CREATE TABLE customer\_reviews (

review\_id SERIAL PRIMARY KEY,

customer\_id INT,

review\_text TEXT

);

INSERT INTO customer\_reviews (customer\_id, review\_text)

VALUES (1, 'This film was fantastic! I highly recommend it.');

SELECT \* FROM actor

WHERE NOT first\_name LIKE 'P%';

SELECT \* FROM actor

WHERE actor\_id BETWEEN 4 AND 10;

SELECT \* FROM actor

WHERE LENGTH(first\_name) > ALL(select LENGTH(first\_name) from actor);

--the following uses the ALL clause to find all customers who have no outstanding DVD's

SELECT customer\_id, first\_name, last\_name

FROM customer

WHERE customer\_id <> ALL (

SELECT

customer\_id

FROM rental

WHERE return\_date IS NULL

);

SELECT customer\_id, first\_name, last\_name

FROM customer c

WHERE EXISTS (

SELECT 1

FROM rental r

WHERE r.customer\_id = c.customer\_id

AND r.return\_date IS NULL

);

SELECT first\_name

FROM customer

WHERE EXISTS (SELECT amount FROM payment WHERE customer.customer\_id = payment.customer\_id AND amount > 10);

–wild card with boolean

SELECT\*, (SELECT EXISTS (select 1 from film where description ILIKE '%squirrel%')) FROM film

WHERE description ILIKE '%squirrel%'

OR description ILIKE '%mouse%'

OR description ILIKE '%chipmunk%'

OR description ILIKE '%hampster%'

SELECT film\_id, title

FROM film

WHERE rental\_rate > SOME (

SELECT rental\_rate

FROM film

WHERE replacement\_cost > 20

);

SELECT first\_name

FROM staff

WHERE staff.first\_name = SOME (

SELECT first\_name

FROM actor);

SELECT c.customer\_id, c.first\_name, c.last\_name, p.amount

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id;

SELECT c.customer\_id, c.first\_name, c.last\_name, COUNT(r.rental\_id) AS rental\_count

FROM customer c

LEFT JOIN rental r ON c.customer\_id = r.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

ORDER BY rental\_count DESC;

SELECT c.customer\_id, c.first\_name, c.last\_name, COUNT(p.payment\_id) AS payment\_count

FROM customer c

LEFT JOIN payment p ON c.customer\_id = p.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

ORDER BY payment\_count DESC;

SELECT a.address, s.first\_name

FROM address a

LEFT JOIN staff s ON a.address\_id = s.address\_id;

SELECT l.language\_id, l.name AS language\_name, f.film\_id, f.title

FROM language l

FULL OUTER JOIN film f ON l.language\_id = f.language\_id;

**Day 3**

-- Get the distinct film titles from the "Action" category and the "Sci-Fi" category

SELECT title

FROM film

WHERE film\_id IN (

SELECT film\_id

FROM film\_category

WHERE category\_id = (

SELECT category\_id

FROM category

WHERE name = 'Action'

)

)

UNION

SELECT title

FROM film

WHERE film\_id IN (

SELECT film\_id

FROM film\_category

WHERE category\_id = (

SELECT category\_id

FROM category

WHERE name = 'Sci-Fi'

)

);

SELECT category\_id, COUNT(\*) AS category\_count

FROM film\_category

GROUP BY category\_id

ORDER by COUNT(\*) ASC;

SELECT c.category\_id, c.name AS category\_name, COUNT(\*) AS film\_count

FROM category c

INNER JOIN film\_category fc ON c.category\_id = fc.category\_id

GROUP BY c.category\_id, c.name

ORDER BY film\_count DESC;

SELECT rating, COUNT(\*) AS film\_count

FROM film

GROUP BY rating;

SELECT c.category\_id, c.name AS category\_name, COUNT(f.film\_id) AS film\_count, SUM(p.amount) AS total\_amount

FROM category c

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

JOIN film f ON fc.film\_id = f.film\_id

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

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

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY c.category\_id, c.name

ORDER BY total\_amount DESC;

—done with fewer joins:

SELECT c.category\_id, c.name AS category\_name, COUNT(\*) AS film\_count, SUM(p.amount) AS total\_amount

FROM category c

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

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

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

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY c.category\_id, c.name

ORDER BY total\_amount DESC;

SELECT f.rating, SUM(p.amount) AS total\_revenue

FROM film f

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

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

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY f.rating

ORDER BY total\_revenue DESC;

SELECT c.customer\_id, c.first\_name, c.last\_name, COUNT(\*) AS rental\_count

FROM customer c

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

GROUP BY c.customer\_id, c.first\_name, c.last\_name

HAVING COUNT(\*) > 10

ORDER BY c.last\_name ASC;

SELECT c.customer\_id, c.first\_name, c.last\_name

FROM customer c

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

ORDER BY last\_name ASC;

SELECT c.customer\_id, c.first\_name, c.last\_name

FROM customer c

ORDER BY last\_name ASC;

SELECT f.rating, SUM(p.amount) AS total\_revenue

FROM film f

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

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

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY f.rating

HAVING SUM(p.amount) > 13000

ORDER BY total\_revenue DESC;

SELECT film\_id, title

FROM film

EXCEPT

SELECT f.film\_id, f.title

FROM film f

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

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

ORDER BY title DESC;

SELECT customer\_id, first\_name, last\_name, (SELECT COUNT(\*) FROM rental WHERE rental.customer\_id = customer.customer\_id) AS rental\_count

FROM customer;

SELECT rental.customer\_id, rental\_count.rental\_count

FROM rental

JOIN (

SELECT customer\_id, COUNT(\*) AS rental\_count

FROM rental

GROUP BY customer\_id

) AS rental\_count

ON rental.customer\_id = rental\_count.customer\_id;

SELECT customer\_id, COUNT(\*) AS rental\_count

FROM rental

GROUP BY customer\_id

SELECT customer\_id, rental\_count

FROM (

SELECT customer\_id, COUNT(\*) AS rental\_count

FROM rental

GROUP BY customer\_id

) AS rental\_counts;

SELECT customer\_id, first\_name, last\_name

FROM customer

WHERE customer\_id IN (

SELECT customer\_id

FROM rental

WHERE return\_date IS NULL

);

SELECT a.city, EXTRACT(MONTH FROM p.payment\_date) AS month, SUM(p.amount) AS total\_payment

FROM payment p

JOIN customer c ON p.customer\_id = c.customer\_id

JOIN address a ON c.address\_id = a.address\_id

GROUP BY a.city, EXTRACT(MONTH FROM p.payment\_date)

HAVING SUM(p.amount) > 1000

ORDER BY total\_payment DESC;

SELECT ct.city, EXTRACT(MONTH FROM p.payment\_date) AS month, SUM(p.amount) AS total\_payment

FROM payment p

JOIN customer c ON p.customer\_id = c.customer\_id

JOIN address a ON c.address\_id = a.address\_id

JOIN city ct ON a.city\_id = ct.city\_id

GROUP BY ct.city, EXTRACT(MONTH FROM p.payment\_date)

HAVING SUM(p.amount) > 31

ORDER BY total\_payment DESC;

SELECT STDDEV(amount) as standard\_deviation, AVG(amount) as average, MIN(amount) as minimum, MAX(amount) as maximum, VARIANCE(amount) as variance

FROM payment;

SELECT category\_id, COUNT(\*) AS category\_count

FROM film\_category

GROUP BY category\_id;

SELECT c.category\_id, c.name AS category\_name, COUNT(\*) AS film\_count

FROM category c

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

GROUP BY c.category\_id, c.name

ORDER BY film\_count DESC;

SELECT c.category\_id, c.name AS category\_name

FROM category c

SELECT c.category\_id, c.name AS category\_name, fc.film\_id

FROM category c

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

SELECT rating, COUNT(\*) AS film\_count

FROM film

GROUP BY rating

ORDER BY COUNT(\*) desc;

SELECT c.category\_id, c.name AS category\_name, COUNT(f.film\_id) AS film\_count, SUM(p.amount) AS total\_amount

FROM category c

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

JOIN film f ON fc.film\_id = f.film\_id

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

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

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY c.category\_id, c.name

ORDER BY total\_amount DESC;

SELECT f.rating, SUM(p.amount) AS total\_revenue

FROM film f

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

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

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY f.rating

HAVING SUM(p.amount) > 13000

ORDER BY total\_revenue DESC;

SELECT c.customer\_id, c.first\_name, c.last\_name, COUNT(\*) AS rental\_count

FROM customer c

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

GROUP BY c.customer\_id, c.first\_name, c.last\_name

HAVING COUNT(\*) > 30;

SELECT f.rating, SUM(p.amount) AS total\_revenue

FROM film f

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

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

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY f.rating

HAVING SUM(p.amount) > 10000

ORDER BY total\_revenue DESC;

SELECT film\_id, title

FROM film

EXCEPT

SELECT f.film\_id, f.title

FROM film f

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

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

**DAY 5**

create function get\_film\_count(len\_from int, len\_to int)

returns int

language plpgsql

as

$$

declare

film\_count integer;

begin

select count(\*)

into film\_count

from film

where length between len\_from and len\_to;

return film\_count;

end;

$$;

select get\_film\_count(40,90);

**DAY 6**

**\*making a view\***

**CREATE OR REPLACE VIEW rented\_out AS**

**SELECT**

**i.inventory\_id,**

**f.title,**

**f.description,**

**f.rental\_duration,**

**f.rental\_rate,**

**i.store\_id,**

**i.last\_update,**

**r.rental\_date,**

**r.return\_date**

**FROM**

**film f**

**JOIN**

**inventory i ON f.film\_id = i.film\_id**

**JOIN**

**rental r ON i.inventory\_id = r.inventory\_id**

**WHERE r.return\_date IS NULL;**

**SELECT\*FROM rented\_out**

**\*create a view then get a rank\***

**CREATE OR REPLACE VIEW movies\_rental\_count AS**

**SELECT**

**f.title AS movie\_title,**

**COUNT(r.rental\_id) AS rental\_count**

**FROM**

**film f**

**JOIN**

**inventory i ON f.film\_id = i.film\_id**

**JOIN**

**rental r ON i.inventory\_id = r.inventory\_id**

**GROUP BY**

**f.title**

**ORDER BY COUNT(r.rental\_id) DESC;**

**SELECT**

**movie\_title,**

**rental\_count,**

**DENSE\_RANK() OVER (ORDER BY rental\_count DESC) AS rank**

**FROM**

**movies\_rental\_count;**

**\*LAG() example\***

**SELECT**

**customer\_id,**

**rental\_id,**

**rental\_date,**

**LAG(rental\_date) OVER (PARTITION BY customer\_id ORDER BY rental\_date) AS previous\_rental\_date,**

**rental\_date - LAG(rental\_date) OVER (PARTITION BY customer\_id ORDER BY rental\_date) AS rental\_time\_diff**

**FROM**

**rental**

**ORDER BY**

**customer\_id, rental\_date;**

**\*WINDOW\***

**SELECT**

**p.customer\_id,**

**r.rental\_id,**

**r.rental\_date, p.amount,**

**SUM(p.amount) OVER my\_window AS cumulative\_revenue,**

**AVG(p.amount) OVER my\_window AS average\_revenue,**

**MAX(p.amount) OVER my\_window AS max\_payment**

**FROM**

**payment p**

**JOIN rental r ON p.rental\_id = r.rental\_id**

**WINDOW**

**my\_window AS (PARTITION BY p.customer\_id ORDER BY r.rental\_date)**

**ORDER BY**

**customer\_id, rental\_date;**

**5th Wave Module 2 day 5 SQL queries:**

SELECT description FROM film

WHERE description ILIKE '%cat%'

OR description ILIKE '%d\_g%'

OR description ILIKE '%bird%';

string\_to\_array([delimited\_column], '[delimiter]')

WITH short\_film AS (

SELECT\*FROM film

LIMIT 20 )

select rating, string\_to\_array(STRING\_AGG(description, '54639 '),'54639 ')

FROM short\_film

GROUP BY rating;

with max\_amount as (select max(amount) as max\_amount

FROM payment)

SELECT\*FROM payment

WHERE amount = (select max(amount) FROM payment)

--create a query that returns all customers tied for the top revenue

CREATE OR REPLACE VIEW top\_customers AS

WITH customer\_revenue as (SELECT customer\_id, sum(amount) as customer\_total

FROM payment

GROUP BY customer\_id

ORDER BY customer\_total desc)

SELECT\*FROM customer\_revenue

WHERE customer\_total = (select max(customer\_total) FROM customer\_revenue)

SELECT\*FROM top\_customers

--If we want to rank movies by the ones rented most

--We should probably only consider the ones in inventory

--to make it fair.

--But, there will be a scenario, where we will need to

--do a full outer join of inventory onto rental in order

--to get a fair rental count FOR EACH FILM.

--there will be another scenario where it won't matter so much.

--Describe each scenario where we need, or don't need the outer join.

SELECT inventory\_id from

inventory

WHERE inventory\_id NOT IN (SELECT i.inventory\_id FROM inventory i

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

SELECT inventory\_id

FROM inventory

WHERE inventory\_id NOT IN (SELECT inventory\_id FROM rental)

with rental\_count\_table as (

SELECT i.film\_id, count(i.film\_id) as rental\_count

FROM inventory i JOIN rental r ON r.inventory\_id = i.inventory\_id

GROUP BY i.film\_id

ORDER by rental\_count )

SELECT\*FROM rental\_count\_table

WHERE rental\_count <=

(SELECT

PERCENTILE\_CONT(0.3) WITHIN GROUP (ORDER BY rental\_count)

FROM rental\_count\_table);

**GENERATE SERIES**

SELECT generate\_series(

'2023-01-01'::date, -- Replace with your start date

'2023-12-31'::date, -- Replace with your end date

'1 day'::interval -- Replace with your desired interval

)::date AS date\_value;

**GETTING TOTAL PAYMENT PER CITY PER MONTH**

select \* from

(select ct.city, month

from city ct

cross join

(select distinct EXTRACT(MONTH FROM p.payment\_date) AS month

from payment p))

left join

(SELECT ct.city as city, EXTRACT(MONTH FROM p.payment\_date) AS month, SUM(p.amount) AS total\_payment

FROM payment p

full JOIN customer c ON p.customer\_id = c.customer\_id

full JOIN address a ON c.address\_id = a.address\_id

JOIN city ct ON a.city\_id = ct.city\_id

GROUP BY city, month)

using (city, month)

Hackerrank Occupation Solution

SELECT MAX(CASE WHEN Occupation = 'Doctor' THEN Name END),

MAX(CASE WHEN Occupation = 'Professor' THEN Name END),

MAX(CASE WHEN Occupation = 'Singer' THEN Name END),

MAX(CASE WHEN Occupation = 'Actor' THEN Name END)

FROM ( SELECT Name, Occupation, ROW\_NUMBER()

OVER (PARTITION BY Occupation ORDER BY Name) AS name\_level FROM OCCUPATIONS ) AS names\_order

GROUP BY name\_level ORDER BY name\_level;

*Alternate solution here:*

with doc\_table as (select Name, DENSE\_RANK() OVER (ORDER BY Name asc) AS id

from OCCUPATIONS

where Occupation = 'Doctor'),

prof\_table as (select Name, DENSE\_RANK() OVER (ORDER BY Name asc) AS id

from OCCUPATIONS

where Occupation = 'Professor'),

sing\_table as (select Name, DENSE\_RANK() OVER (ORDER BY Name asc) AS id

from OCCUPATIONS

where Occupation = 'Singer'),

act\_table as (select Name, DENSE\_RANK() OVER (ORDER BY Name asc) AS id

from OCCUPATIONS

where Occupation = 'Actor')

select doc.Name, prof.Name, sing.Name, act.Name

from doc\_table doc

right join act\_table act on [act.id](http://act.id/) = [doc.id](http://doc.id/)

right join sing\_table sing on [sing.id](http://sing.id/) = [act.id](http://act.id/)

right join prof\_table prof on [prof.id](http://prof.id/) = [sing.id](http://sing.id/)