**NAMES: UWIMANA ANGELIQUE**

**STUDENT ID:28868**

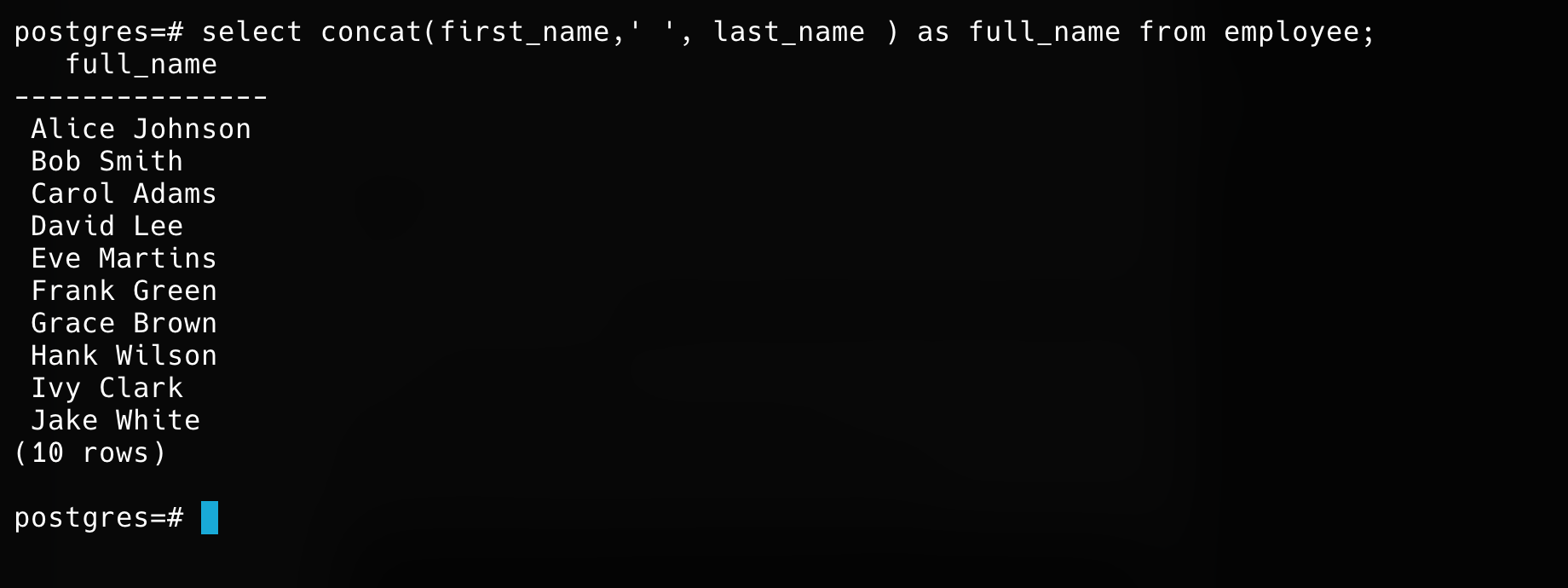
**DBMS ASSIGNMENT 1 group F**

**ABOUT FUNCTIONS**

# STRING FUNCTIONS

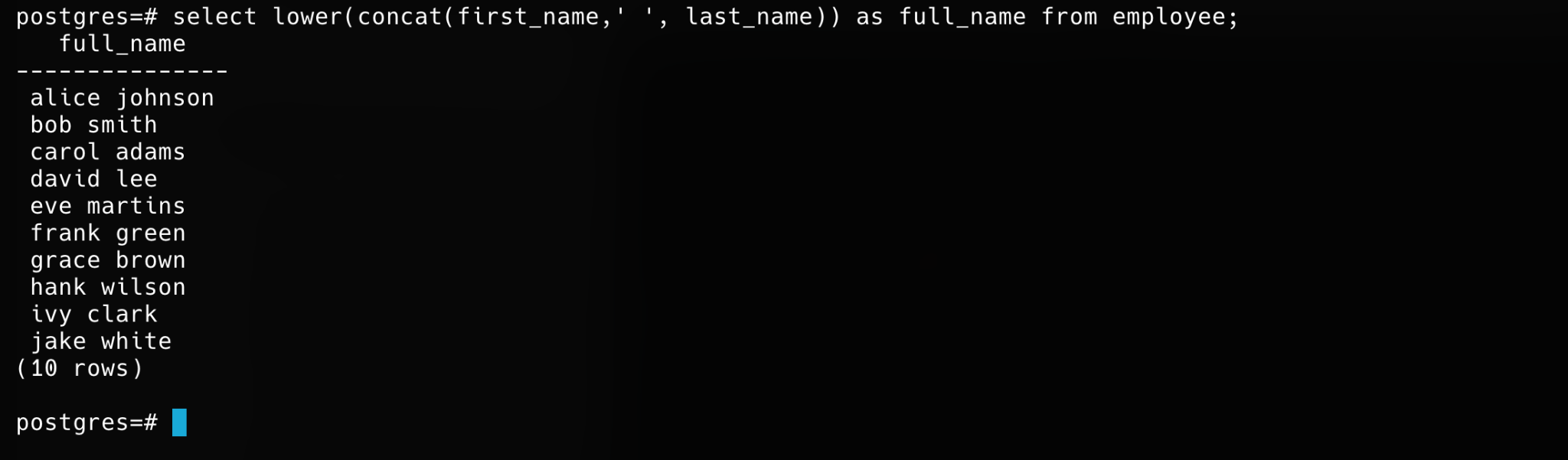
## Concatenate first and last name as full\_name.

select concat(first\_name,' ', last\_name ) as full\_name from employee;



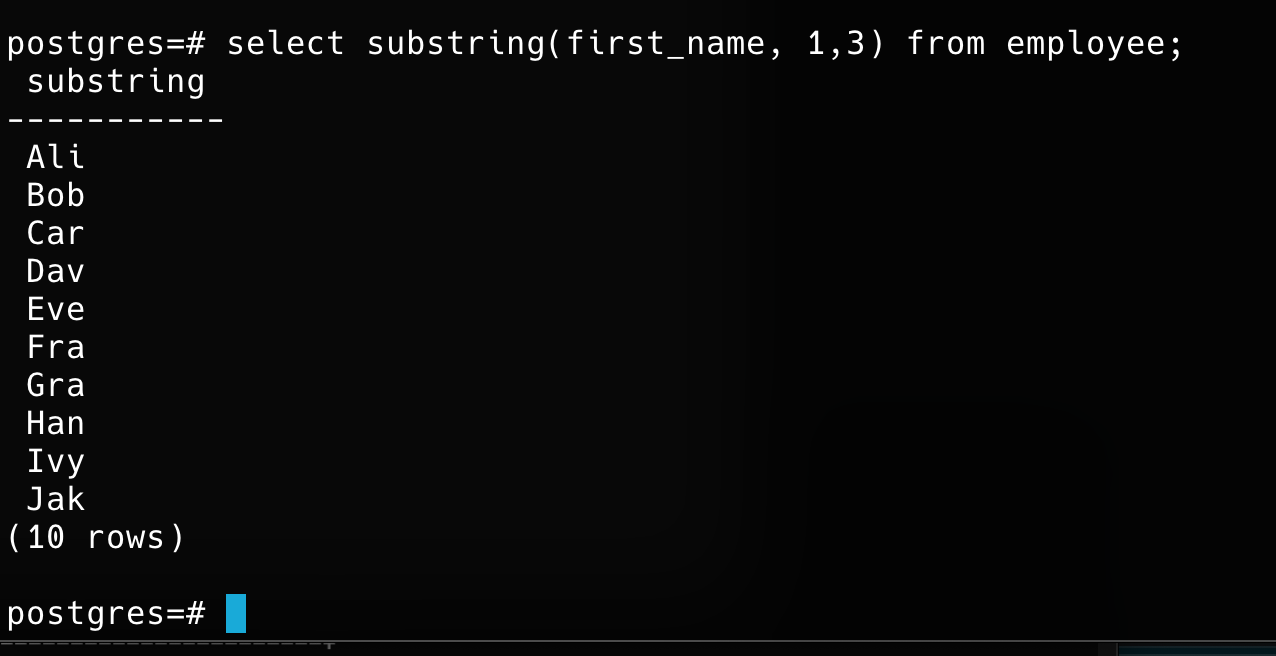
## Convert all employee names to lowercase

select lower(concat(first\_name,' ', last\_name)) AS full\_name from employee;



## Extract first 3 letters of the employee's first name

select substring(first\_name, 1,3) from employee;



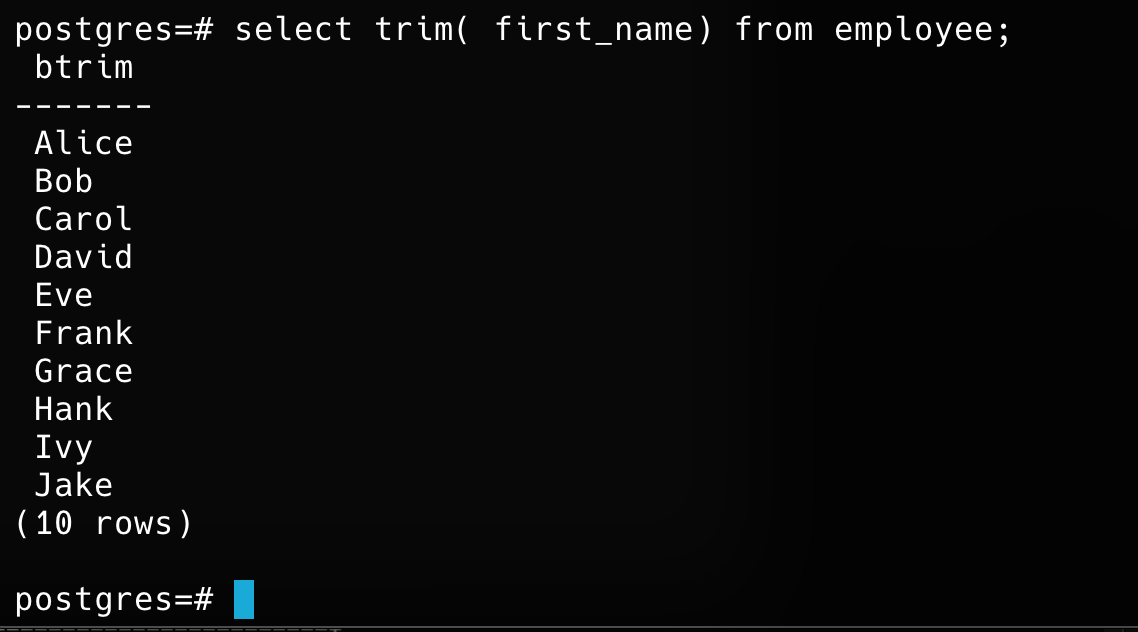
## Replace '@company.com' in email with '@org.com'

select replace(email, '@company.com', '@org.com') from employee;



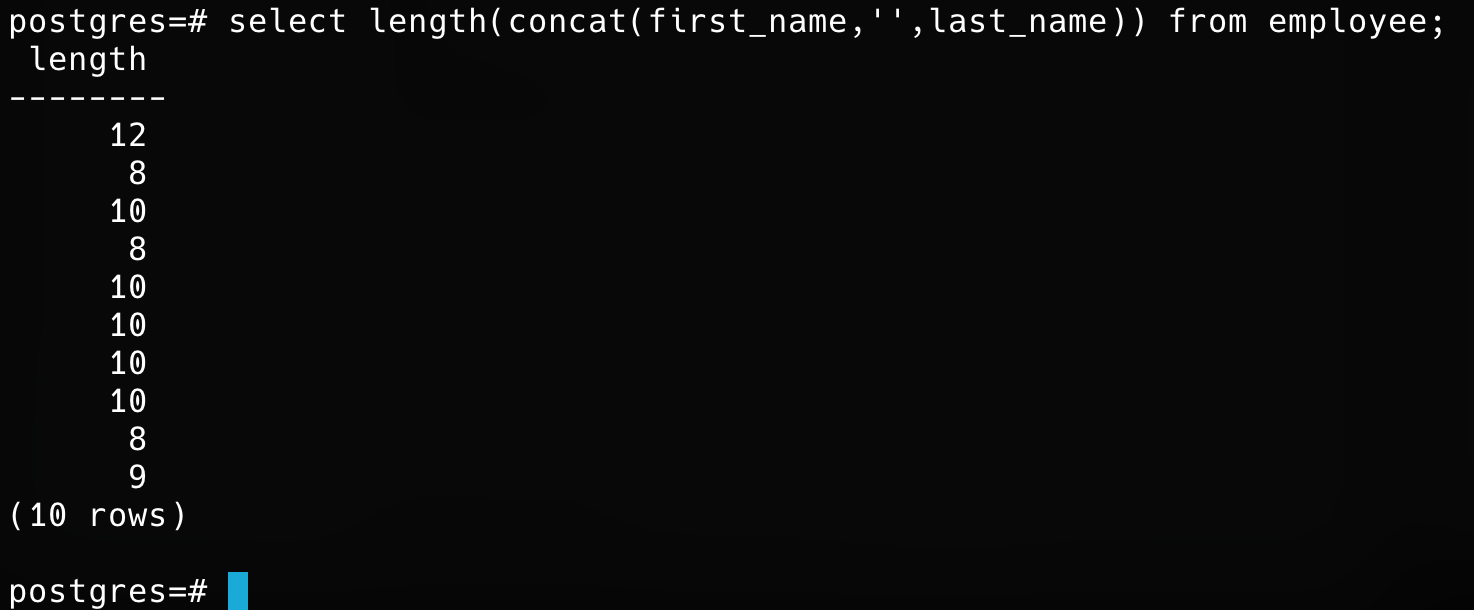
## Trim spaces from a padded string.

select trim( first\_name) from employee;



## Count characters in an employee’s full name

select length(concat(first\_name,'',last\_name)) from employee;

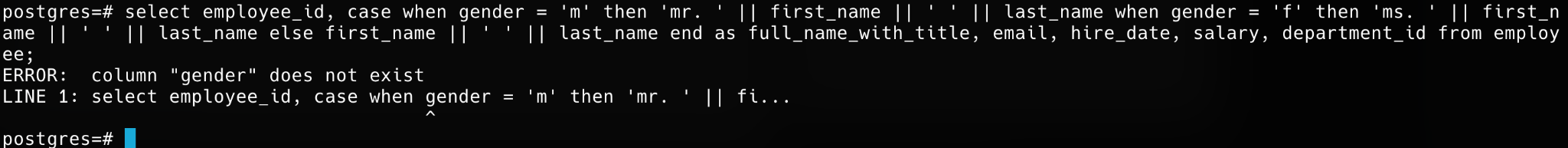


## Find position of '@' in email using INSTR()/CHARINDEX().

select position( '@' in email) as postion from employee;

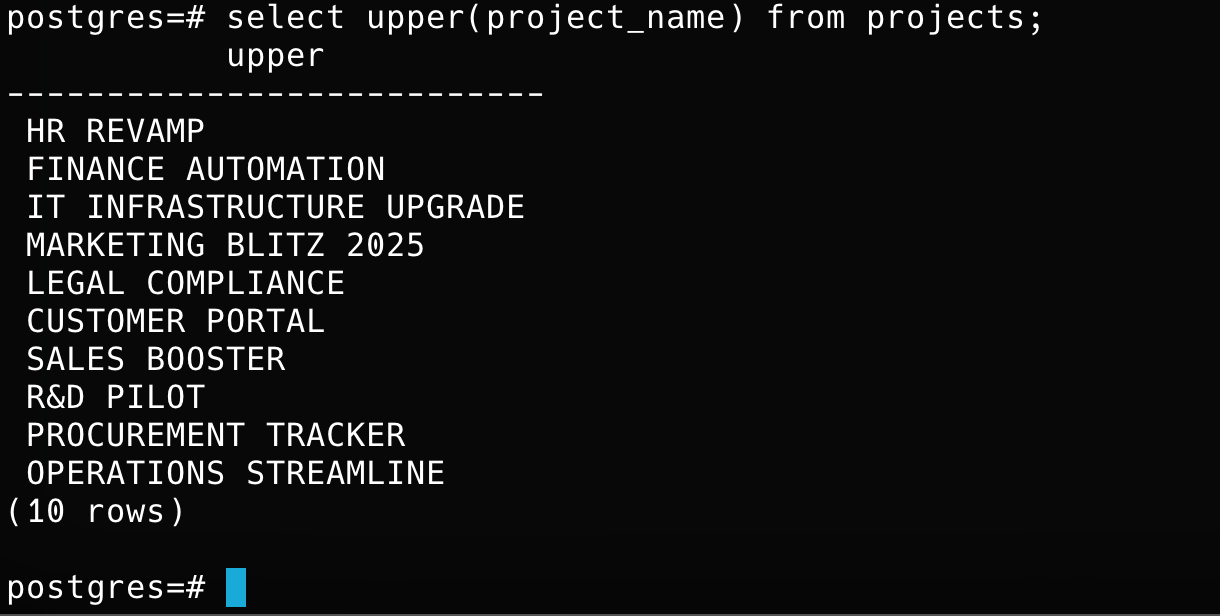
## Add ‘Mr.’ or ‘Ms.’ before names based on gender (assume gender exists).

select employee\_id, case when gender = 'm' then 'mr. ' || first\_name || ' ' || last\_name when gender = 'f' then 'ms. ' || first\_name || ' ' || last\_name else first\_name || ' ' || last\_name end as full\_name\_with\_title, email, hire\_date, salary, department\_id from employee;



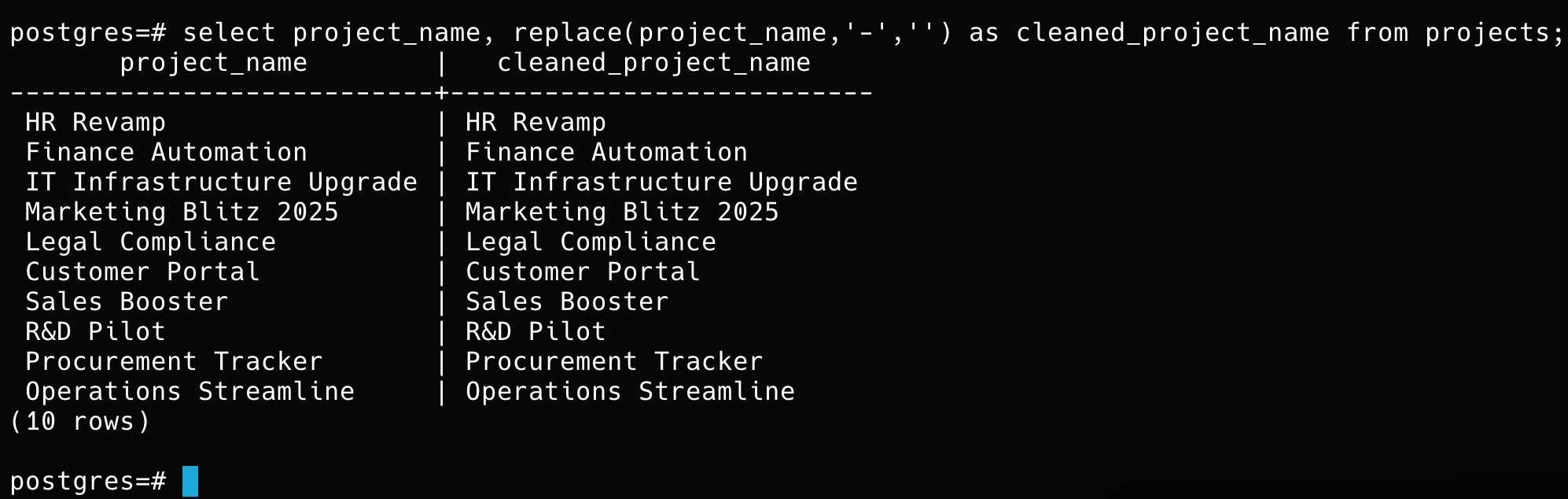
## Format project names to uppercase.

select upper(project\_name) from projects;



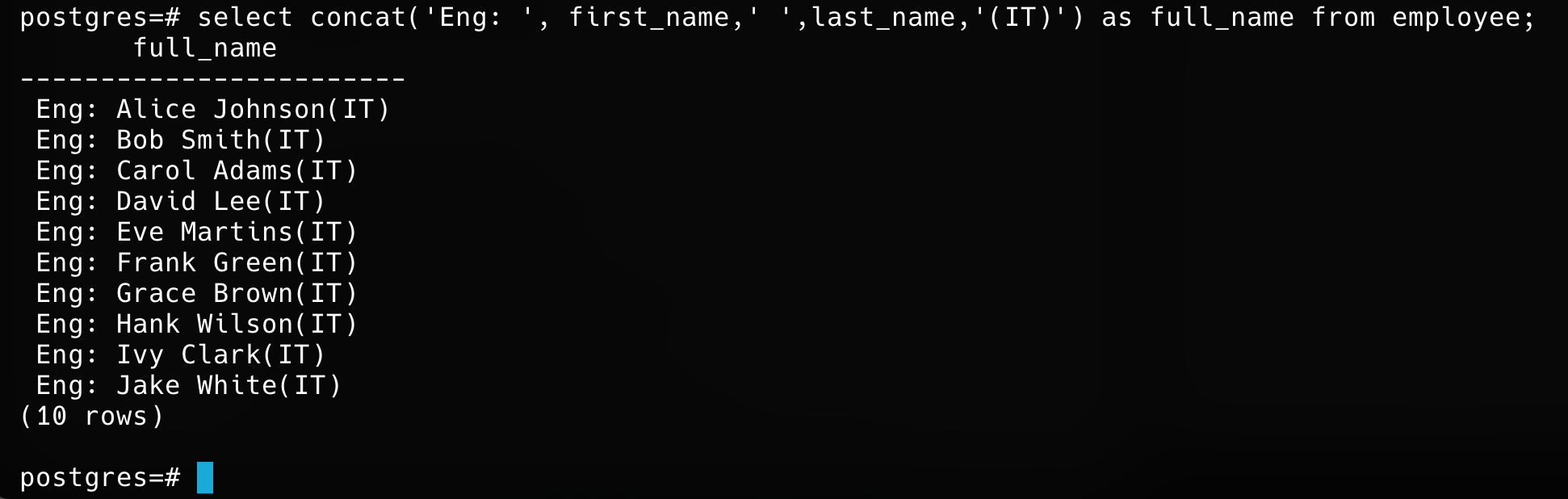
## Remove any dashes from project names.

select project\_name, replace(project\_name,'-','') as cleaned\_project\_name from projects;



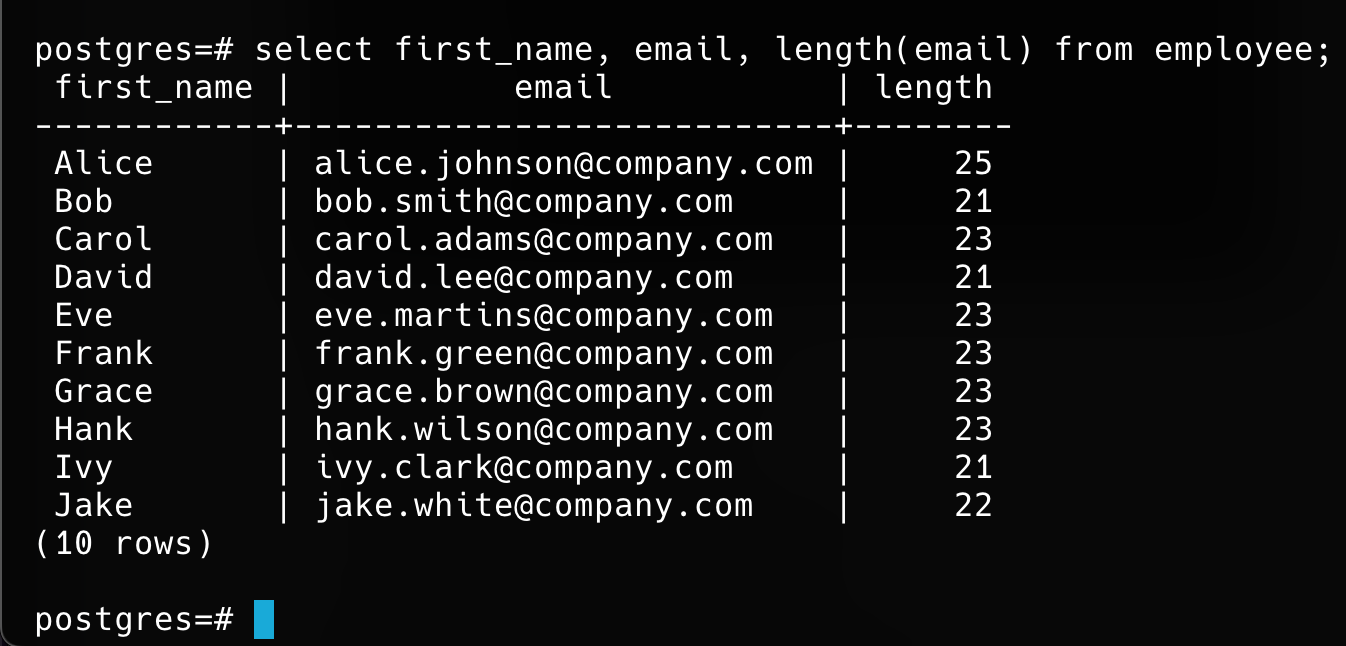
## Create a label like “Emp: John Doe (HR)”

select concat(‘Eng: ', first\_name,' ',last\_name,'(IT)') as full\_name from employee;



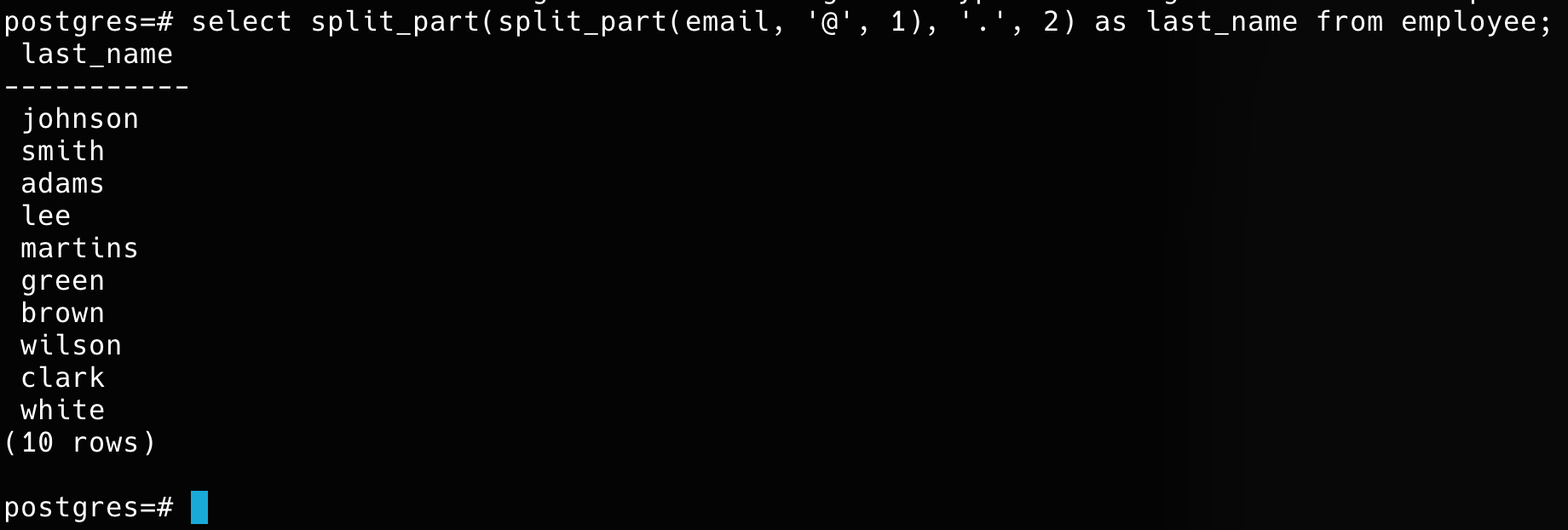
## Check email length for each employees

select first\_name, email, length(email) from employee;



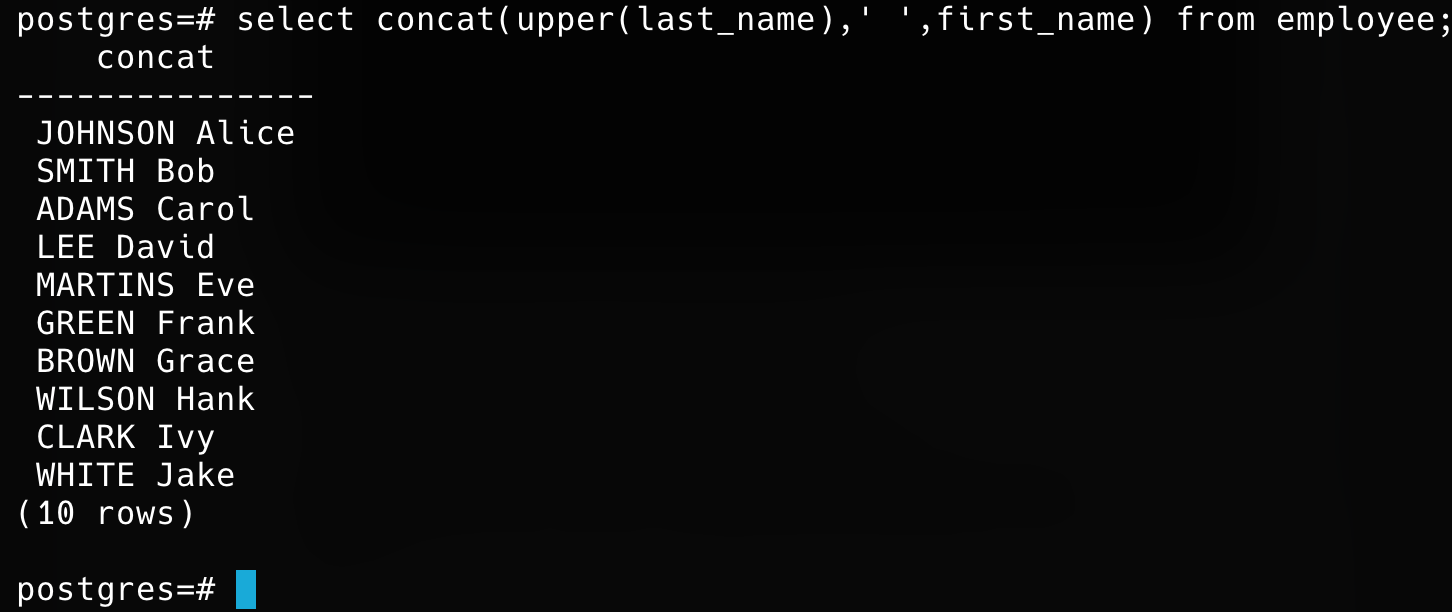
## Extract last name only from email (before @).

select split\_part(split\_part(email, '@', 1), '.', 2) as last\_name from employee;



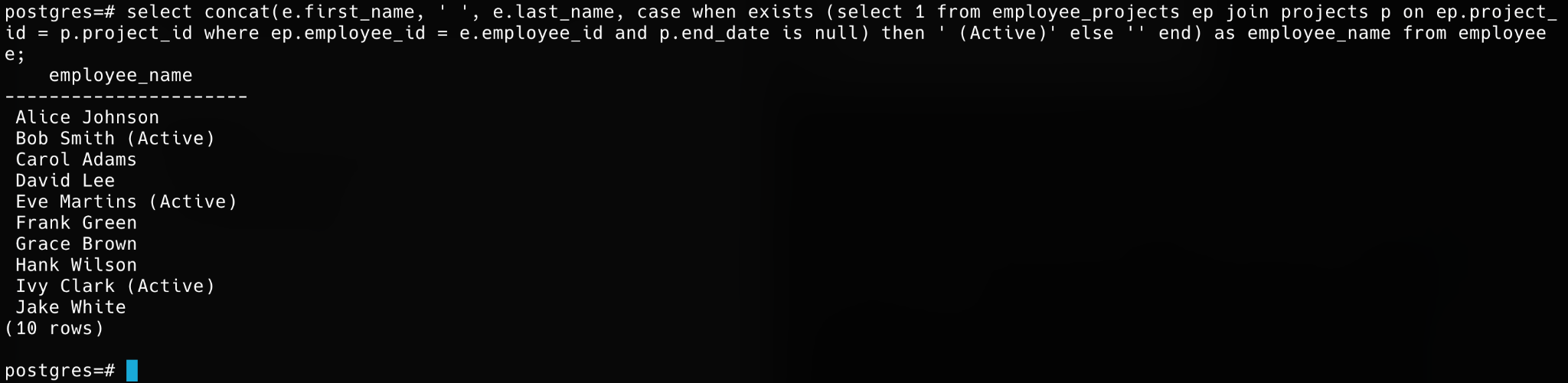
## Format: “LASTNAME, Firstname” using UPPER and CONCAT

select concat(upper(last\_name),' ',first\_name) from employee;



## Add “(Active)” next to employee names who have current projects.

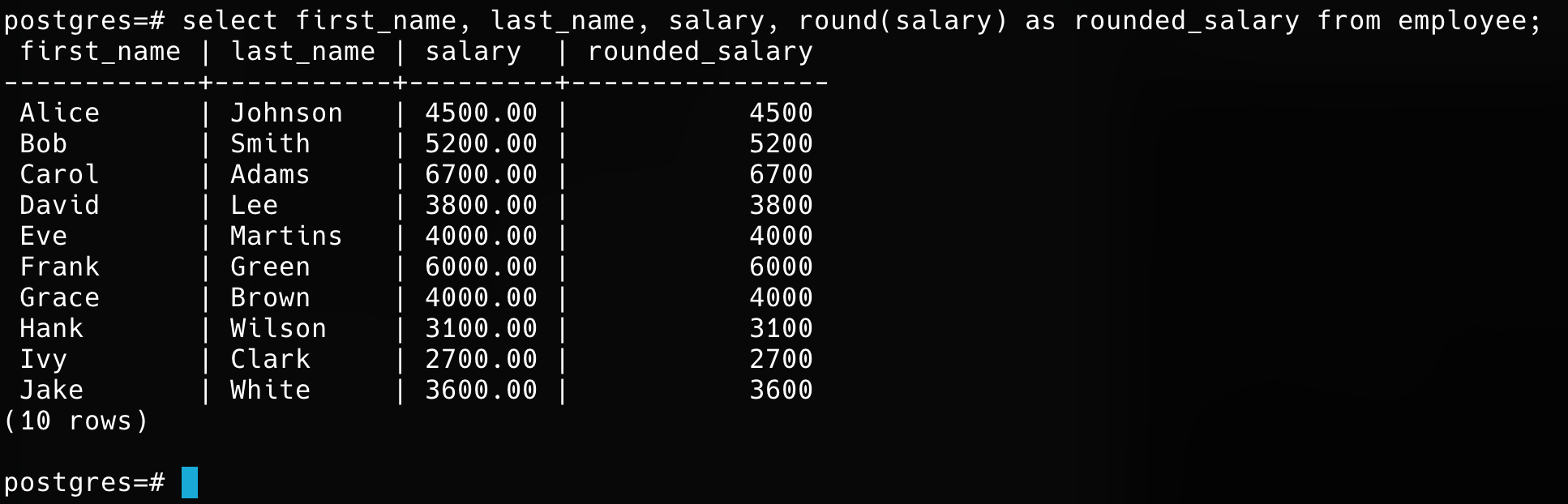
select concat(e.first\_name, ' ', e.last\_name, case when exists (select 1 from employee\_projects ep join projects p on ep.project\_id = p.project\_id where ep.employee\_id = e.employee\_id and p.end\_date is null) then ' (Active)' else '' end) as employee\_name from employee e;



# NUMERIC FUNCTIONS

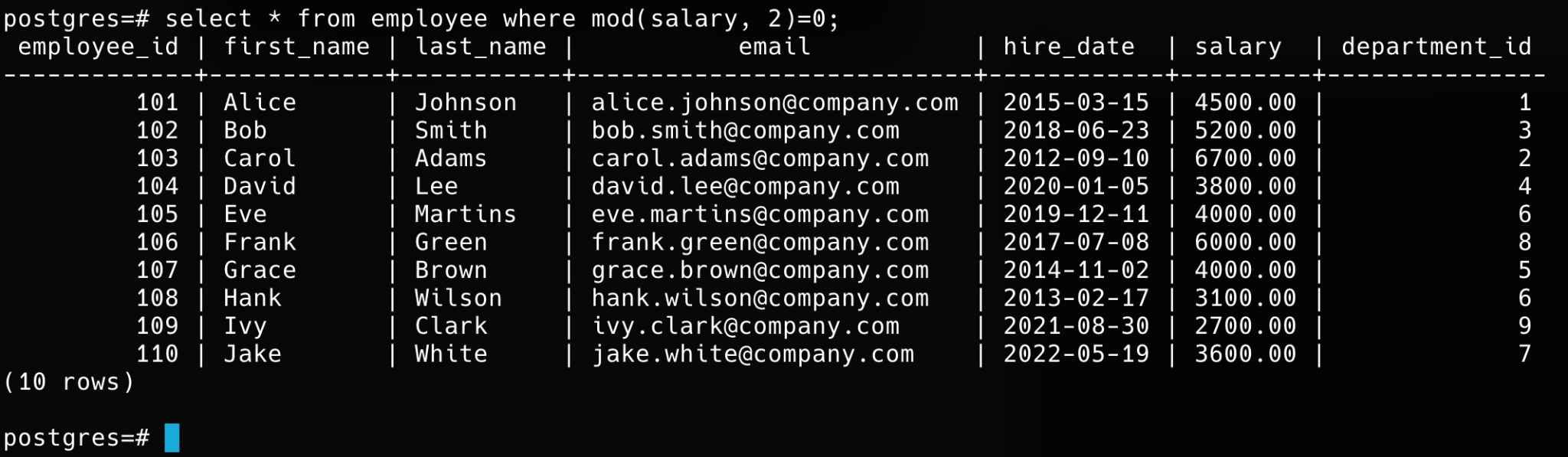
## Round salary to the nearest whole number

select first\_name, last\_name, salary, round(salary) as rounded\_salary from employee;



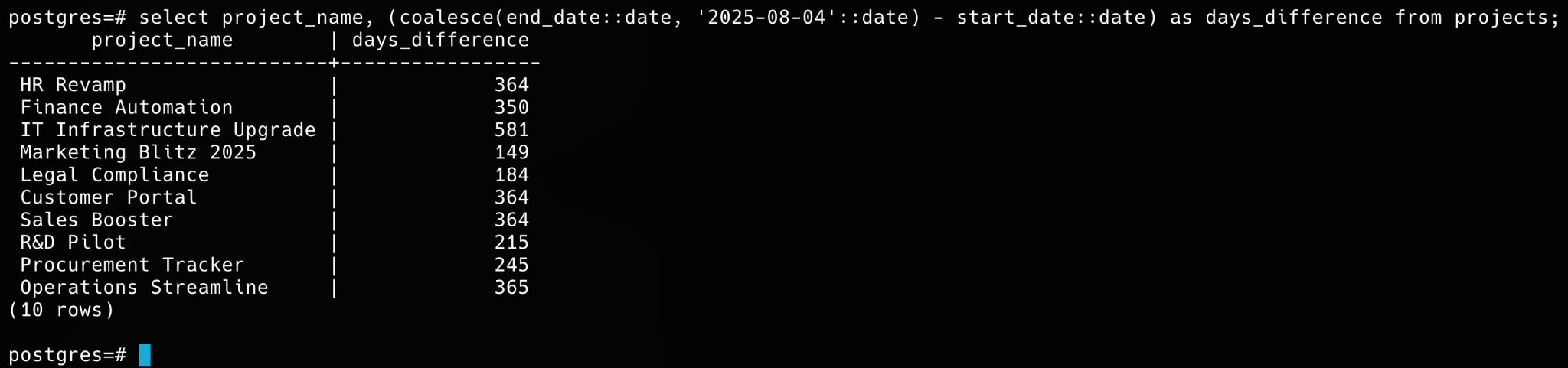
## Show only even salaries using MOD.

select \* from employee where mod(salary, 2)=0;



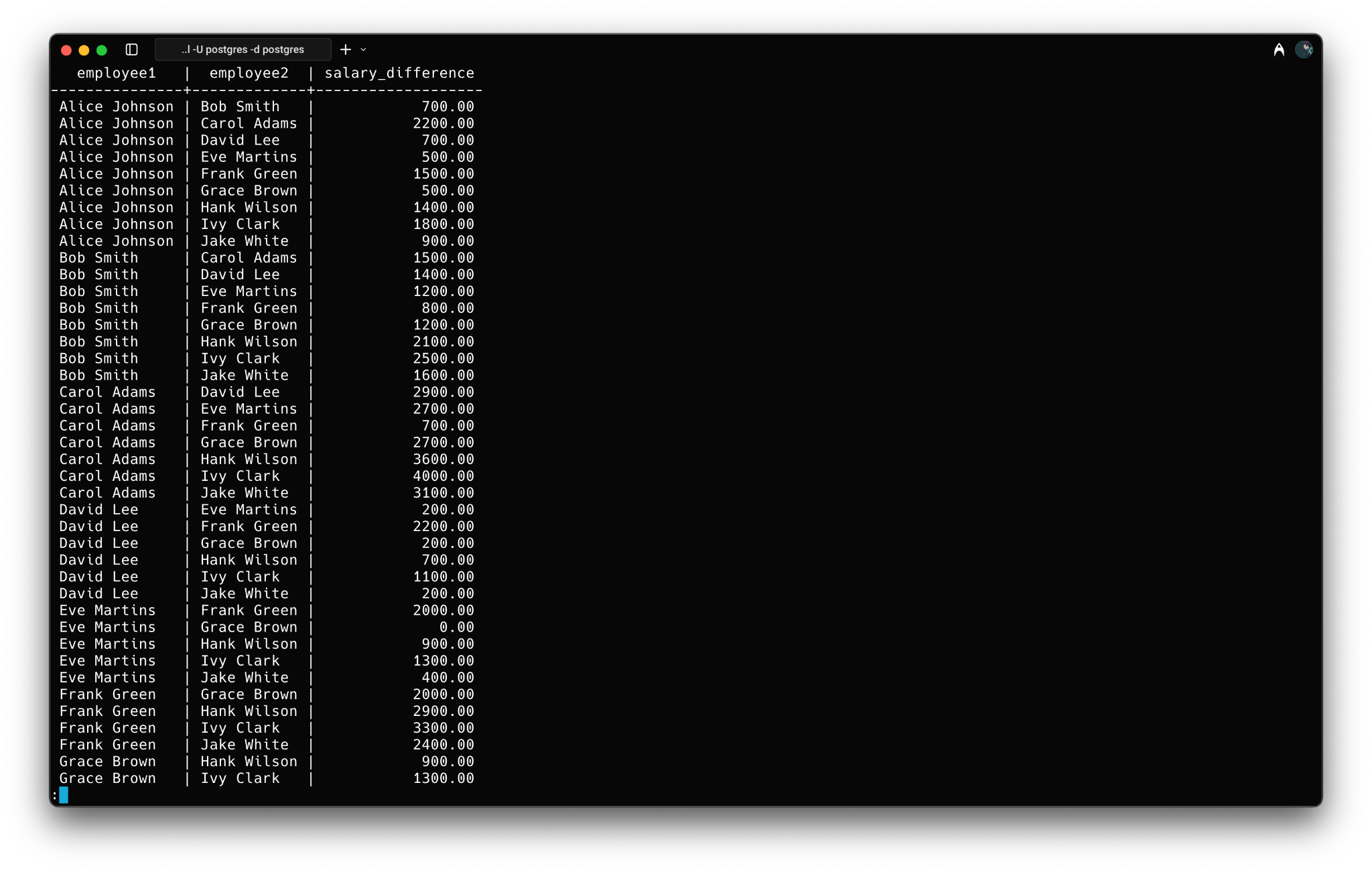
## Show difference between two project end/start dates using DATEDIFF (i.e DATEDIFF not supported in postgress)

select project\_name, (coalesce(end\_date::date, '2025-08-04'::date) - start\_date::date) as days\_difference from projects;



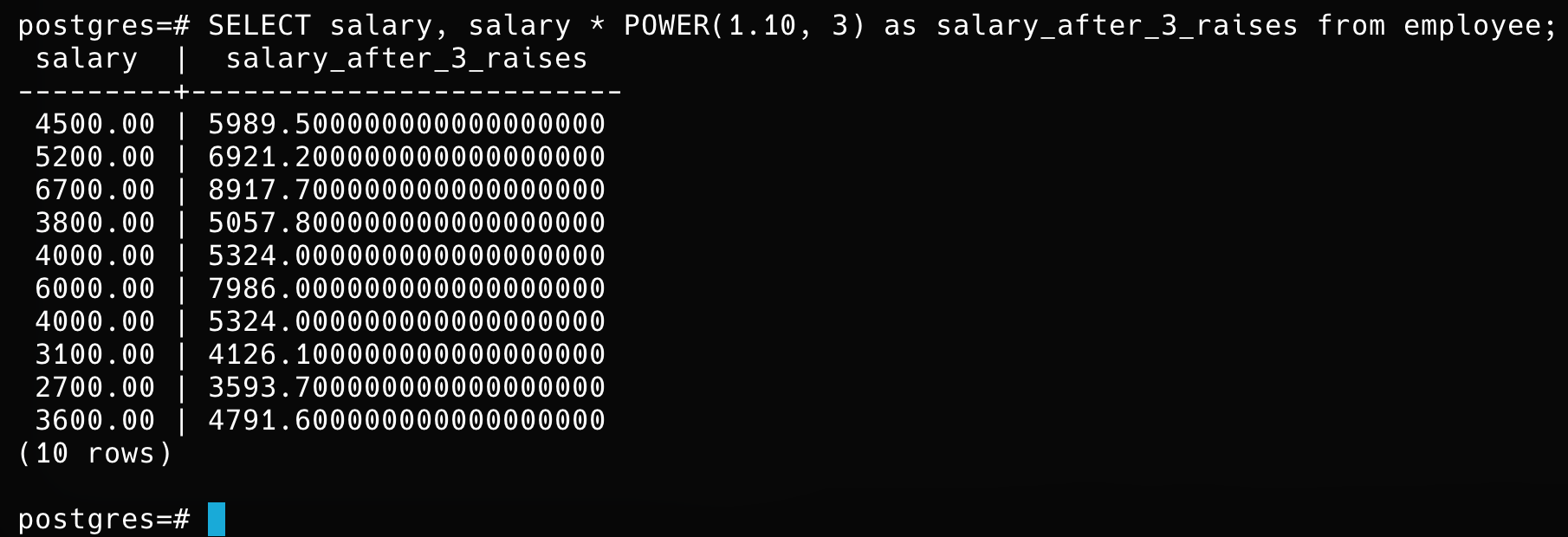
## Show absolute difference in salaries between two employees.

select concat(e1.first\_name, ' ', e1.last\_name) as employee1, concat(e2.first\_name, ' ', e2.last\_name) as employee2, abs(e1.salary - e2.salary) as salary\_difference from employee e1 join employee e2 on e1.employee\_id < e2.employee\_id;



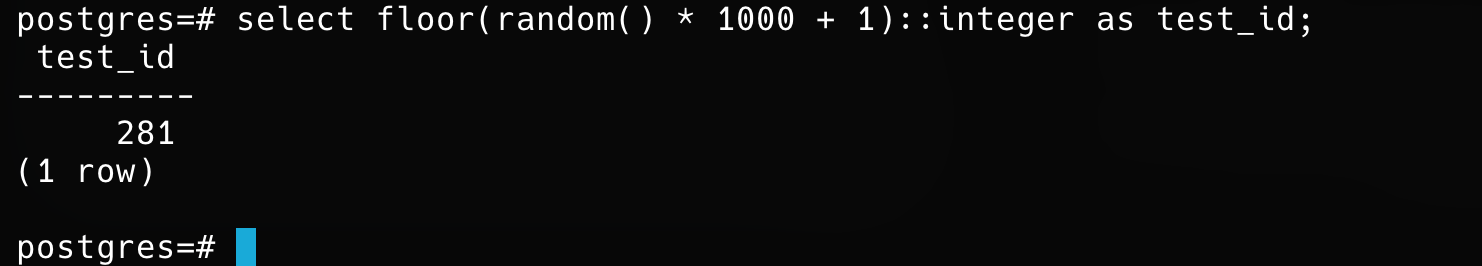
## Raise salary by 10% using POWER.

select salary, salary \* power(1.10, 3) as salary\_after\_3\_raises from employee;



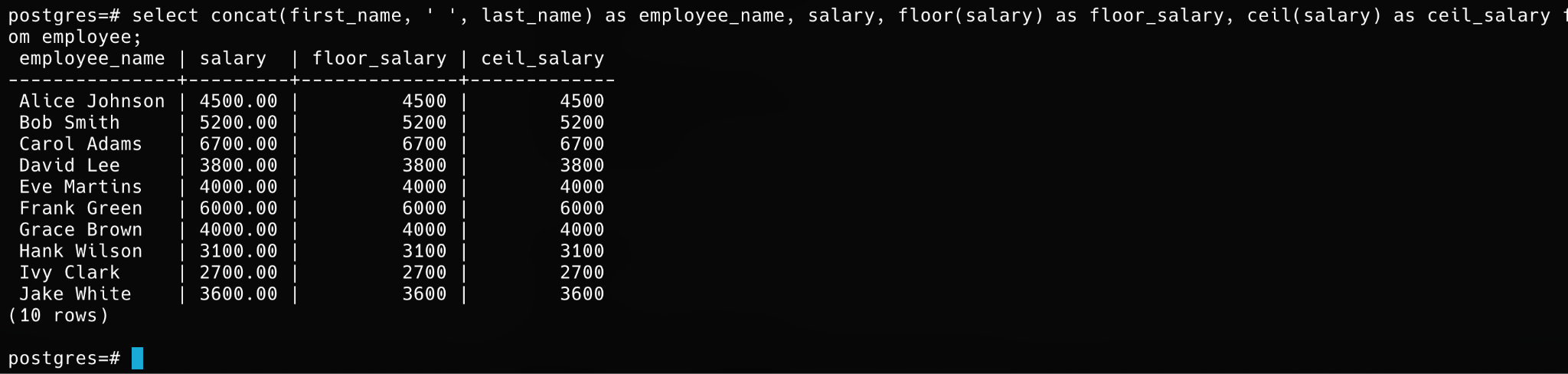
## Generate a random number for testing IDs.

select floor(random() \* 1000 + 1)::integer as test\_id;



## Use CEIL and FLOOR on a floating salary

select concat(first\_name, ' ', last\_name) as employee\_name, salary, floor(salary) as floor\_salary, ceil(salary) as ceil\_salary from employee;

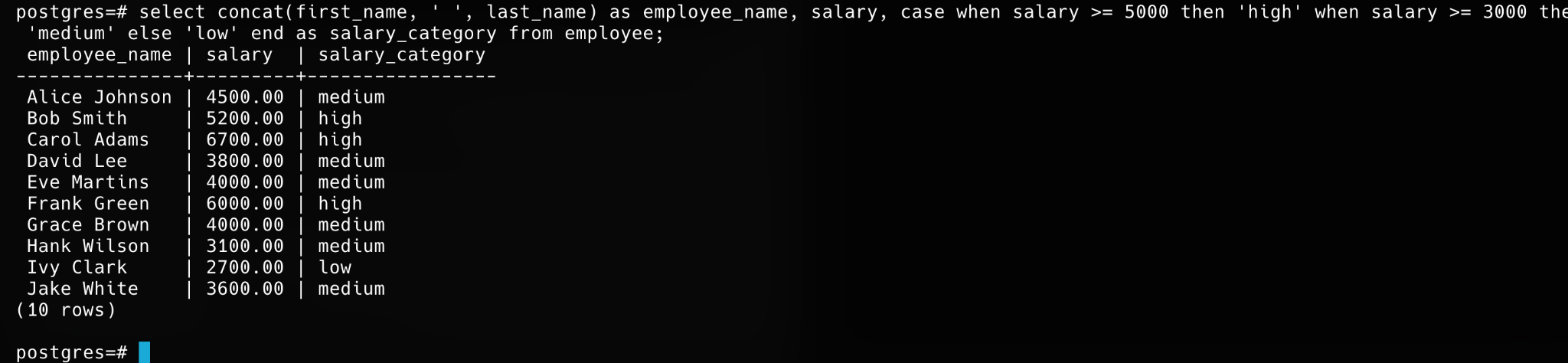


## Use LENGTH() on phone numbers (assume column exists).

select concat(first\_name, ' ', last\_name) as employee\_name, phone\_number, length(phone\_number) as phone\_number\_length from employee;

## Categorize salary: High/Medium/Low using CASE.

select concat(first\_name, ' ', last\_name) as employee\_name, salary, case when salary >= 5000 then 'high' when salary >= 3000 then 'medium' else 'low' end as salary\_category from employee;



## Count digits in salary amount.

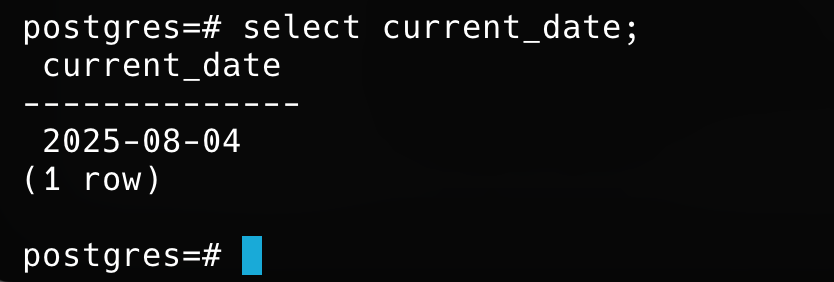
select salary, length(replace(salary::text, '.', '')) as digit\_count from employee;



# DATE/TIME FUNCTIONS

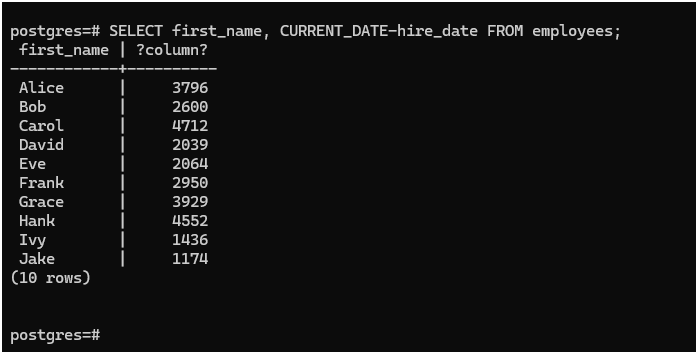
## Show today’s date using CURRENT\_DATE.

select current\_date;



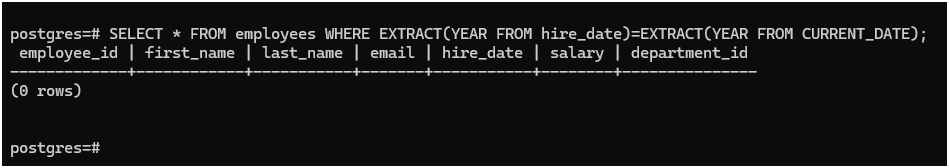
## Calculate how many days an employee has worked.

select first\_name, last\_name, hire\_date, datediff(current\_date, hire\_date) as days\_worked from employee;



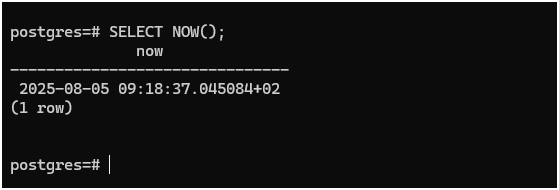
## Show employees hired in the current year.

SELECT first\_name, last\_name, hire\_date FROM employee WHERE YEAR(hire\_date) = YEAR(CURRENT\_DATE);



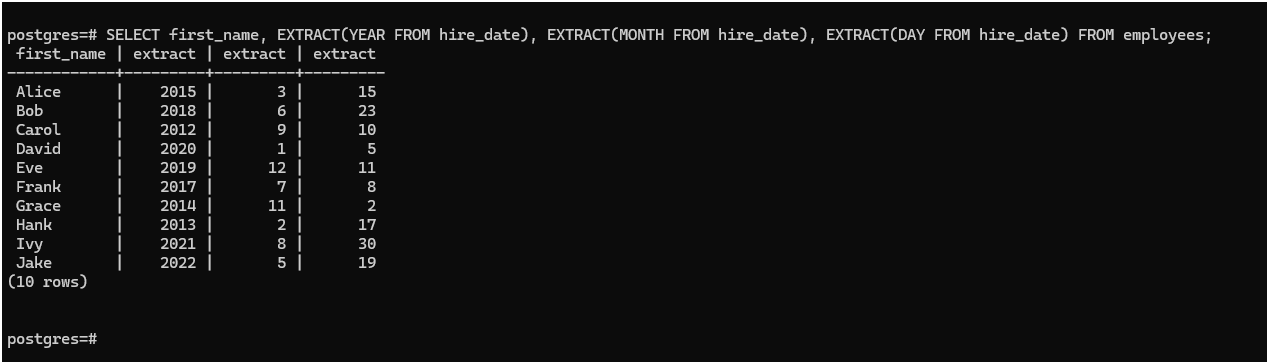
## Display current date and time using NOW().

select now() as current\_datetime;



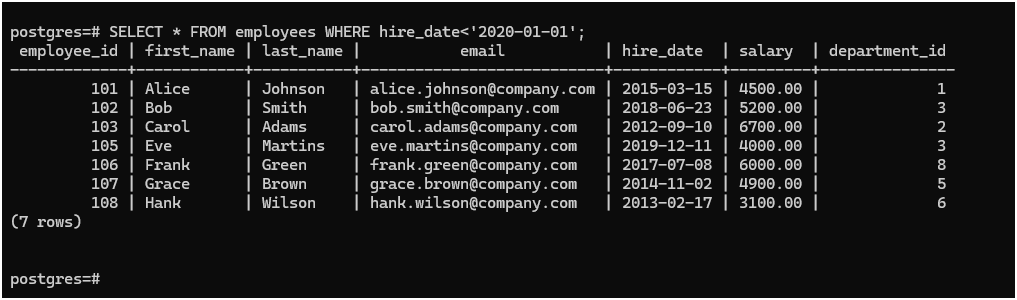
## Extract the year, month, and day from hire\_date.

SELECT first\_name, last\_name, hire\_date, YEAR(hire\_date) AS hire\_year, MONTH(hire\_date) AS hire\_month, DAY(hire\_date) AS hire\_day FROM employee;



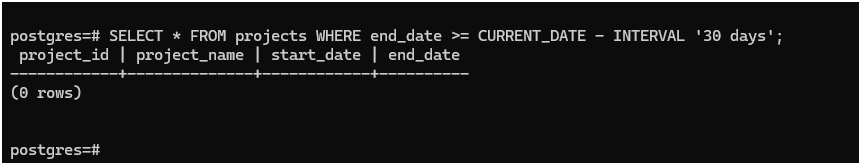
## Show employees hired before 2020.

SELECT first\_name, last\_name, hire\_date FROM employee WHERE hire\_date < '2020-01-01';



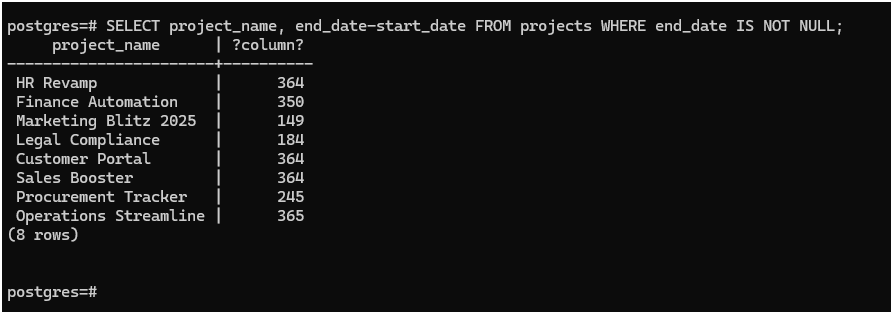
## List projects that ended in the last 30 days.

SELECT project\_name, end\_date FROM projects WHERE end\_date BETWEEN DATE\_SUB(CURRENT\_DATE, INTERVAL 30 DAY) AND CURRENT\_DATE;



## Calculate total days between project start and end dates.

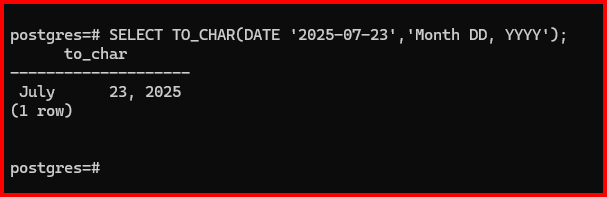
SELECT project\_name, start\_date, end\_date, DATEDIFF(end\_date, start\_date) AS total\_duration\_days FROM projects;



## Format date: ‘2025-07-23’ to ‘July 23, 2025’ (use CONCAT).

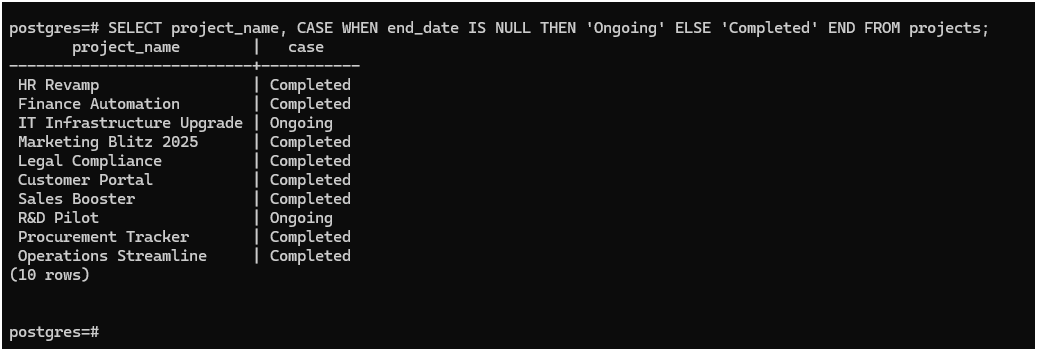
SELECT CONCAT( MONTHNAME('2025-07-23'), ' ', DAY('2025-07-23'), ', ', YEAR('2025-07-23') ) AS

formatted\_date;



## Add a CASE: if project still active (end\_date IS NULL), show ‘Ongoing’.

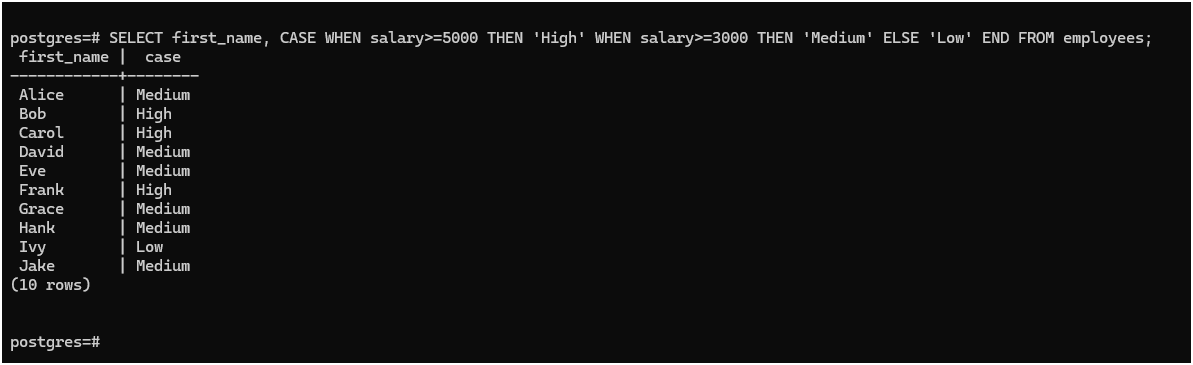
SELECT project\_name, start\_date, CASE WHEN end\_date IS NULL THEN 'Ongoing' ELSE DATE\_FORMAT(end\_date, '%Y-%m-%d') END AS project\_status\_or\_end\_date FROM projects;



# CONDITIONAL FUNCTIONS

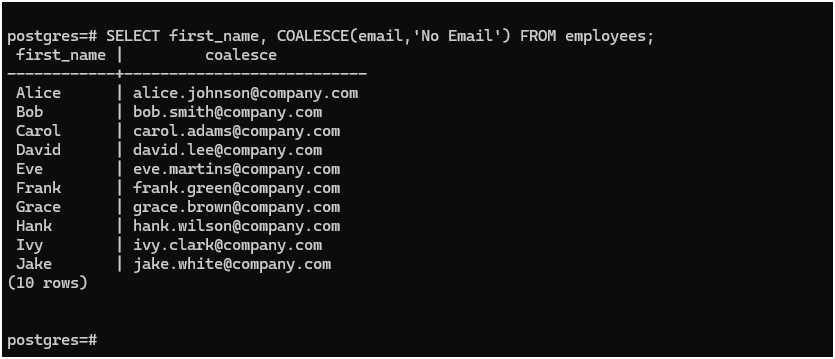
1. **Use CASE to label salaries.**

SELECT first\_name, last\_name, salary, CASE WHEN salary >= 5000 THEN 'High' WHEN salary >= 3500 THEN 'Medium' ELSE 'Low' END AS salary\_label FROM employee;



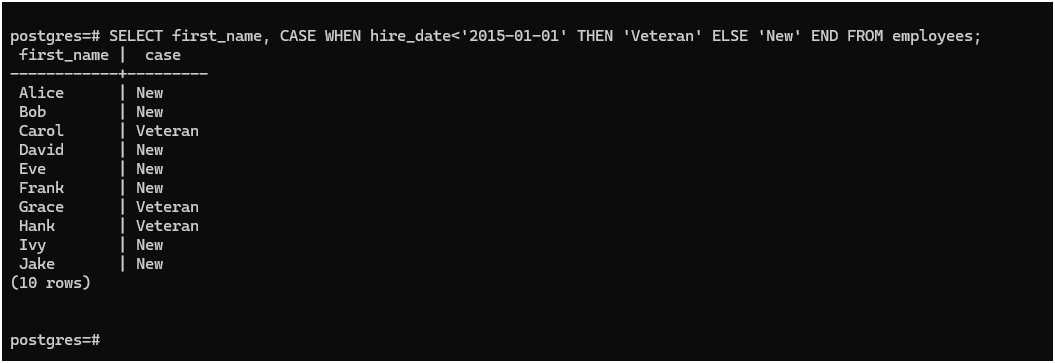
## Use COALESCE to show ‘No Email’ if email is NULL.

SELECT first\_name, last\_name, COALESCE(email, 'No Email') AS email\_address FROM employee;



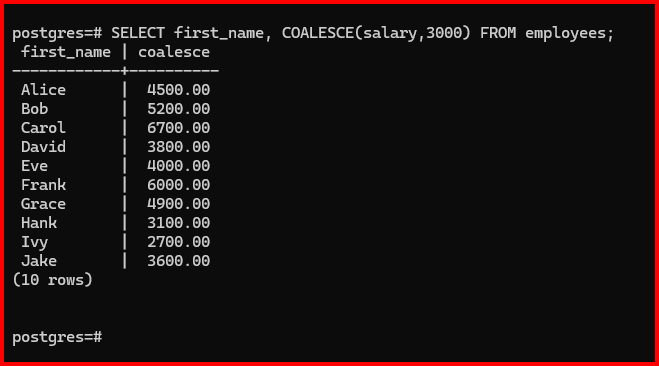
## CASE: If hire\_date < 2015, mark as ‘Veteran’

SELECT first\_name, last\_name, hire\_date, CASE WHEN hire\_date < '2015-01-01' THEN 'Veteran' ELSE 'Newcomer' END AS employee\_status FROM employee;



## If salary is NULL, default it to 3000 using COALESCE

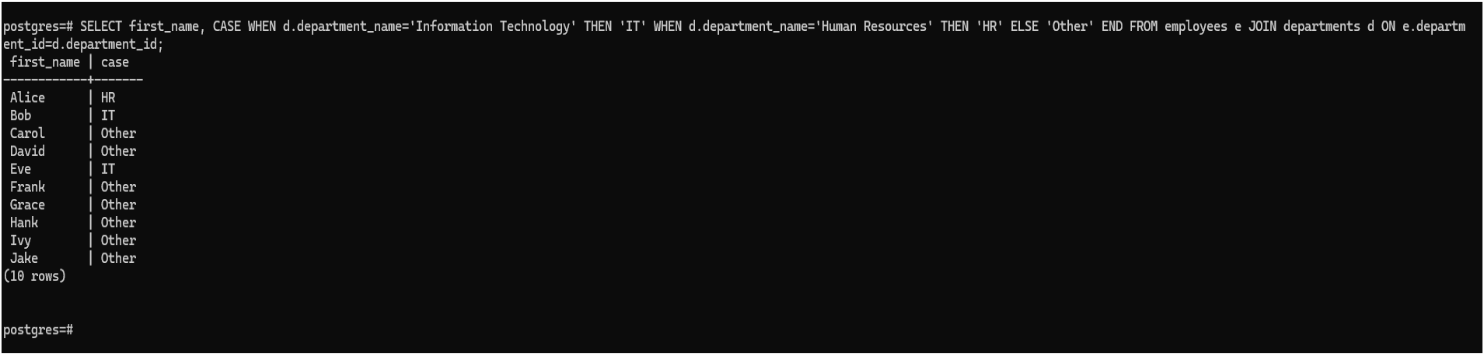
SELECT first\_name, last\_name, COALESCE(salary, 3000) AS salary\_with\_default FROM employee;



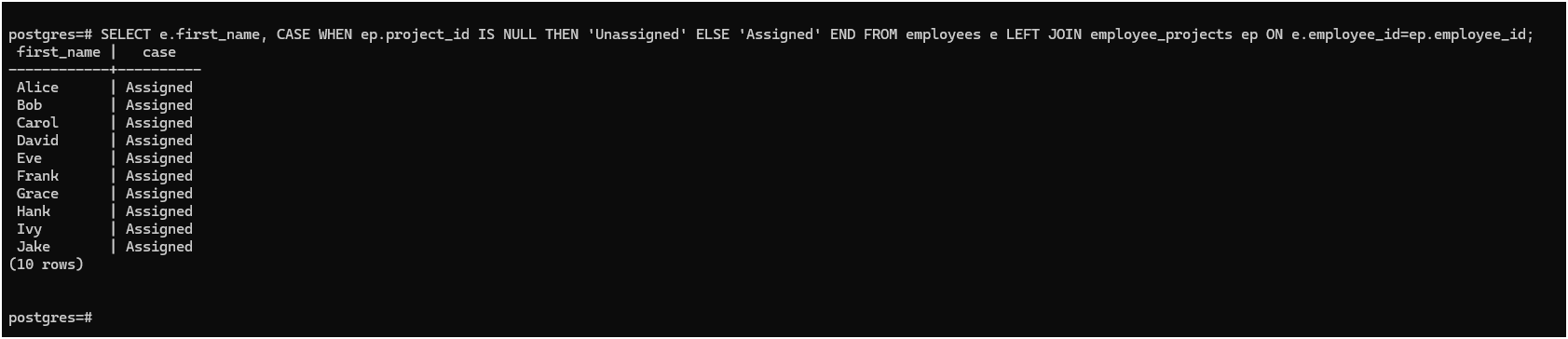
## CASE: Categorize departments (IT, HR, Other).

SELECT first\_name, last\_name, department, CASE WHEN department = 'IT' THEN 'IT' WHEN department

= 'HR' THEN 'HR' ELSE 'Other' END AS department\_category FROM employee;

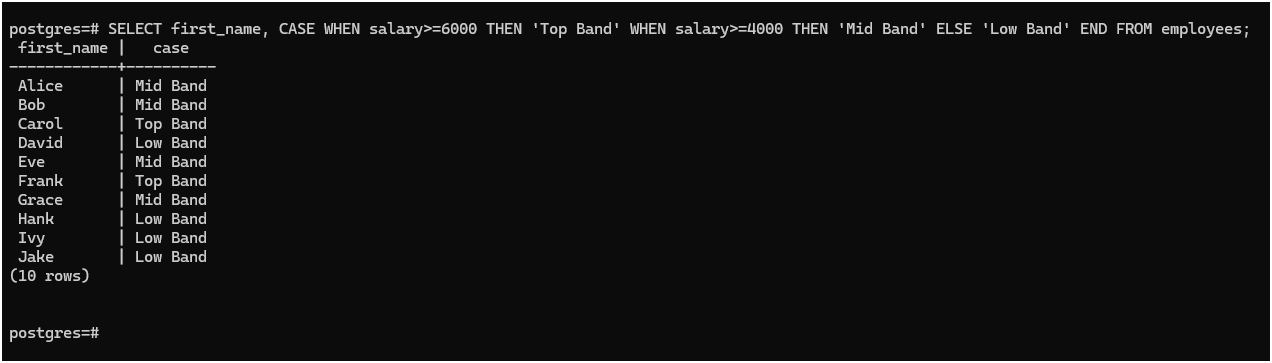


**41. CASE: If employee has no project, mark as ‘Unassigned’.**

****

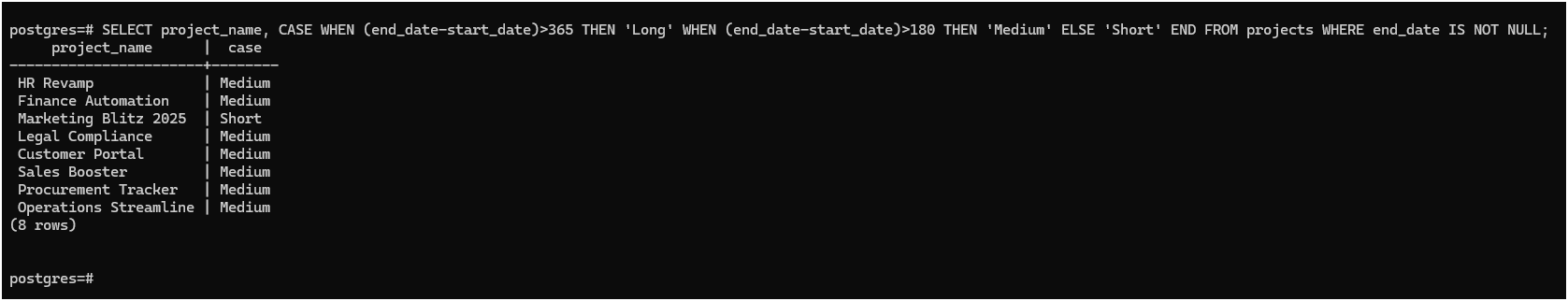
## CASE: Show tax band based on salary

SELECT first\_name, last\_name, salary, CASE WHEN salary >= 5000 THEN 'High Tax Band' WHEN salary >= 3500 THEN 'Mid Tax Band' WHEN salary >= 1000 THEN 'Low Tax Band' ELSE 'No Tax' END AS tax\_band FROM employee;



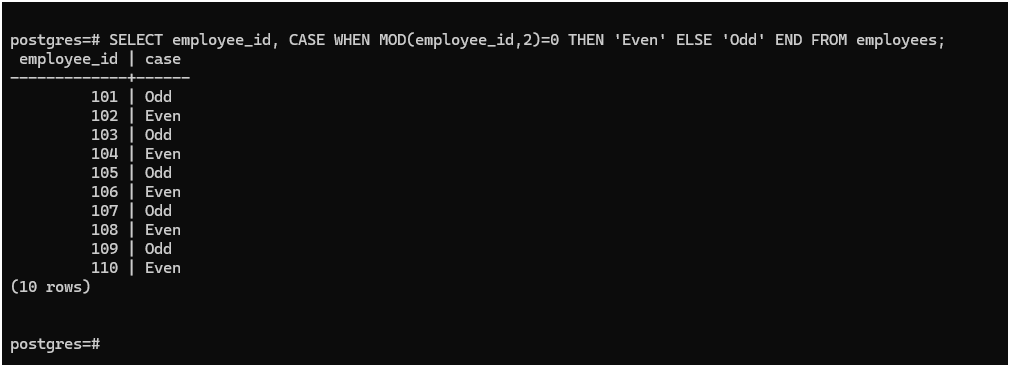
## Use nested CASE to label project duration

SELECT project\_id, project\_name, start\_date, end\_date, DATEDIFF(end\_date, start\_date) AS duration\_days, CASE WHEN DATEDIFF(end\_date, start\_date) < 350 THEN 'Short' WHEN DATEDIFF(end\_date, start\_date) BETWEEN 350 AND 480 THEN CASE WHEN DATEDIFF(end\_date, start\_date) <= 90 THEN 'Medium' ELSE 'Moderately Long' END ELSE 'Long' END AS duration\_label FROM projects;



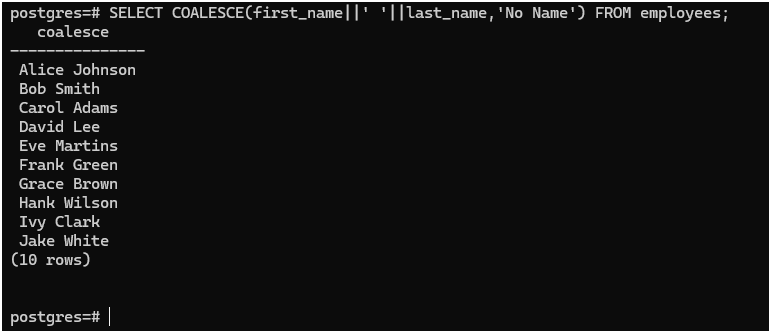
## Use CASE with MOD to show even/odd salary IDs.

SELECT employee\_id, first\_name, salary, CASE WHEN MOD(employee\_id, 2) = 0 THEN 'Even' ELSE 'Odd' END AS id\_parity FROM employee;



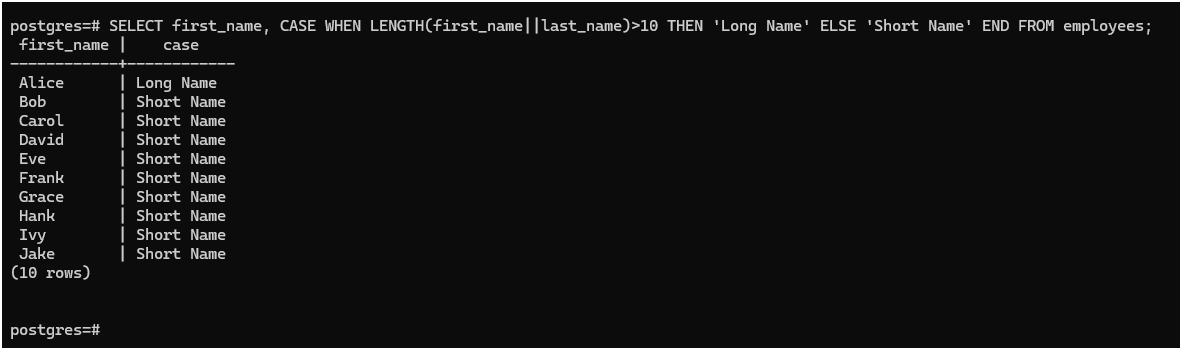
## Combine COALESCE + CONCAT for fallback names.

SELECT employee\_id, CONCAT( COALESCE(first\_name, 'Unknown'), ' ', COALESCE(last\_name, 'Unknown') ) AS full\_name FROM employee;



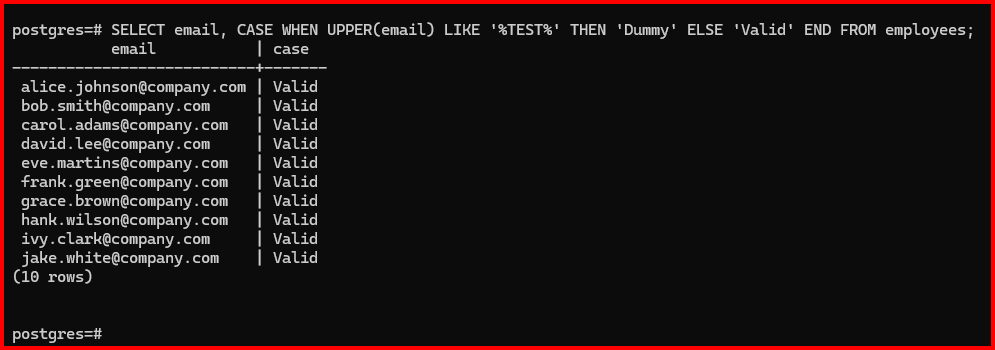
## CASE with LENGTH(): if name length > 10, label “Long Name”.

SELECT first\_name, LENGTH(first\_name) AS name\_length, CASE WHEN LENGTH(first\_name) > 10 THEN 'Long Name' ELSE 'Short Name' END AS name\_label FROM employee;



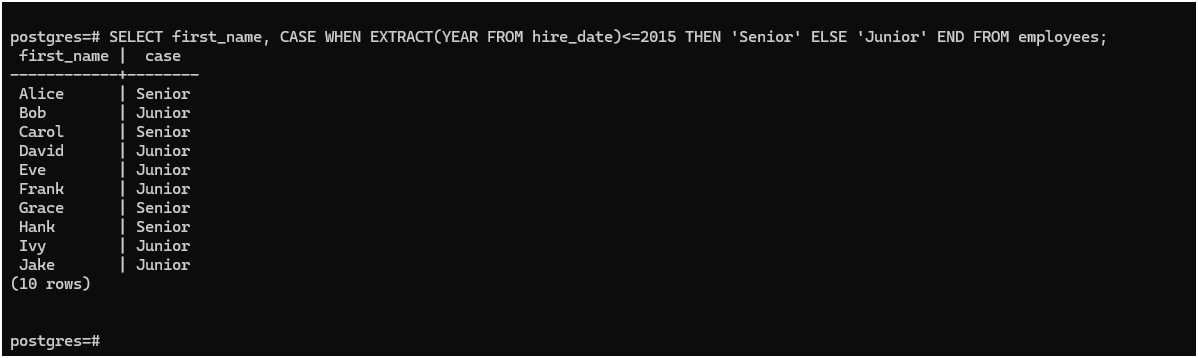
## CASE + UPPER(): if email has ‘TEST’, mark as dummy account.

SELECT email, CASE WHEN UPPER(email) LIKE '%TEST%' THEN 'Dummy Account' ELSE 'Real Account' END AS account\_type FROM employee;



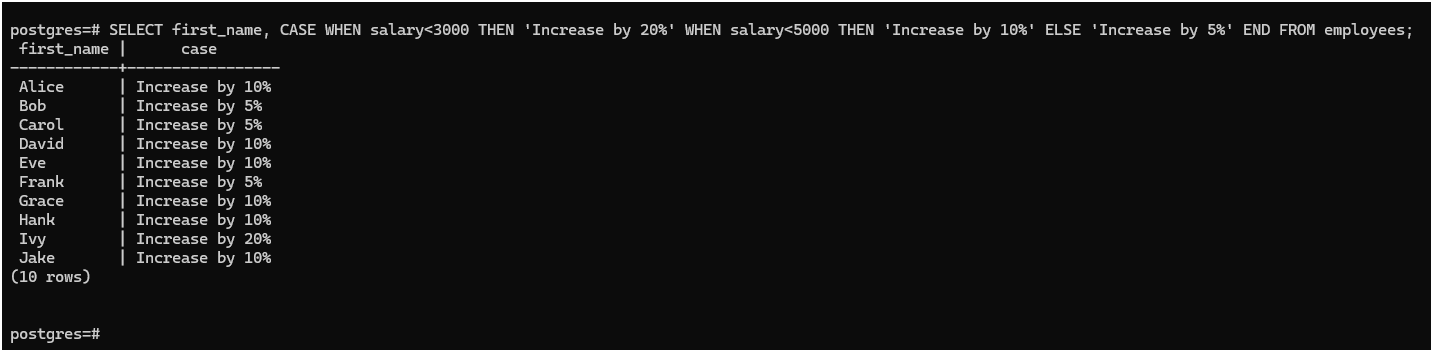
## CASE: Show seniority based on hire year (e.g., Junior/Senior)

SELECT first\_name, hire\_date, CASE WHEN YEAR(hire\_date) <= YEAR(CURDATE()) - 10 THEN 'Senior' WHEN YEAR(hire\_date) <= YEAR(CURDATE()) - 5 THEN 'Mid-Level' ELSE 'Junior' END AS seniority\_level FROM employee;



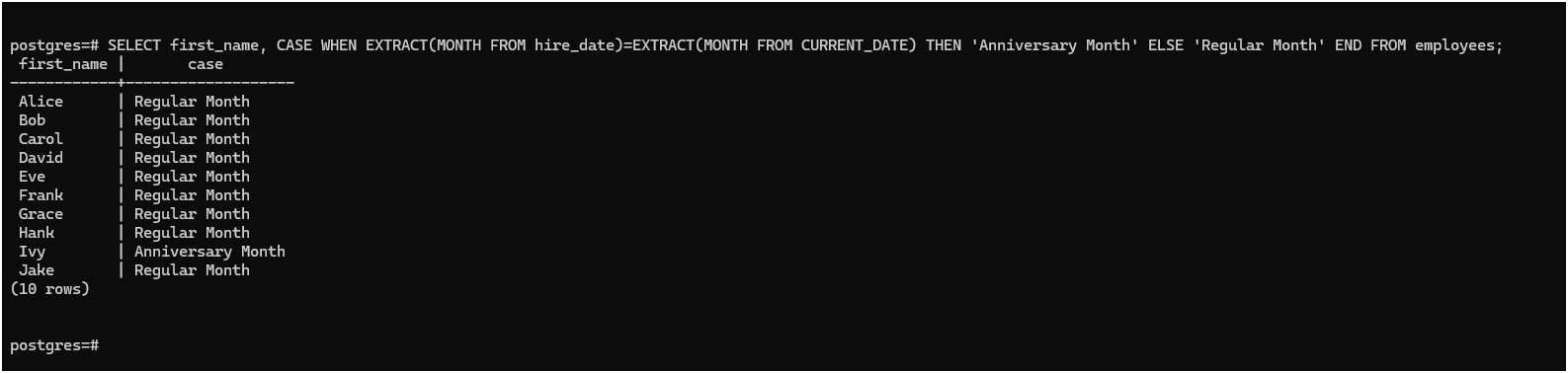
## Use CASE to determine salary increment range.

SELECT employee\_id, first\_name, salary, CASE WHEN salary >= 5000 THEN 'Increase by 5%' WHEN salary >= 3500 THEN 'Increase by 10%' WHEN salary >= 2000 THEN 'Increase by 15%' ELSE 'Increase by 20%' END AS increment\_range FROM employee;



## Use CASE with CURDATE() to determine anniversary month.

SELECT employee\_id, first\_name, hire\_date, CASE WHEN MONTH(hire\_date) = MONTH(CURDATE()) THEN 'Anniversary Month' ELSE 'Not Anniversary Month' END AS anniversary\_status FROM employee;



**DBMS ASSIGNMENT1.**