# D191 PA Josh Butcher

A.

1. The data I used will determine who is the most popular actor in movies being rented. I will use the first name ,actor.first_name VARCHAR, and name it first_name VARCHAR(45)) and last name actor.last_name VARCHAR and name it last_name VARCHAR(45)) from the actor table and I will take the rental dates, rental.date_rented DATE, renamed to date_rented DATE  to find out how many times a movie containing the actors has been rented. The summary table will take that data into the full_name VARCHAR(91) and times_rented INT fields.

2. The tables I am using are the rental table joined to the inventory table by the inventory_id. Then I will join to the film_actor table by the film_id. Finally, I am joining the actor table by the actor_id.

3. For the detail table, I am using the actor.first_name and actor.last_name to capture the name of the actor in the movie. I am also using rental.rental_date to get a date for each time a movie was rented containing that actor. In the summary table, I am combining first_name and last_name from the detailed table, to be a full name, and counting the dates of rentals of movies containing that actor from the detailed table to show a total number of rentals, times_rented.

4. I will concatenate first_name and last_name from the detailed table to be used in the summary table as full_name with a space in the middle in the summary table. That will allow me to see the complete name of the actor in one field rather than having it spread between two columns, when there could be multiple actors with the same first name or multiple actors with the same last name. I will also transform the dates of the rentals, date_rented into a total count of how many times a movie was rented containing that actor, times_rented.

5. The DVD rental store may want to stock more of the movies with a particular actor in it, if that actor seems to be popular. Stocking more of the movies containing the top actor would likely increase sales as people would want to watch all the movies containing that actor.

6. I think refreshing the data once each quarter would be enough to satisfy the stakeholders. It would be rare that an actor has multiple movies be released in the same quarter.

B. Create detailed table

```
DROP TABLE IF EXISTS detailed;
CREATE TABLE detailed (   first_name
VARCHAR(45),        last_name
VARCHAR(45),
   date_rented DATE
   );
```

Create summary table

```
DROP TABLE IF EXISTS summary;
CREATE TABLE summary (
full_name VARCHAR(91),
   times_rented INT
   );
```

C. Extract raw data

```
INSERT INTO detailed (
            first_name, last_name, date_rented)
SELECT actor.first_name, actor.last_name, rental.rental_date
FROM rental
JOIN inventory
USING (inventory_id)
JOIN film_actor
USING (film_id)
JOIN actor
USING (actor_id);
```

Verify accuracy of data

Check if the amount of distinct names in table "actor" matches the amount of distinct names in table "detailed"

Multiple actors can appear in the same movie

```
SELECT COUNT(DISTINCT(first_name, last_name)) from actor;
SELECT COUNT(DISTINCT(first_name, last_name)) from detailed;
```

D. Transformation function

Combining the first and last name with concatenation
Counting total number of rental dates each actor has been a part of.

```
DROP FUNCTION IF EXISTS update_summary() CASCADE;
CREATE FUNCTION update_summary()
RETURNS TRIGGER AS $$
BEGIN
DELETE FROM summary;
INSERT INTO summary (
        SELECT
                CONCAT_WS(' ', first_name, last_name),
                COUNT(date_rented)
        FROM detailed
        GROUP BY 1
        ORDER BY 2 DESC);
RETURN NEW;
END; $$ LANGUAGE PLPGSQL;
```

E. Create Trigger

```
DROP TRIGGER IF EXISTS refresh_trigger ON detailed;
CREATE TRIGGER refresh_trigger
AFTER INSERT OR UPDATE ON detailed
FOR EACH STATEMENT
EXECUTE PROCEDURE update_summary();
```

F. Create stored procedure

```
  DROP PROCEDURE IF EXISTS refresh_summary();
  CREATE PROCEDURE refresh_summary()
  LANGUAGE PLPGSQL
  AS $$
  BEGIN
DELETE FROM detailed;
INSERT INTO detailed (
          first_name, last_name, date_rented)
  SELECT actor.first_name, actor.last_name, rental.rental_date
  FROM rental
  JOIN inventory
  USING (inventory_id)
  JOIN film_actor
```

```
USING (film_id)
JOIN actor
USING (actor_id)
ORDER BY 3 DESC;
END; $$;
```

Call stored procedure

```
CALL refresh_summary();
```

View results

```
SELECT * FROM detailed;
SELECT * FROM summary;
```

F 1. The stored procedure can be automated to be run at the first of every quarter.
PgAgent or an external cron jobs can be used to schedule updates. PgAgent offers a nice user interface to easily automate SQL procedures.

G. Panopto Video Link:
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=1c703383-366c-465a-8f15ae8300380e55

H. I did not use any web sources for this Assessment.

I. I did not use any sources, no citations were needed.