Business Report

I have made this business report in order to answer the question, how to increase efficiency of the store, making sure there enough copies of popular rents while making sure not to waste money on a surplus of unwanted inventory. In this data report I made two table one shows the detailed data, showing all the data extracted from the tables I used including category, film\_category, film. Inventory and rental databases. The second one being a more summarized version of the data, where in I only show the columns that are deemed the most important.

Code for detailed report

Create table detailed\_report as

/\* This creates the table and then populates it with data it extracted\*/

Select film.film\_id, title, description, release\_year, language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating,

film.last\_update as film\_last\_update, special\_features, fulltext, film\_category.category\_id, film\_category.last\_update as film\_category\_last\_update,

name as category, category.last\_update as category\_last\_update,store\_id, inventory.last\_update as inventory\_last\_update, rental\_id

rental\_date, return\_date, staff\_id, rental.last\_update as rental\_last\_update

from film

inner join film\_category

on film.film\_id = film\_category.film\_id

inner join category

on film\_category.category\_id = category.category\_id

inner join inventory

on film.film\_id = inventory.film\_id

inner join rental

on inventory.inventory\_id = rental.inventory\_id

order by title;

select \*

from detailed\_report

Code for Summary report

Create table Top\_Rented\_DVDs as

Select count(title), title

from detailed\_report

group by title,

order by count(title) desc

In the detailed section the fields included will be film\_id, title, description, release\_year, language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating, last\_update, special\_features and fulltext from the film database. The fields have the type integer, character varying, text, integer, smallint, smallint, numeric, smallint, numeric, mpaa\_rating, timestamp without time zone, text, tsvector respectively. Category\_id and last\_update from the film\_category database. With smallint, and timestamp without time zone and the data types. Name and last\_update from the category database. Character variation and timestamp without time zone are the data types. Store\_id and last\_update from the inventory database. Rental\_id, rental\_date, return\_date, last\_update and staff\_id from the rental database. With integer, timestamp without time zone (x3 in a row), and another integer as the data type.

Each field that has “\_id” suffix is a primary key identifier film\_id identifies the film category\_id identifies the category ect. The last\_update shows when the value was last updated, because last\_update has the same name in multiple tables I will add the prefix of whatever table it was in, for example film\_last\_update. Title, description, release year, rental\_rate, replacement cost, rental\_duration are self-explanatory. Length is the length of the movie, special features have things like behind the scenes and commentary. Fulltext has tags and associations that the movie might have like African, drama ect.

The summarized database includes title, and a count column which counts the number of times the previously mentioned title has been rented. The title shows the title of the film and is character varying text, and count is an integer.

In the detailed section the last\_update in the film database will need to be transformed to specify it is the last time the film was updated in the database. There are many other last\_update columns and having multiple of them will cause confusion.

Using the data provided a wide range of to increase profitability are possible. For example, DVDs stock may be increased or decreased depending on demand. DVDs that aren’t being sold well could go in a bargain bin and DVDs that are, could have their price adjusted. For the data to remain relevant I would recommend the database be updated just before the DVDs are restocked that way we may increase or decrease stock depending on demands.

While the summary database can be used predict how much stock should go to each store, the detailed report can be used to create a “snapshot” of current trends. By snapshot I mean taking the data points gathered on the detailed database and plotting it onto a graph. Doing this would allow you see for example horror movies becoming more popular leading up to October. Then you could advertise certain movies using the information gathered from the created graph. This would help sales.

This code creates a trigger which will activate code to update the summary table every time the detail table is updated.

Create Trigger updateeach

After Insert on detailed\_report

for each row

EXECUTE PROCEDURE update();

And the following is the aforementioned code the trigger activates.

Create function update()

returns trigger

language plpgsql

as $$

Begin

insert into

Top\_Rented\_DVDs(count, title)

values(new.count(title), title);

Return new;

End;

$$

The code below will be used to clear and update the content within the summary and detailed report. I believe that the procedure should be run at the beginning of each month so that the data would be fresh enough to predict patterns and trends.

Create or replace procedure refresh()

language plpgsql

as $$ begin

Truncate table detailed\_report;

Truncate table top\_Rented\_DVDs;

insert into detailed\_report

Select film.film\_id, title, description, release\_year, language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating,

film.last\_update as film\_last\_update, special\_features, fulltext, film\_category.category\_id, film\_category.last\_update as film\_category\_last\_update,

name as category, category.last\_update as category\_last\_update,store\_id, inventory.last\_update as inventory\_last\_update, rental\_id

rental\_date, return\_date, staff\_id, rental.last\_update as rental\_last\_update

from film

inner join film\_category

on film.film\_id = film\_category.film\_id

inner join category

on film\_category.category\_id = category.category\_id

inner join inventory

on film.film\_id = inventory.film\_id

inner join rental

on inventory.inventory\_id = rental.inventory\_id

order by title;

insert into Top\_rented\_dvds

select count(title), title from detailed\_report group by title

order by count(title) desc;

end; $$

Sources

There were no sources web or otherwise, third-party code, in-text citations or references used in this. It is original content.