The purpose of this report is to find the average revenue per film in each category. We will use that information to identify how much of each film category we should stock up on based on the information in the **summary table** in which the higher **average revenue per title** for that category, the more we should stock up on films in that category.

The **summary table**lists the **number of titles** in each **category**. There is an **average revenue per title** column, which is calculated by dividing **rental revenue** by **number of titles** for each **category**. The **summary table** is sorted by **average revenue per title**in descending order. The **detailed table** combines columns from **film, inventory, rental, film category***,* and **category** tables to aggregate the data for use in **summary table***,* The **detailed table** is ordered by **film id** of each film. The **summary table**contains **category, category id, number of titles in category, rental revenue,** and **average revenue per title,** respectively.

The **category**in **summary table** comes from the **name** column in the **category**table while the **category id** in **summary table** comes from **category id**in **category** table. Other columns the **summary table** have identical names to their counterparts in the **detailed table**.

The **number of titles in category**column in **summary table**is a custom transformation that counts the number of titles in each respective category from the **detailed table**. The **rental revenue**column in the **summary table**is a custom transformation that counts the total revenue for each film in their respective category from the **detailed table**. The **average revenue per title** columnin **summary table**is also a custom transformation that divides the **number of titles in category**column by the **rental revenue**column.

When **detailed table**is modified, it invokes the **update summary table trigger** on **detailed table**, which executes the **update summary table**stored procedure, which refreshes the **summary table** to match the respective data taken from the **detailed table**.

A stored procedure to **update detailed table** has been created. It refreshes the entire **detailed table** to match each columns respective source. It should be set to automatically update every week so that we can see use the latest information when making decisions about what film categories to stock up on more. Use the pgAgent to schedule the weekly updates, since it is the official cross-platform solution for job scheduling on PostgreSQL and once set up, it can run on the database server itself, without needing any external connection to the database server.

To schedule the execution of the **update detailed table** stored procedure, first install the pgAgent plugin for PostgreSQL. Then go to pgAdmin and right click “pgAgent Jobs,” select “Create” and then select “pgAgent Job…” In the window that shows up, after specifying a name for this item, select “Steps” than add a new entry, name it “Weekly Update,” and select the **“dvdrental”** in the drop-down menu for Database. Now click the tab “Schedules,” click the “+” symbol on the top right corner, put in your chosen name for the schedule, then for “Start” just let it select the current date, time, and time zone for you. Next click “Schedules” then select the day you want the refresh to occur for the drop-down box. Now for the drop-down boxes of both “Month Days” and “Months” choose “Select All.” Then select the precise “Hours” and “Minutes” through their respective drop-down boxes, to specify the exact time that you want this scheduled procedure to run. (1)

# Works Cited

1. **Creating a pgAgent Job — pgAdmin 4 6.8 documentation. *pgAdmin.* [Online] [Cited: 5 10, 2022.] https://www.pgadmin.org/docs/pgadmin4/latest/pgagent\_jobs.html.**