**10 consultas SQL que debe conocer como científico de datos**

Aprenda las consultas SQL más utilizadas en 5 minutos con ejemplos



**Introducción**

El lenguaje de consulta estructurado o SQL es un lenguaje simple basado en consultas que es fácil de leer y escribir. Es uno de los idiomas más universales del mundo.

En el mundo de hoy, todos sabemos lo importantes que son los datos. Veremos las consultas SQL más comunes con ejemplos.

El objetivo es mostrarle consultas básicas y avanzadas que se utilizan en la ciencia de datos.

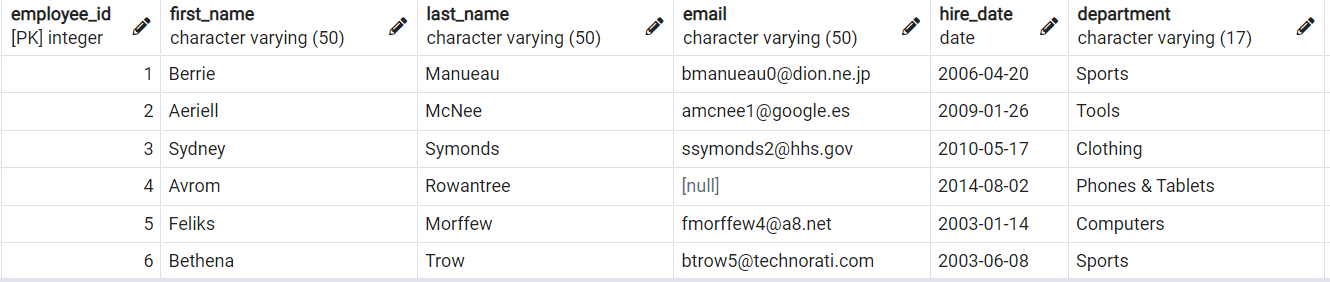
*Nota: Todas las consultas están escritas en****PostgreSQL****.*

***¡Comencemos!***

**1. Seleccione Todas las columnas**

Si desea seleccionar todos los campos disponibles en la tabla, utilice la siguiente sintaxis: \* nos ayuda a elegir todas las columnas de la tabla.

SELECT **\*** FROM employees

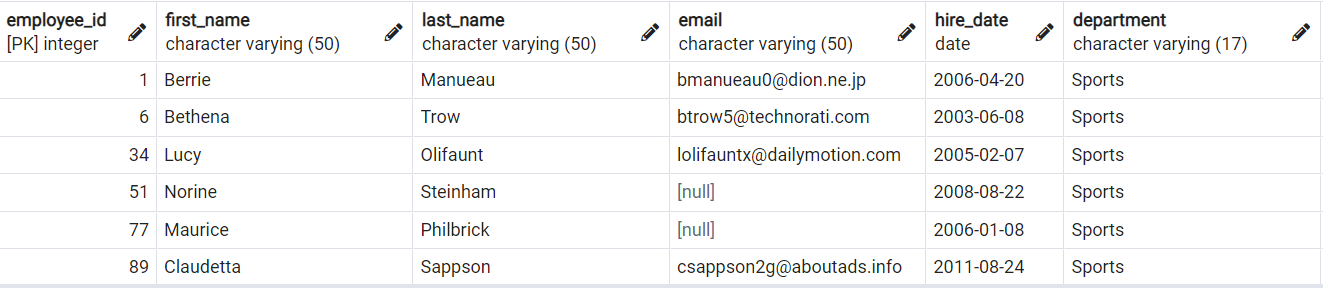


**2. Donde Declaración**

Podemos usar la cláusula **WHERE**, que filtra los datos en función de la declaración dada.

**Ejemplo:** Escriba una consulta para imprimir todas las variables para**el Departamento de Deportes**.

SELECT \* FROM employees  
**WHERE department = 'Sports'**

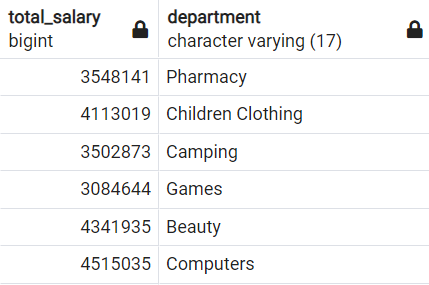


**3. Agrupar por y tener cláusula**

La cláusula **Agrupar por** agrupa filas que tienen los mismos valores.

**Ejemplo:** ¿Cuál es el salario total de cada departamento?

SELECT SUM(salary) as total\_salary,department   
FROM employees  
**GROUP BY** department

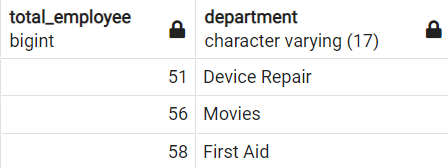


Agrupar por ejemplo

En SQL, las funciones de agregación como **SUM, AVG, MAX, MIN**y**COUNT** no se pueden usar en la cláusula **WHERE**. Si queremos filtrar nuestra tabla por una función de agregación, necesitamos usar la cláusula **HAVING**.

**Ejemplo**: ¿Qué departamentos tienen más de 50 empleados?

SELECT count(\*) as total\_employee,department   
FROM employees  
GROUP BY department  
**HAVING COUNT(\*) > 50**

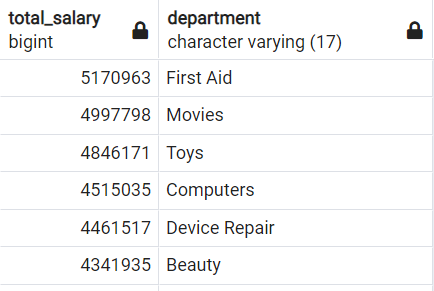


**4. Ordene por y limite**

**Ejemplo:** Encuentre el salario total por cada departamento y clasifique **descendentemente por**columna de salario total.

*Tenga en cuenta que el alias se puede usar en orden por.*

SELECT SUM(salary) as total\_salary,department FROM employees  
GROUP BY department  
**ORDER BY** total\_salary desc



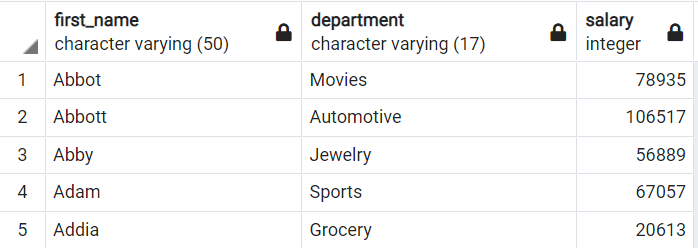
Orden por cláusula

El comando Limitar se utiliza para especificar el número de registros que se van a devolver.

**Ejemplo**: Escriba una consulta que encuentre a los primeros 5 empleados con sus first\_name, departamento y salario y ordenados por su first\_name.

*Nota: La cláusula Orden por defecto ordena el resultado en orden ASCENDENTE.*

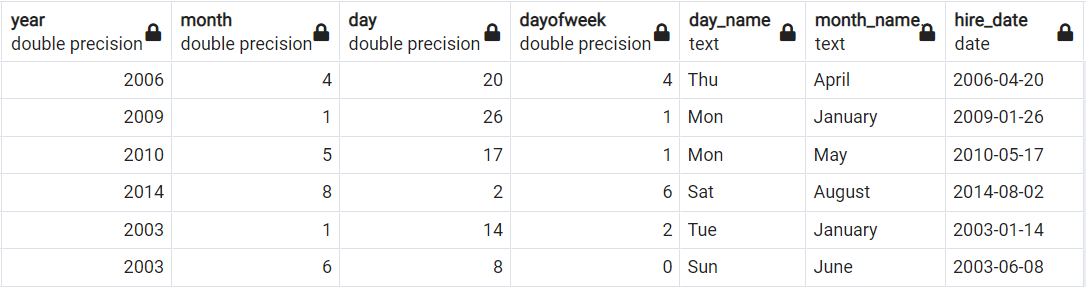
SELECT first\_name,department,salary from employees  
ORDER BY first\_name  
**LIMIT 5**



**5. Funciones de fecha**

En PostgreSQL, puede extraer fácilmente valores de columnas de fecha. Verá las funciones de fecha más utilizadas a continuación.

SELECCIONE   
date\_part('año',hire\_date) como año,date\_part  
('mes',hire\_date) como mes,date\_part  
('día',hire\_date) como día,date\_part  
('dow',hire\_date) como díadesemana,to\_char  
(hire\_date, 'Dy') como day\_name,to\_char  
(hire\_date,'Mes') como month\_name,hire\_date  
  
 DE empleados



Salida de funciones de fecha

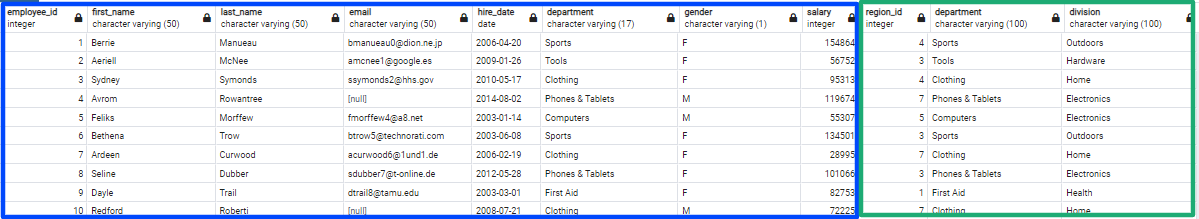
**6. Unirse hacia adentro, izquierda o derecha**

**La cláusula Inner Join** crea una nueva tabla (no física) combinando filas que tienen valores coincidentes en dos o más tablas.

**Ejemplo:** Consultar toda la información de los empleados y sus divisiones del departamento.

*Nota: La tabla azul es la primera tabla y la tabla verde es nuestra segunda tabla.*

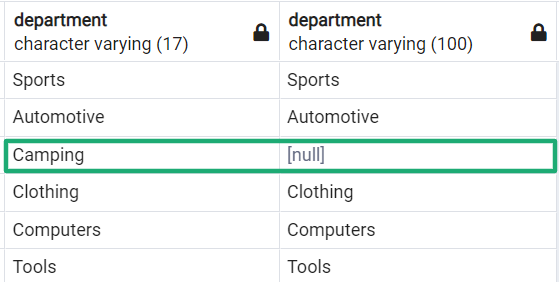
SELECT \* FROM employees e  
**INNER JOIN departments d  
ON e.department = d.department**



La **unión izquierda**devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no se encuentran filas coincidentes en la tabla derecha, se utiliza **NULL**. (viceversa para Right Join)

**Ejemplo:** Escriba una consulta que imprima todos los departamentos de los empleados y haga coincidir los departamentos de la tabla de departamentos.

SELECT e.department,d.department FROM employees e  
LEFT JOIN departments d  
ON e.department = d.department



**7. Subconsultas**

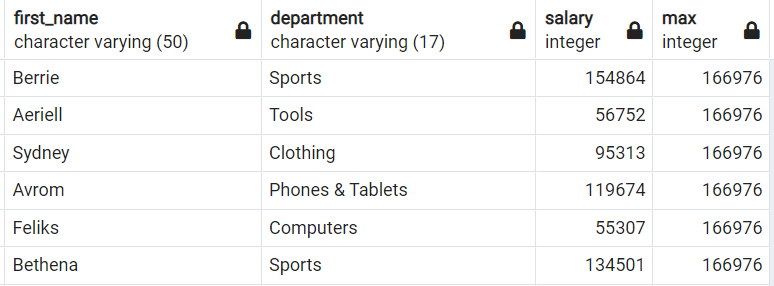
Una subconsulta es una consulta SQL anidada dentro de una consulta más grande.

Una subconsulta puede ocurrir en:

* una cláusula SELECT
* a Cláusula FROM
* a Cláusula WHERE

**Ejemplo:**Consulta first\_name, departamento y salario de cada empleado y también salario máximo dado.

SELECT first\_name,department,salary,(SELECT max(salary) FROM employees)  
FROM employees



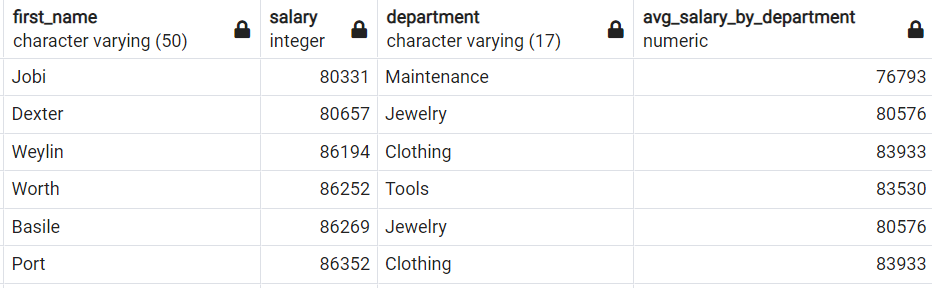
QUE

**8. Subconsultas correlacionadas**

Una subconsulta correlacionada es una forma de leer cada fila de una tabla y comparar los valores de cada fila con los datos relacionados. Se utiliza siempre que una subconsulta debe devolver un resultado o conjunto de resultados diferente para cada fila candidata considerada por la consulta principal.

**Ejemplo:** Escriba una consulta que encuentre el nombre, el salario, el departamento y el salario promedio por departamento.

SELECT first\_name,salary,department,round((SELECT AVG(salary)   
 FROM employees e2  
 WHERE e1.department = e2.department  
 GROUP BY department )) as avg\_salary\_by\_department  
FROM employees e1   
WHERE salary > (SELECT AVG(salary)   
 FROM employees e2  
 WHERE e1.department = e2.department  
 GROUP BY department )  
ORDER BY salary



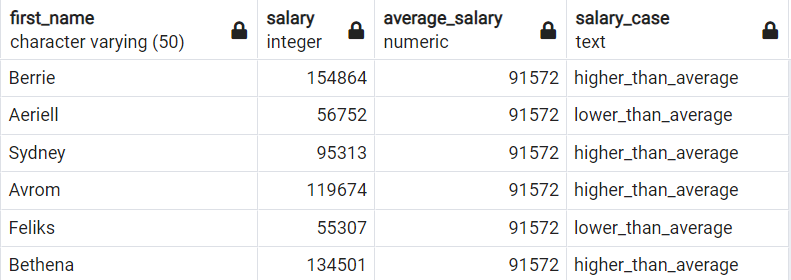
**9. Cláusula Case When**

La instrucción CASE se utiliza para implementar la lógica en la que desea establecer el valor de una columna en función de los valores de otras columnas.

Es similar a la instrucción IF-ELSE en Excel.

**Ejemplo:** Escriba una consulta para imprimir el nombre, el salario y el salario promedio, así como una nueva columna que muestre si el salario de los empleados es más alto que el promedio o no.

SELECT first\_name,salary,(SELECT ROUND(AVG(salary)) FROM employees) as average\_salary,  
(CASE WHEN salary > (SELECT AVG(salary) FROM employees) THEN 'higher\_than\_average'  
ELSE 'lower\_than\_average' END) as Salary\_Case  
FROM employees



**10. Funciones de la ventana**

Las funciones de ventana aplican funciones agregadas y de clasificación sobre una ventana en particular (conjunto de filas). La cláusula OVER se utiliza con funciones de ventana para definir esa ventana. La cláusula OVER hace dos cosas:

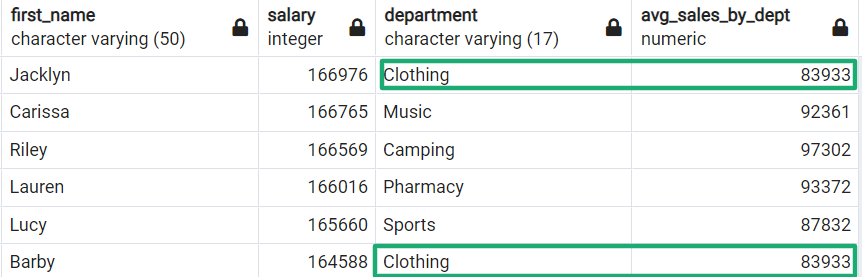
* Particiones filas para formar un conjunto de filas (se utiliza la cláusula PARTITION BY).
* Ordena filas dentro de esas particiones en un orden particular (se utiliza la cláusula ORDER BY).

Varias funciones agregadas como SUM(), COUNT(), AVERAGE(), MAX() y MIN() aplicadas sobre una ventana en particular (conjunto de filas) se denominan **funciones de ventana agregada.**

**10.1. Ejemplos de agregación**

La siguiente consulta le dará el salario promedio de cada departamento.

SELECT first\_name,salary,department,  
ROUND(AVG(salary) OVER(PARTITION BY department)) as avg\_sales\_by\_dept  
FROM employees  
ORDER BY salary DESC



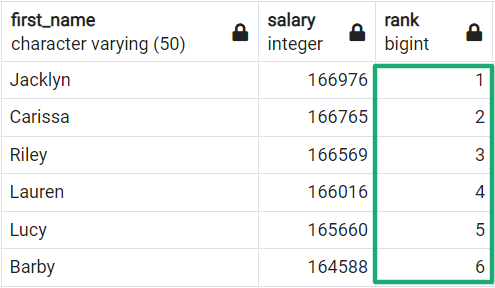
Funciones de ventana agregadas

**10.2.Clasificación de los valores**

La función Rank() es una función de ventana que asigna un rango a cada fila dentro de una partición de un conjunto de resultados.

En el ejemplo siguiente se ordena la tabla por el salario (descendente). Un valor de rango de 1 es el valor salarial más alto.

SELECT first\_name,salary,RANK() OVER(ORDER BY salary DESC)  
FROM employees



Estas son las consultas que utilizan comúnmente los profesionales de datos.