Assignment 4

1. Create the DVDRental database in Postgres by adding a new database and “restoring” the DB using the dvdrental.tar file. (All tables will exist in the Public schema)
2. Which customer had the most rentals? Return the first name and last name as a single column and the count of rentals per customer in descending order. *Use aggregation and CTE statement*

**ANSWER:**

Eleanor Hunt

Query to return all customers and their count of rentals:

SELECT COUNT(\*), CONCAT(first\_name,' ',last\_name) AS full\_name

FROM customer

JOIN rental ON customer.customer\_id=rental.customer\_id

GROUP BY customer.customer\_id

ORDER BY COUNT(\*) DESC

;

Query to return only the customer with the most rentals:

WITH customer\_rentals AS (

SELECT COUNT(\*) AS rentals, customer.customer\_id AS ci

FROM customer

JOIN rental ON customer.customer\_id=rental.customer\_id

GROUP BY customer.customer\_id

)

SELECT rentals, CONCAT(first\_name,' ',last\_name) AS full\_name

FROM customer\_rentals

JOIN customer ON customer\_rentals.ci=customer.customer\_id

WHERE rentals = (SELECT MAX(rentals) FROM customer\_rentals)

;

1. Did that customer rent any movie more than once? (provide the query to validate)

**ANSWER:**

No

Query:

WITH customer\_rentals AS (

SELECT COUNT(\*) AS rentals, customer.customer\_id AS ci

FROM customer

JOIN rental ON customer.customer\_id=rental.customer\_id

GROUP BY customer.customer\_id

), max\_rentals AS (

SELECT rentals,ci

FROM customer\_rentals

WHERE rentals = (SELECT MAX(rentals) FROM customer\_rentals)

)

SELECT COUNT(\*), film.film\_id, film.title

FROM customer

JOIN rental ON customer.customer\_id=rental.customer\_id

JOIN inventory ON rental.inventory\_id=inventory.inventory\_id

JOIN film ON inventory.film\_id=film.film\_id

WHERE customer.customer\_id IN (SELECT ci FROM max\_rentals)

GROUP BY film.film\_id

HAVING COUNT(\*)>1

;

1. What is the customer’s favorite movie category? Return the name and the total number of films they have rented in that category.

**ANSWER:**

Sci-fi

Query:

WITH customer\_rentals AS (

SELECT COUNT(\*) AS rentals, customer.customer\_id AS ci

FROM customer

JOIN rental ON customer.customer\_id=rental.customer\_id

GROUP BY customer.customer\_id

), max\_rentals AS (

SELECT rentals,ci

FROM customer\_rentals

WHERE rentals = (SELECT MAX(rentals) FROM customer\_rentals)

), categories AS (

SELECT customer.customer\_id, category.category\_id, category.name

FROM customer

JOIN rental ON customer.customer\_id=rental.customer\_id

JOIN inventory ON rental.inventory\_id=inventory.inventory\_id

JOIN film ON inventory.film\_id=film.film\_id

JOIN film\_category ON film.film\_id=film\_category.film\_id

JOIN category ON film\_category.category\_id=category.category\_id

)

SELECT COUNT(\*), categories.name

FROM categories

WHERE categories.customer\_id IN (SELECT ci FROM max\_rentals)

GROUP BY categories.name

ORDER BY COUNT(\*) DESC

LIMIT 1

;

A screenshot of a computer

AI-generated content may be incorrect.

1. Write a trigger to delete a customer if they become inactive.

**ANSWER:**

CREATE OR REPLACE FUNCTION delete\_inactive()

RETURNS trigger

AS

$BODY$

BEGIN

IF (NEW.activebool=FALSE)

THEN DELETE FROM customer WHERE customer.customer\_id=New.customer\_id ;

END IF;

RETURN NULL; -- because this is an AFTER event trigger

END;

$BODY$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER inactive\_customer

AFTER UPDATE ON customer

FOR EACH ROW

EXECUTE FUNCTION delete\_inactive();

1. How can denormalization aid business users answer questions on the database more efficiently? Would it be more advantageous to create a VIEW or to use a NoSQL technology? List pros and cons of both approaches. Your answer should be a minimum of four paragraphs explaining: how denormalization can help and a comparative analysis for VIEWs vs NoSQL. You will need to perform research online or at the library to answer this question, be sure to cite at LEAST 3 pros and 3 cons for each approach. Cite all sources for your answer.

*What is denormalization?*