSQL Documentation](https://www.postgresql.org/docs/current/index.html) PD - Links to official documentation for PostgreSQL.

[Udemy Course "Intro to PostgreSQL Databases with Pgadmin"](https://wgu.udemy.com/course/intro-to-postgresql-databases-with-pgadmin/learn/lecture/12154282?start=0#overview) UC - Links to Udemy course involving PostgreSQL with PGadmin4.

[Class Material "SQL for Data Analytics"](https://ebookcentral.proquest.com/lib/westerngovernors-ebooks/reader.action?docID=5888693) CM - Links to the book given in the “course material” section of the course.

[pgAdmin4 Official Documentation](https://www.pgadmin.org/docs/pgadmin4/6.21/pgagent.html) PG - Links to the pgAdmin documentation website.