## Table of Contents

Table of Contents

A. Summarizing a potential real-world written business report that can be created from the DVD Dataset.

* A.1-2. Identifying specific fields  and datatypes in both the detailed table and the summary table.
* [A.3 Identifying specific tables in the dataset needed to populate the Detailed and Summary tables.](#a3)
* [A.4 Custom transformation needed for the Detailed table in the form of a User Defined Function.](#a4)
* [A.5 Explain the different business uses of the detailed table section and the summary table section of the report.](#a5)
* [A.6 Explain how frequently your report should be refreshed to remain relevant to stakeholders.](#a6)

Code for Functions, Procedures, and Triggers.

* [B. User Defined Functions](#b)
* [C and D. Queries for creating tables and updating the detailed table](#c_and_d)
* [E. Trigger for updating the summary table when the detailed table is altered](#e)
* [F. Stored Procedures to automate table creation, table truncation, and the ETL process](#f)
  + F.1 Job Scheduling

References

### 

### A.  Summarizing a potential real-world written business report that can be created from the DVD Dataset.

One real-world written business report that can be created from the DVD Dataset is based on determining the top ranking films in a given quarter through finding out which films were rented  the most during that quarter.

A side question I also had was the number of times a given film was returned late, how many days on average was that film returned late, and whether or not returning a film late was correlated with how many times a film was rented (a hypothesis being that perhaps more sought-after films were more often returned late).

#### A.1-2. Identifying specific fields  and datatypes in both the detailed table and the summary table.

| Description for Detailed Table Columns | Field, Type |
| --- | --- |
| Identifier for the rental | rental\_id INT |
| Identifier for the store\* | store\_id SMALLINT |
| Identifier for the category\* | category\_id INT |
| Identifier for the film | film\_id INT |
| Title of the film | title VARCHAR(255) |
| Genre name of the film | genre\_name VARCHAR(25) |
| Year of release of the film | release\_year INT |
| Length of the film in minutes | length SMALLINT |
| MPAA rating of the film\* | rating MPAA\_RATING |
| Rental rate for the film\* | rental\_rate NUMERIC(4, 2) |
| Amount paid for the rental | amount\_paid NUMERIC(5, 2) |
| Duration of the rental in days | rental\_duration SMALLINT |
| Number of days the film was rented | days\_rented SMALLINT |

Note: \* The reason why I chose to include these identifiers was to leave optional room open for performing other aggregations in the future to perhaps find new insights, like determining the profitability of one store vs another, or determining if rental rate or rating impacted DVD rental rates.

| Description For Summary Table Columns | Field, Type |
| --- | --- |
| Identifier for the film | film\_id INT |
| Genre name of the film | genre\_name VARCHAR(25) |
| Title of the film | title VARCHAR(255) |
| Duration of the rental in days | rental\_duration SMALLINT |
| Number of times the film has been rented | times\_rented INT |
| Number of times the film has been returned late | times\_returned\_late INT |
| Average number of days the film is rented | average\_days\_rented NUMERIC |
| Average number of days the film is returned late | average\_days\_returned\_late NUMERIC |
| Percentage of times the film was returned late | percentage\_returned\_late NUMERIC(5, 2) |

#### A.3 Identifying specific tables in the dataset needed to populate the Detailed and Summary tables.

The detailed and summary table require data from the following tables: category, film category, film, inventory, rental, and payment

The most important tables are film, rental, and payment as these provide the bulk of the information needed to answer the question as to which films are most popular.

The summary table is built on the detailed table, using data from the detailed table after the ETL process is finished for the detailed table.

#### A.4 Custom transformation needed for the Detailed table in the form of a User Defined Function.

The days\_rented column in the detailed table is the result of return\_date - rental\_date, and by default returns an INTERVAL datatype. I decided to convert that datatype into an integer based off of some rounded values, which could be useful for getting a general idea of the approximate days rented at a glance (both for a single rental and for all the rentals of a given film), and could also work well with plotting.

The function acting on the detailed table (excerpt from original extraction query):

INSERT INTO quarterly\_detailed\_report  
 SELECT  
 ...  
 rented\_days\_elapsed(return\_date - rental\_date) AS days\_rented

The code for the User Defined Function in question:

CREATE FUNCTION rented\_days\_elapsed(  
 days\_rented INTERVAL  
)   
RETURNS SMALLINT AS  
$$  
 DECLARE days\_elapsed SMALLINT;  
BEGIN   
 days\_elapsed = ROUND(  
 (EXTRACT (DAY FROM days\_rented) +   
 EXTRACT (HOUR FROM days\_rented)/24 +   
 EXTRACT(MINUTE FROM days\_rented)/(24 \* 60)), 0  
 );  
 RETURN days\_elapsed;  
END;  
$$  
LANGUAGE plpgsql;

See the full code for the User Defined Function [here](file:///X:\Vault\OneDrive\Documents\Projects\Programming\Github\WGU%20Class%20Project\Advanced%20Data%20Management\sql_files\B_user_defined_function.sql)

#### A.5 Explain the different business uses of the detailed table section and the summary table section of the report.

The detailed table provides a much more granular view of each and every single time a film was rented. It includes data such as the duration a given film was rented for, and the amount the renter paid in total, which is important data for the summary table. It also includes data not used by the summary view such as the film rating and length, which is useful information for finding other insights that could be explored at a later time.

The summary table provides data related to how often a given film was rented, the average number of days it was rented for, and the percentage a given film was returned late, among other columns. This data is useful for being able to rapidly query information related to customer behavior regarding a specific film, and to find out which films are the most sought after.

#### A.6 Explain how frequently your report should be refreshed to remain relevant to stakeholders.

I would recommend refreshing the report every quarter, which is easy thanks to the quarterly\_extraction procedure. This procedure takes two DATE parameters, which are start and end times.

### Code for Functions, Procedures, and Triggers.

#### B. User Defined Functions

The User Defined Function (UDF) for transforming data needed for the days\_rented column in the detailed table is called rented\_days\_elapsed. Click [here](file:///X:\Vault\OneDrive\Documents\Projects\Programming\Github\WGU%20Class%20Project\Advanced%20Data%20Management\sql_files\B_user_defined_function.sql).

#### C and D. Queries for creating tables and updating the detailed table

The queries for creating both of the tables and filling the Click [here](file:///X:\Vault\OneDrive\Documents\Projects\Programming\Github\WGU%20Class%20Project\Advanced%20Data%20Management\sql_files\C_&_D_create_tables_extract_data_query.sql)

#### [E. Trigger for updating the summary table when the detailed table is altered](#e)

Code for the trigger is found [here](file:///X:\Vault\OneDrive\Documents\Projects\Programming\Github\WGU%20Class%20Project\Advanced%20Data%20Management\sql_files\E_triggers.sql).

The trigger detailed\_updated\_inserted\_or\_deleted calls a wrapper function q\_summary\_refresh, which calls the q\_summary\_extract procedure with the proper argument. The argument is based off of what initiated the trigger, whether it was an INSERT UPDATE or DELETE.

I took advantage of the TG\_OP variable that is automatically created when a trigger is called (Postgres Docs 43.10 and 39.1, 2023).

#### F. Stored Procedures to automate table creation, table truncation, and the ETL process

To view the stored procedures, click [here](file:///X:\Vault\OneDrive\Documents\Projects\Programming\Github\WGU%20Class%20Project\Advanced%20Data%20Management\sql_files\F_ETL_Procedure.sql)

Note: There are two procedures in this file. Calling quarterly\_detailed\_report will automatically set up the tables if they don’t exist, perform the ETL process needed for the detailed table, then call q\_summary\_extract to set up the ETL process for the summary table.

Details - Three things to consider:

First: You will need to have compiled the rented\_days\_elapsed UDF above before running these procedures.

Second: There are two procedures here that need to be compiled.

* quarterly\_detailed\_report
* q\_summary\_extract

The reason for this, is because q\_summary\_extract can be called for two different reasons.

Either to extract data for the summary table (through quarterly\_detailed\_report) after both the detailed and summary tables are created by quarterly\_detailed\_report.

OR

To extract and refresh data automatically (through the use of a trigger) when the detailed table has an INSERT UPDATE or DELETE applied to it.

Third: Both quarterly\_detailed\_report and q\_summary\_extract take arguments. But all you have to do to get started after compilation is simply call quarterly\_detailed\_report with a start and stop date range. For this data set I recommend calling the procedure like this:

CALL quarterly\_detailed\_report('2005-05-24', '2005-08-23');

That way you’ll get a continuous 91 days worth of data.

#### F.1 Job Scheduling

You can schedule the quarterly\_extraction to be run through the *“pgAgent Jobs”* job scheduling tool. It does not come with PostgreSQL by default, but it can be downloaded and installed through the *Stack Builder for PostgreSQL* for your server. An example of the parameters you would use ensure both the detailed and summary tables are truncated and fresh data is extracted, transformed, and loaded onto the tables every quarter is in the bottom of the F\_ETL\_Procedure.sql file.

### References

PostgreSQL Documentation. (2023). CREATE TRIGGER. <https://www.postgresql.org/docs/15/trigger-definition.html>

PostgreSQL Documentation. (2023). PL/pgSQL - Triggers. <https://www.postgresql.org/docs/15/plpgsql-trigger.html>