**D191 Data Report**

Prepared by

Arthur J. Amende

Student ID: 002180137

# INTRODUCTION

This data report and associated SQL tables, functions, and procedures will answer the question:

Who are the top five grossing actors by DVD rental payments?

# A1: DATA DESCRIPTION

The report complies data on each actor and the associated rental payments for DVDs of films they starred in.

# A2: IDENTIFYING SPECIFIC TABLES

Data is collected by a series of INNER JOINS on the tables:

* actor
* film\_actor
* film
* inventory
* rental
* payment

From the ‘actor’ table the actor\_id, first\_name, and last\_name are collected.

From the ‘film\_actor’ table the actor\_id is associated to a film\_id.

From the ‘film’ table the film title is collected from the film\_id.

From the ‘inventory’ table the film\_id is associated to an inventory\_id.

From the ‘rental’ table the inventory\_id is associated to rental\_id.

From the ‘payment’ table the rental\_id is associated to a payment\_id.

In this manner each actor is directly associated to a payment. Multiple actors may be associated to the same payment if they starred in the same film.

# A3: IDENTIFYING SPECIFIC FIELDS

The detailed table, actor\_rental\_payment, has three fields:

* payment\_id (int)
* actor\_id (int)
* amount (numeric(5,2))

payment\_id and actor\_id are used as a Composite Primary Key.

The summary table, actor\_rental\_payment, as three fields:

* actor\_id (int)
* actor\_name (varchar(100))
* sum\_amount (numeric(5,2))

actor\_id is used as the Primary Key.

# A4: FIELD TRANSFORMATION

The function actor\_full\_name(a\_id int) will transform the fields first\_name and last\_name from the actor table into a single full name called actor\_name in the summary table actor\_rental\_payment. The name is combined in the summary table for ease of reading.

The function sum\_rental\_payments(a\_id int) summarizes the amount of payments for each actor from the detail table for the summary table.

# A5: BUSINESS USES

The detail table is used to collect all relevant information into a single table. Currently the required data is spread across six different tables in the database. Creating a single collection point makes the data human readable and allows the data accuracy to be verified.

The detail table has too much data for a person to be able to draw conclusions. The summary table analyzes the data in the detail table to extract information. From this information we can then draw conclusions and answers to the question 'Who are the top grossing actors by DVD rental payments?'.

# A6: REPORT FRESHNESS

The report should be refreshed on a weekly basis every Monday using a job scheduling tool like pgAgent. The primary DVD rental days are Friday, Saturday, and Sunday. Compiling the report every Monday will allow the business to take action of any emerging trends in actor popularity or decline in popularity.

# F1: DATA FRESHNESS

The data should be refreshed on a weekly basis every Monday using the job scheduling tool pgAgent. pgAgent requires the installation of pgAdmin. After pgAdmin is installed pgAgent can be installed and a pgAgent extension created which will create all the tables and functions for the pgAgent to operate (Dias, 2020). Once pgAgent is fully installed a weekly scheduled run of the update\_all() procedure can be created to update the data and report every Monday morning.

# SUMMARY

The top five grossing actors by DVD rental payments are

1. $3129.17 Gina DeGeneres
2. $2543.75 Matthew Carrey
3. $2426.92 Mary Keitel
4. $2403.81 Scarlett Damon
5. $2403.18 Walter Torn

# References

Dias, H. (2020, Feburary 3). *An Overview of Job Scheduling Tools for PostgreSQL*. Retrieved from several nines: https://severalnines.com/database-blog/overview-job-scheduling-tools-postgresql

PostgreSQL. (2021a). *PostgreSQL CREATE PROCEDURE*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-create-procedure/

PostgreSQL. (2021b). *PostgreSQL CREATE TRIGGER*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/creating-first-trigger-postgresql/

# APPENDIX A: CODE









