A. Report: What are the 25 most rented films?

A1. The data used for the report includes data for each rental, data about each rented film, including category.

A2. The 2 tables that supply absolutely needed data are the rental and film tables. The category table is also included to give extra data on each rental/film.

A3. The detailed section of the report will include rental\_id, film\_id, title, release\_year, rental\_rate, rating, category\_id and category name. The summary section will only include the title of the film and the number of lifetime rentals.

A4. The rental\_id field will need to be transformed by grouping together each individual film and counting the number of rentals each has received. These results will be limited to the top 25.

A5. The summary section of the report gives a very simple, at-a-glance view of the top 25 rented movies. The detailed section of the report can be used to filter rentals by category and other important data like rating, release year etc. This can be used by the business to promote and recommend popular and well-liked films.

A6. The report should be refreshed once a month to remain relevant to stakeholders.

Video URL: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=565474c2-a41e-4a1b-bf0f-ae4e01864df3>

-- B.

-- CREATE detailed table

DROP TABLE IF EXISTS detailed;

CREATE TABLE detailed (

rental\_id integer,

film\_id integer,

title varchar,

release\_year year,

rental\_rate numeric (4,2),

rating mpaa\_rating,

category\_id integer,

category\_name varchar (25)

);

-- To view empty detailed table

-- SELECT \* FROM detailed;

-- CREATE summary table

DROP TABLE IF EXISTS summary;

CREATE TABLE summary (

title varchar,

rental\_amount numeric (14,2)

);

-- To view empty summary table

-- SELECT \* FROM summary;

-- C.

-- Extract raw data from sql database into detailed table

INSERT INTO detailed (

rental\_id,

film\_id,

title,

release\_year,

rental\_rate,

rating,

category\_id,

category\_name)

SELECT

r.rental\_id, f.film\_id, f.title, f.release\_year, f.rental\_rate,

f.rating, c.category\_id, c.name

FROM film AS f

INNER JOIN film\_category AS fc ON fc.film\_id = f.film\_id

INNER JOIN category AS c ON c.category\_id = fc.category\_id

INNER JOIN inventory AS i ON i.film\_id = f.film\_id

INNER JOIN rental AS r ON r.inventory\_id = i.inventory\_id;

-- To view contents of detailed table

-- SELECT \* FROM detailed;

-- To verify accuracy of data, compare the aggregated rental amount in the summary table to the raw data

/\*

SELECT SUM(rental\_amount)

FROM (SELECT COUNT(\*) AS rental\_amount FROM rental

INNER JOIN inventory AS i ON i.inventory\_id = rental.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id

GROUP BY f.film\_id

ORDER BY rental\_amount DESC

LIMIT 25) tmp;

\*/

-- SELECT SUM(rental\_amount) FROM summary;

-- For a more complex, but accurate view of the data, use:

/\*

SELECT COUNT(\*) AS rental\_amount FROM rental

INNER JOIN inventory AS i ON i.inventory\_id = rental.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id

GROUP BY f.film\_id

ORDER BY rental\_amount DESC

LIMIT 25;

\*/

-- SELECT \* FROM summary;

-- D.

-- CREATE FUNCTION refreshing the summary table with a data transformation

-- Transforming rental\_amount from the detailed table with grouping and counting the number of rentals and limiting to the top 25 films.

CREATE FUNCTION summary\_25\_most\_rented\_films()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM summary;

INSERT INTO summary (

SELECT title, COUNT(\*) AS rental\_amount

FROM detailed

GROUP BY film\_id, title

ORDER BY rental\_amount DESC

LIMIT 25);

RETURN NEW;

END; $$ LANGUAGE PLPGSQL;

-- E. CREATE TRIGGER

CREATE TRIGGER summary\_refresh

AFTER INSERT ON detailed

FOR EACH STATEMENT

EXECUTE PROCEDURE summary\_25\_most\_rented\_films();

-- F. CREATE STORED PROCEDURE

-- To be automated to run on a monthly basis, the last day of every month

-- 1. Use the external pgAgent application as a job scheduling tool

CREATE PROCEDURE refresh\_reports()

LANGUAGE PLPGSQL

AS $$

BEGIN

DELETE FROM detailed;

INSERT INTO detailed (

rental\_id,

film\_id,

title,

release\_year,

rental\_rate,

rating,

category\_id,

category\_name)

SELECT

r.rental\_id, f.film\_id, f.title, f.release\_year, f.rental\_rate,

f.rating, c.category\_id, c.name

FROM film AS f

INNER JOIN film\_category AS fc ON fc.film\_id = f.film\_id

INNER JOIN category AS c ON c.category\_id = fc.category\_id

INNER JOIN inventory AS i ON i.film\_id = f.film\_id

INNER JOIN rental AS r ON r.inventory\_id = i.inventory\_id;

END; $$;

-- To call stored procedure

-- CALL refresh\_reports();

-- To view results

-- SELECT \* FROM detailed;

-- SELECT \* FROM summary;