

# Desafio Numero 3 Modulo SQL

## Creación Tabla Usuarios

### Código

```
CREATE TABLE usuarios (  
    id SERIAL PRIMARY KEY,  
    rol VARCHAR,  
    email VARCHAR,  
    nombre VARCHAR,  
    apellido VARCHAR  
);
```

### Salida

Estadísticas 1 ✕	
Name	Value
Updated Rows	0
Query	CREATE TABLE usuarios ( id SERIAL PRIMARY KEY, rol VARCHAR, email VARCHAR, nombre VARCHAR, apellido VARCHAR )
Start time	Thu May 23 16:13:53 CLT 2024
Finish time	Thu May 23 16:13:53 CLT 2024

### Ingreso 5 usuarios con roles

## Ingreso 3 usuarios con roles

### Código

```
INSERT INTO usuarios (rol, email, nombre, apellido)
VALUES
    ('administrador', 'juanadmin@gracar.com', 'Juan', 'grau'),
    ('usuario', 'raul.gonzalez@gracar.com', 'raul', 'gonzalez'),
    ('usuario', 'ramon.ramirez@gracar.com', 'ramon', 'ramirez'),
    ('usuario', 'carlo.uentes@gracar.com', 'carlos', 'fuentes'),
    ('usuario', 'eduardo.montenegro@gracar.com', 'eduardo', 'Montenegro');
```

## Creación Tabla Posts

```
CREATE TABLE posts (
    id SERIAL PRIMARY KEY,
    titulo VARCHAR,
    contenido TEXT,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    destacado BOOLEAN,
    usuario_id BIGINT
);
```

### Salida

Name	Value
Updated Rows	0
Query	CREATE TABLE posts ( id SERIAL PRIMARY KEY, titulo VARCHAR, contenido TEXT, fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP, fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

```
destacado BOOLEAN,  
usuario_id BIGINT  
)  
Start time    Thu May 23 16:29:54 CLT 2024  
Finish time   Thu May 23 16:29:54 CLT 2024
```

## Ingreso Posts

```
INSERT INTO posts (id, titulo, contenido, fecha_creacion,  
fecha_actualizacion, destacado, usuario_id)
```

```
VALUES
```

```
(1, 'Título del post 1', 'Contenido del post 1', CURRENT_TIMESTAMP,  
CURRENT_TIMESTAMP, true, 1),
```

```
(2, 'Título del post 2', 'Contenido del post 2', CURRENT_TIMESTAMP,  
CURRENT_TIMESTAMP, true, 1),
```

```
(3, 'Título del post 3', 'Contenido del post 3', CURRENT_TIMESTAMP,  
CURRENT_TIMESTAMP, false, 2),
```

```
(4, 'Título del post 4', 'Contenido del post 4', CURRENT_TIMESTAMP,  
CURRENT_TIMESTAMP, false, 2),
```

```
(5, 'Título del post 5', 'Contenido del post 5', CURRENT_TIMESTAMP,  
CURRENT_TIMESTAMP, false, NULL);
```

```
-- luego asigno usuarios con update
```

```
UPDATE posts SET usuario_id = 1 WHERE id IN (1, 2); -- Asignar al  
usuario administrador
```

```
UPDATE posts SET usuario_id = 3 WHERE id IN (3, 4); -- Asignar a  
otro usuario
```

```
select *  
  
from posts;
```

The screenshot shows a SQL query editor with the query `select * from posts;` entered. Below the query bar, a table of results is displayed. The table has 9 columns: `id`, `titulo`, `contenido`, `fecha_creacion`, `fecha_actualizacion`, `destacado`, and `usuario_id`. The results show 5 rows of data.

	id	titulo	contenido	fecha_creacion	fecha_actualizacion	destacado	usuario_id
1	5	Título del post 5	Contenido del post 5	2024-05-23 16:59:08.213	2024-05-23 16:59:08.213	[ ]	[NULL]
2	1	Título del post 1	Contenido del post 1	2024-05-23 16:59:08.213	2024-05-23 16:59:08.213	[v]	1
3	2	Título del post 2	Contenido del post 2	2024-05-23 16:59:08.213	2024-05-23 16:59:08.213	[v]	1
4	3	Título del post 3	Contenido del post 3	2024-05-23 16:59:08.213	2024-05-23 16:59:08.213	[ ]	3
5	4	Título del post 4	Contenido del post 4	2024-05-23 16:59:08.213	2024-05-23 16:59:08.213	[ ]	3

## Creamos tabla Comentarios

```
CREATE TABLE comentarios (  
  
    id SERIAL PRIMARY KEY,  
  
    contenido TEXT,  
  
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  
    usuario_id BIGINT,  
  
    post_id BIGINT  
  
);
```

## salida

The screenshot shows the output of a SQL query in a dark-themed editor. The output is divided into two sections: 'Updated Rows' and 'Query'. The 'Updated Rows' section shows '0'. The 'Query' section shows the full SQL statement for creating the 'comentarios' table.

Name	Value
Updated Rows	0
Query	CREATE TABLE comentarios ( id SERIAL PRIMARY KEY, contenido TEXT, fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

```
usuario_id BIGINT,  
post_id BIGINT  
)
```

Start time Thu May 23 17:09:41 CLT 2024

Finish time Thu May 23 17:09:41 CLT 2024

## Insertamos Comentarios

```
INSERT INTO comentarios (id, contenido, usuario_id, post_id)  
VALUES  
  (1, 'Este es el comentario 1', 1, 1),  
  (2, 'Este es el comentario 2', 2, 1),  
  (3, 'Este es el comentario 3', 3, 1),  
  (4, 'Este es el comentario 4', 1, 2),  
  (5, 'Este es el comentario 5', 2, 2);
```

## Salida

Name	Value
Updated Rows	5
Query	<pre>INSERT INTO comentarios (id, contenido, usuario_id, post_id) VALUES   (1, 'Este es el comentario 1', 1, 1),   (2, 'Este es el comentario 2', 2, 1),   (3, 'Este es el comentario 3', 3, 1),   (4, 'Este es el comentario 4', 1, 2),   (5, 'Este es el comentario 5', 2, 2)</pre>
Start time	Thu May 23 17:11:46 CLT 2024
Finish time	Thu May 23 17:11:46 CLT 2024

2 Cruza los datos de la tabla usuarios y posts, mostrando las siguientes columnas:

nombre y email del usuario junto al título y contenido del post.

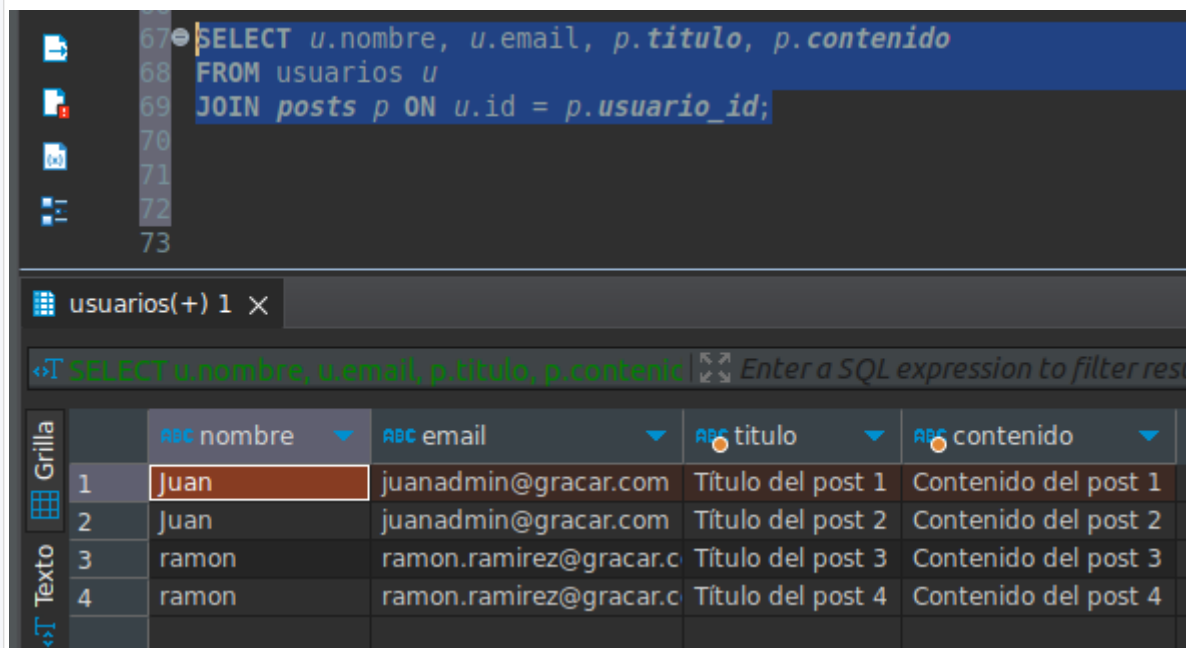
Como en este caso necesitamos mostrar solo las coincidencias utilizaremos JOIN o Inner Join:

## Código

```
SELECT u.nombre, u.email, p.titulo, p.contenido
FROM usuarios u
JOIN posts p ON u.id = p.usuario_id;
```

como alias para la tabla usuarios usamos u y para posts la letra p

## Salida



The screenshot shows a SQL IDE interface. At the top, a query is entered in a text editor:

```
SELECT u.nombre, u.email, p.titulo, p.contenido
FROM usuarios u
JOIN posts p ON u.id = p.usuario_id;
```

Below the editor, the results are displayed in a table view. The table has 5 columns: nombre, email, titulo, and contenido. The first two rows show results for the user 'Juan', and the next two rows show results for the user 'ramon'.

	nombre	email	titulo	contenido
1	Juan	juanadmin@gracar.com	Título del post 1	Contenido del post 1
2	Juan	juanadmin@gracar.com	Título del post 2	Contenido del post 2
3	ramon	ramon.ramirez@gracar.c	Título del post 3	Contenido del post 3
4	ramon	ramon.ramirez@gracar.c	Título del post 4	Contenido del post 4

### 3 Muestra el id, título y contenido de los posts de los administradores.

a. El administrador puede ser cualquier id.

## Código

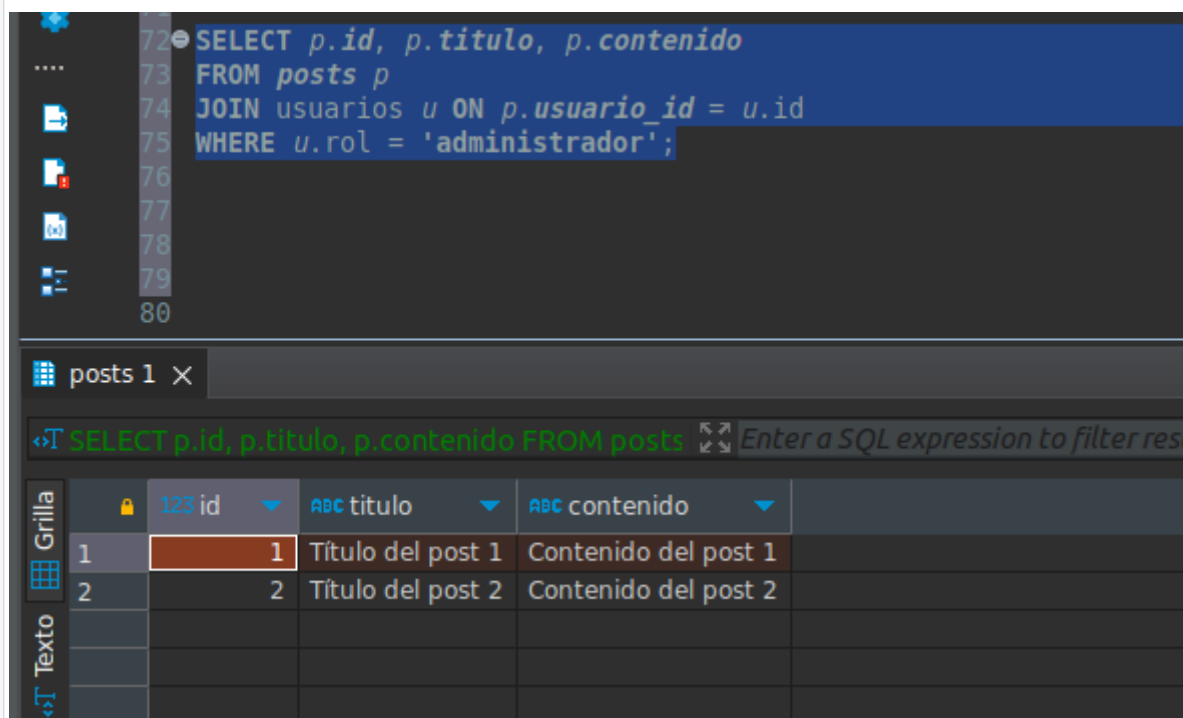
```
SELECT p.id, p.titulo, p.contenido

FROM posts p

JOIN usuarios u ON p.usuario_id = u.id

WHERE u.rol = 'administrador';
```

## Salida



The screenshot shows a SQL IDE interface. The top pane displays a SQL query: `SELECT p.id, p.titulo, p.contenido FROM posts p JOIN usuarios u ON p.usuario_id = u.id WHERE u.rol = 'administrador';`. The bottom pane shows the results in a table view. The table has 5 columns: an index, a lock icon, 'id', 'titulo', and 'contenido'. There are 2 rows of data.

		123 id	ABC titulo	ABC contenido
1		1	Título del post 1	Contenido del post 1
2		2	Título del post 2	Contenido del post 2

## 4 Cuenta la cantidad de posts de cada usuario.

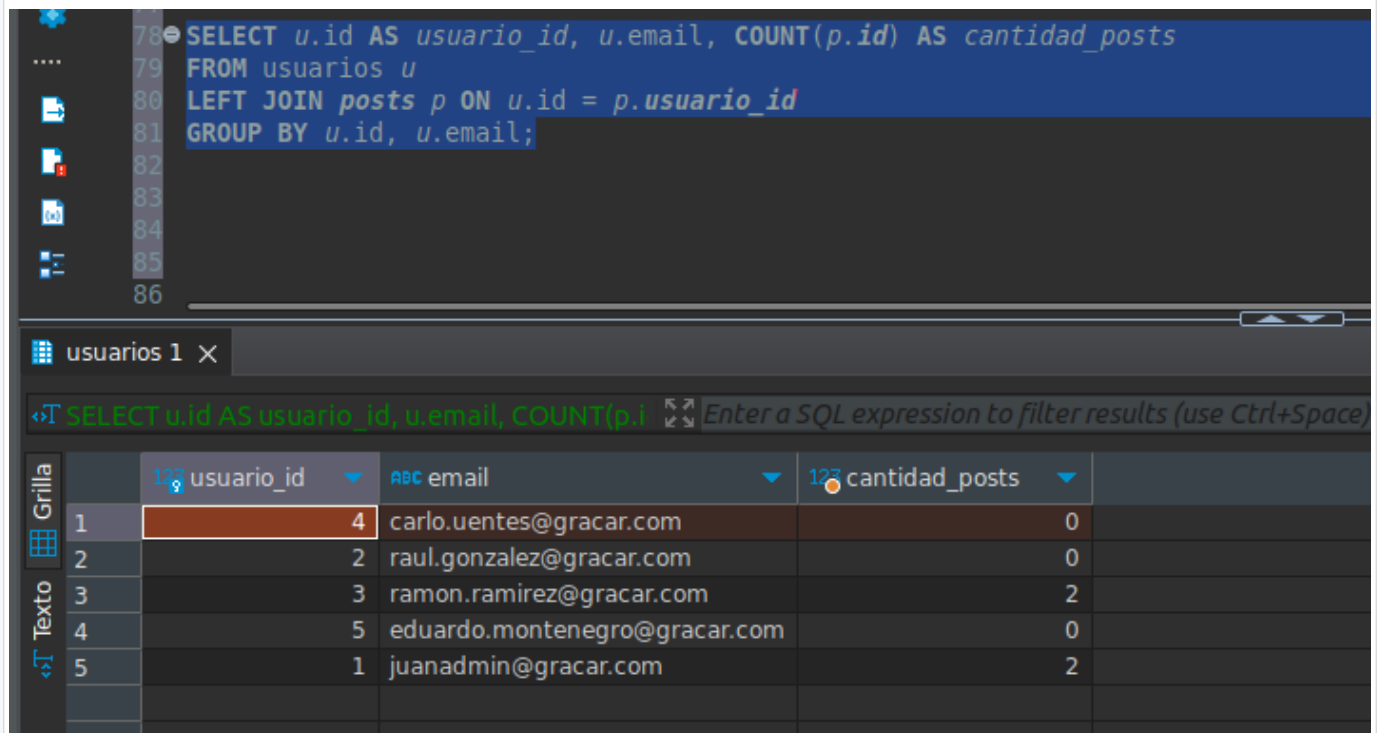
a. La tabla resultante debe mostrar el id e email del usuario junto con la cantidad de posts de cada usuario.

en este caso como tomamos la tabla usuarios como pivote y le agregamos información de la tabla posts utilizaremos left join

## Código

```
SELECT u.id AS usuario_id, u.email, COUNT(p.id) AS cantidad_posts  
  
FROM usuarios u  
  
LEFT JOIN posts p ON u.id = p.usuario_id  
  
GROUP BY u.id, u.email;
```

## Salida



The screenshot shows a SQL IDE interface. The top pane displays the SQL query: `SELECT u.id AS usuario_id, u.email, COUNT(p.id) AS cantidad_posts FROM usuarios u LEFT JOIN posts p ON u.id = p.usuario_id GROUP BY u.id, u.email;`. The bottom pane shows the results in a table view with columns: `usuario_id`, `email`, and `cantidad_posts`. The table contains 5 rows of data.

	usuario_id	email	cantidad_posts
1	4	carlo.uentes@gracar.com	0
2	2	raul.gonzalez@gracar.com	0
3	3	ramon.ramirez@gracar.com	2
4	5	eduardo.montenegro@gracar.com	0
5	1	juanadmin@gracar.com	2

## 5 Muestra el email del usuario que ha creado más posts.

a. Aquí la tabla resultante tiene un único registro y muestra solo el email

## Código

```
SELECT u.email  
  
FROM usuarios u  
  
LEFT JOIN posts p ON u.id = p.usuario_id
```



```
LEFT JOIN posts p ON u.id = p.usuario_id

GROUP BY u.id, u.email

ORDER BY COUNT(p.id) DESC

LIMIT 1;
```

## Salida

The screenshot shows a SQL IDE interface. The top pane displays a SQL query:

```
83 SELECT u.email
84 FROM usuarios u
85 LEFT JOIN posts p ON u.id = p.usuario_id
86 GROUP BY u.id, u.email
87 ORDER BY COUNT(p.id) DESC
88 LIMIT 1;
```

The bottom pane shows the results of the query in a table view. The table has one column, 'email', and one row with the value 'ramon.ramirez@gracar.com'. The table is titled 'usuarios 1'.

email
ramon.ramirez@gracar.com

## 6. Muestra la fecha del último post de cada usuario

## Código

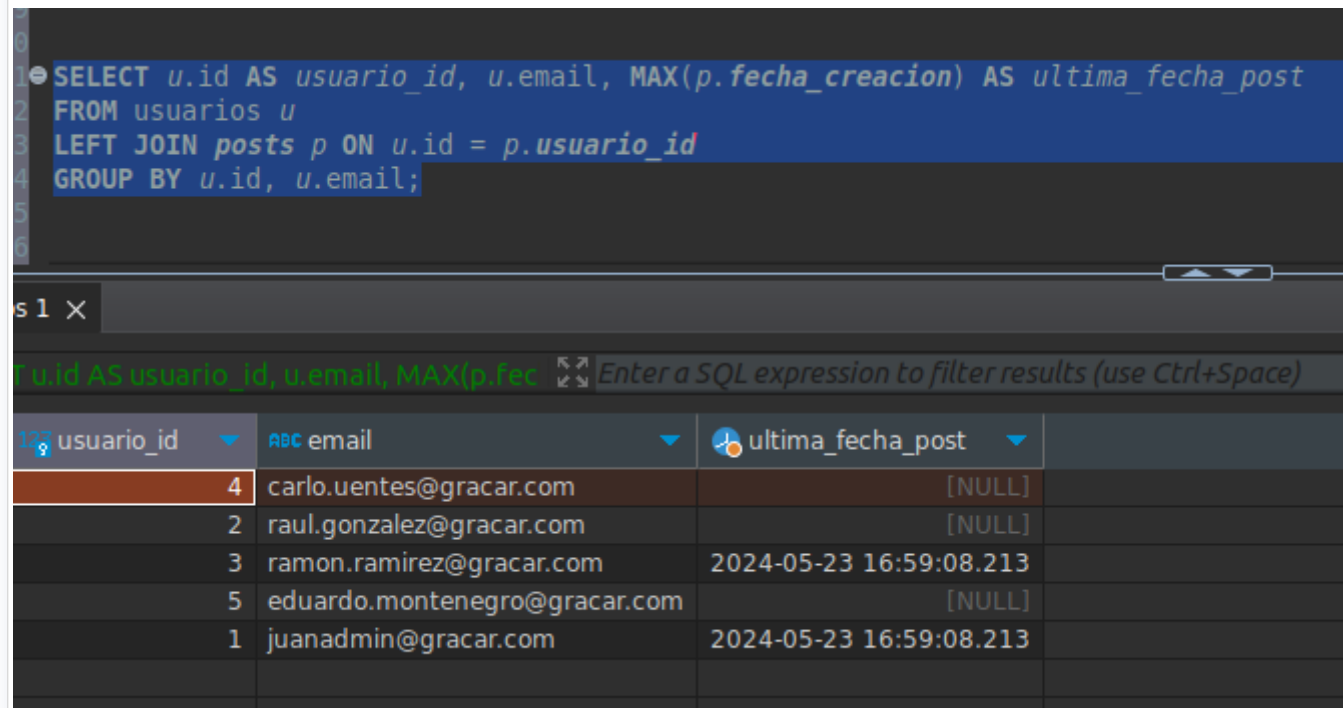
```
SELECT u.id AS usuario_id, u.email, MAX(p.fecha_creacion) AS
ultima_fecha_post

FROM usuarios u

LEFT JOIN posts p ON u.id = p.usuario_id

GROUP BY u.id, u.email;
```

## Salida



The screenshot shows a SQL IDE with a query editor at the top and a results pane below. The query is:

```
SELECT u.id AS usuario_id, u.email, MAX(p.fecha_creacion) AS ultima_fecha_post
FROM usuarios u
LEFT JOIN posts p ON u.id = p.usuario_id
GROUP BY u.id, u.email;
```

The results pane shows a table with 5 rows and 3 columns: `usuario_id`, `email`, and `ultima_fecha_post`. The data is as follows:

usuario_id	email	ultima_fecha_post
4	carlo.uentes@gracar.com	[NULL]
2	raul.gonzalez@gracar.com	[NULL]
3	ramon.ramirez@gracar.com	2024-05-23 16:59:08.213
5	eduardo.montenegro@gracar.com	[NULL]
1	juanadmin@gracar.com	2024-05-23 16:59:08.213

7. Muestra el título y contenido del post (artículo) con más comentarios.

## Código

```
SELECT p.titulo, p.contenido
FROM posts p
LEFT JOIN comentarios c ON p.id = c.post_id
GROUP BY p.id, p.titulo, p.contenido
ORDER BY COUNT(c.id) DESC
LIMIT 1;
```

## Salida



The screenshot shows a SQL IDE with a query editor. The query is:

```
SELECT p.titulo, p.contenido
FROM posts p
```

```
98 LEFT JOIN comentarios c ON p.id = c.post_id
99 GROUP BY p.id, p.titulo, p.contenido
100 ORDER BY COUNT(c.id) DESC
101 LIMIT 1;
102
103
104
105
106
```

posts 1 ×

SELECT p.titulo, p.contenido FROM posts p LE *Enter a SQL expression to filter results (use Ctrl+Space)*

🔒	ASC titulo	ABC contenido	
	Título del post 1	Contenido del post 1	

## 8-Muestra en una tabla el título de cada post, el contenido de cada post y el contenido

de cada comentario asociado a los posts mostrados, junto con el email del usuario que lo escribió.

## Código

```
SELECT p.titulo AS "Título del Post", p.contenido AS "Contenido del Post",
       c.contenido AS "Contenido del Comentario", u.email AS "Email del
Usuario"
FROM posts p
LEFT JOIN comentarios c ON p.id = c.post_id
LEFT JOIN usuarios u ON c.usuario_id = u.id;
```

## Salida

```
103 SELECT p.titulo AS "Título del Post", p.contenido AS "Contenido del Post",
104        c.contenido AS "Contenido del Comentario", u.email AS "Email del Usuario"
105 FROM posts p
106 LEFT JOIN comentarios c ON p.id = c.post_id
107 LEFT JOIN usuarios u ON c.usuario_id = u.id;
108
```

(+) 1 ×

ET p.titulo AS "Título del Post", p.conteni *Enter a SQL expression to filter results (use Ctrl+Space)*

ABC Título del Post	ABC Contenido del Post	ABC Contenido del Comentario	ABC Email del Usuario	
Título del post 1	Contenido del post 1	Este es el comentario 1	juanadmin@gracar.com	
Título del post 1	Contenido del post 1	Este es el comentario 2	raul.gonzalez@gracar.com	
Título del post 1	Contenido del post 1	Este es el comentario 3	ramon.ramirez@gracar.com	
Título del post 2	Contenido del post 2	Este es el comentario 4	juanadmin@gracar.com	
Título del post 2	Contenido del post 2	Este es el comentario 5	raul.gonzalez@gracar.com	
Título del post 5	Contenido del post 5	[NULL]	[NULL]	
Título del post 4	Contenido del post 4	[NULL]	[NULL]	
Título del post 3	Contenido del post 3	[NULL]	[NULL]	

## 9 Muestra el contenido del último comentario de cada usuario

### Código

```
SELECT u.email
FROM usuarios u
LEFT JOIN comentarios c ON u.id = c.usuario_id
WHERE c.id IS NULL;
```

### Salida

```
110 SELECT u.email, c.contenido AS "Último Comentario"
111 FROM usuarios u
112 LEFT JOIN comentarios c ON u.id = c.usuario_id
113 WHERE c.fecha_creacion = (SELECT MAX(fecha_creacion) FROM comentarios WHERE usuario_id = u.id);
114
115
116 SELECT u.email
```

usuarios(+) 1 X

CT u.email, c.contenido AS "Último Come" Enter a SQL expression to filter results (use Ctrl+Space)

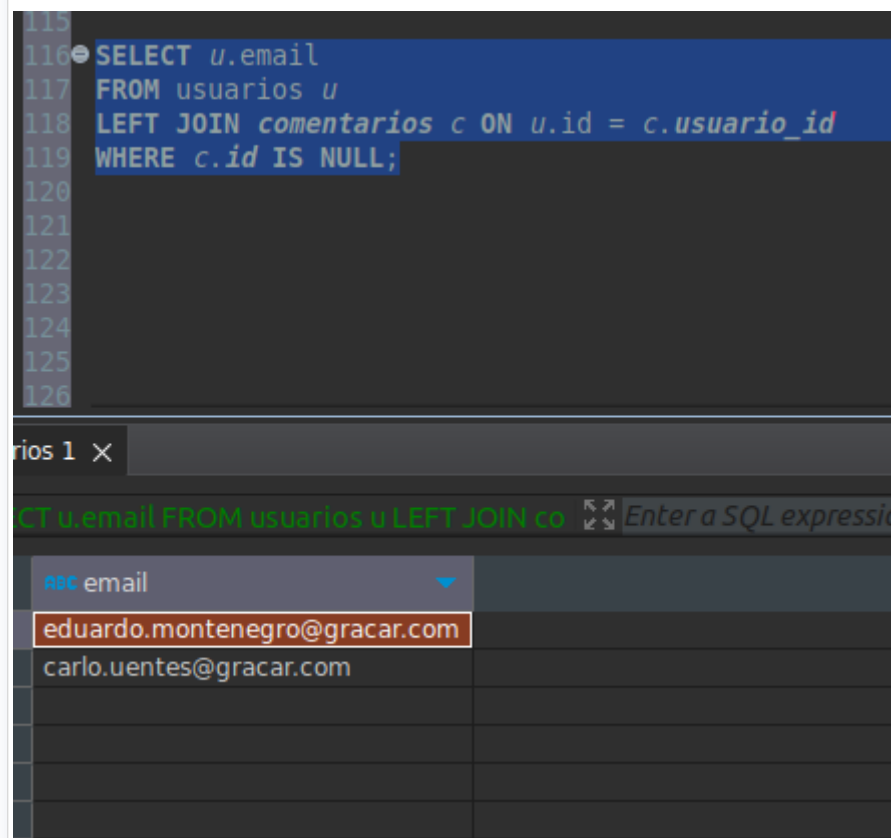
asc email	asc Último Comentario	
juanadmin@gracar.com	Este es el comentario 1	
raul.gonzalez@gracar.com	Este es el comentario 2	
ramon.ramirez@gracar.com	Este es el comentario 3	
juanadmin@gracar.com	Este es el comentario 4	
raul.gonzalez@gracar.com	Este es el comentario 5	

## 10 Muestra los emails de los usuarios que no han escrito ningún comentario.

## Código

```
SELECT u.email  
  
FROM usuarios u  
  
LEFT JOIN comentarios c ON u.id = c.usuario_id  
  
WHERE c.id IS NULL;
```

## Salida



The screenshot shows a SQL IDE interface. The top part displays a SQL query in a dark-themed editor with line numbers 115 to 126. The query is: `SELECT u.email FROM usuarios u LEFT JOIN comentarios c ON u.id = c.usuario_id WHERE c.id IS NULL;`. Below the editor, there is a tab labeled 'usuarios 1' with a close button. Below the tab, there is a table with the query results. The table has two columns: 'email' and an empty column. The first row contains the email 'eduardo.montenegro@gracar.com', and the second row contains 'carlo.uentes@gracar.com'. The table is highlighted with a blue selection bar.

```
115  
116 SELECT u.email  
117 FROM usuarios u  
118 LEFT JOIN comentarios c ON u.id = c.usuario_id  
119 WHERE c.id IS NULL;  
120  
121  
122  
123  
124  
125  
126
```

usuarios 1 ×

SELECT u.email FROM usuarios u LEFT JOIN co Enter a SQL expressio

email	
eduardo.montenegro@gracar.com	
carlo.uentes@gracar.com	

```
--- item 1  
--- 1.1 crear BBDD  
create database desafio3_JuanPablo_Grau_315_3digitos;  
  
--- 1.2 crear tabla usuarios  
  
CREATE TABLE usuarios (  
    id SERIAL PRIMARY KEY,
```

```
rol VARCHAR,
email VARCHAR,
nombre VARCHAR,
apellido VARCHAR
);

--- 1.3 insertar datos tabla usuarios

INSERT INTO usuarios (rol, email, nombre, apellido)
VALUES
    ('administrador', 'juanadmin@gracar.com', 'Juan', 'grau'),
    ('usuario', 'raul.gonzalez@gracar.com', 'raul', 'gonzalez'),
    ('usuario', 'ramon.ramirez@gracar.com', 'ramon', 'ramirez'),
    ('usuario', 'carlo.uentes@gracar.com', 'carlos', 'fuentes'),
    ('usuario', 'eduardo.montenegro@gracar.com', 'eduardo', 'Montenegro');

select *
from usuarios;
```

--- 1.4 crea tabla posts

```
CREATE TABLE posts (
    id SERIAL PRIMARY KEY,
    titulo VARCHAR,
    contenido TEXT,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    destacado BOOLEAN,
    usuario_id BIGINT
);
```

--- 1.5 insertar datos tabla posts

```
INSERT INTO posts (id, titulo, contenido, fecha_creacion,
fecha_actualizacion, destacado, usuario_id)
VALUES
    (1, 'Título del post 1', 'Contenido del post 1', CURRENT_TIMESTAMP,
CURRENT_TIMESTAMP, true, 1),
    (2, 'Título del post 2', 'Contenido del post 2', CURRENT_TIMESTAMP,
CURRENT_TIMESTAMP, true, 1),
    (3, 'Título del post 3', 'Contenido del post 3', CURRENT_TIMESTAMP,
CURRENT_TIMESTAMP, false, 2),
    (4, 'Título del post 4', 'Contenido del post 4', CURRENT_TIMESTAMP,
CURRENT_TIMESTAMP, false, 2),
    (5, 'Título del post 5', 'Contenido del post 5', CURRENT_TIMESTAMP,
CURRENT_TIMESTAMP, false, NULL);
```

--- 1.6 luego asigno usuarios con update

```
UPDATE posts SET usuario_id = 1 WHERE id IN (1, 2); -- Asignar al
usuario administrador
UPDATE posts SET usuario_id = 3 WHERE id IN (3, 4); -- Asignar a
otro usuario
```

```
select *
from posts;
```

--- 1.7 crea tabla comentarios

```
CREATE TABLE comentarios (
  id SERIAL PRIMARY KEY,
  contenido TEXT,
  fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  usuario_id BIGINT,
  post_id BIGINT
);
```

--- 1.8 inserta datos tabla comentarios

```
INSERT INTO comentarios (id, contenido, usuario_id, post_id)
VALUES
  (1, 'Este es el comentario 1', 1, 1),
  (2, 'Este es el comentario 2', 2, 1),
  (3, 'Este es el comentario 3', 3, 1),
  (4, 'Este es el comentario 4', 1, 2),
  (5, 'Este es el comentario 5', 2, 2);
```

— item 2 Cruza los datos de la tabla usuarios y posts, mostrando las siguientes columnas:

— nombre y email del usuario junto al título y contenido del post.

```
SELECT u.nombre, u.email, p.titulo, p.contenido
FROM usuarios u
JOIN posts p ON u.id = p.usuario_id;
```

--- item 3 Muestra el id, título y contenido de los posts de los administradores El administrador puede ser cualquier id.

```
SELECT p.id, p.titulo, p.contenido
FROM posts p
JOIN usuarios u ON p.usuario_id = u.id
WHERE u.rol = 'administrador';
```

--- item 4 Cuenta la cantidad de posts de cada usuario.

```
SELECT u.id AS usuario_id, u.email, COUNT(p.id) AS cantidad_posts
FROM usuarios u
```

```
LEFT JOIN posts p ON u.id = p.usuario_id
GROUP BY u.id, u.email;
```

--- item 5 Muestra el email del usuario que ha creado más posts.

```
SELECT u.email
FROM usuarios u
LEFT JOIN posts p ON u.id = p.usuario_id
GROUP BY u.id, u.email
ORDER BY COUNT(p.id) DESC
LIMIT 1;
```

--- item 6 Muestra la fecha del último post de cada usuario

```
SELECT u.id AS usuario_id, u.email, MAX(p.fecha_creacion) AS
ultima_fecha_post
FROM usuarios u
LEFT JOIN posts p ON u.id = p.usuario_id
GROUP BY u.id, u.email;
```

--- item 7 Muestra el título y contenido del post (artículo) con más comentarios.

```
SELECT p.titulo, p.contenido
FROM posts p
LEFT JOIN comentarios c ON p.id = c.post_id
GROUP BY p.id, p.titulo, p.contenido
ORDER BY COUNT(c.id) DESC
LIMIT 1;
```

--- item 8 Muestra en una tabla el título de cada post, el contenido de cada post y el contenido

```
SELECT p.titulo AS "Título del Post", p.contenido AS "Contenido del Post",
       c.contenido AS "Contenido del Comentario", u.email AS "Email del
Usuario"
FROM posts p
LEFT JOIN comentarios c ON p.id = c.post_id
LEFT JOIN usuarios u ON c.usuario_id = u.id;
```

--- item 9 Muestra el contenido del último comentario de cada usuario

```
SELECT u.email, c.contenido AS "Último Comentario"
FROM usuarios u
LEFT JOIN comentarios c ON u.id = c.usuario_id
WHERE c.fecha_creacion = (SELECT MAX(fecha_creacion) FROM comentarios WHERE
usuario id = u.id);
```



```
--- item 10 Muestra los emails de los usuarios que no han escrito ningún comentario.  
SELECT u.email  
FROM usuarios u  
LEFT JOIN comentarios c ON u.id = c.usuario_id  
WHERE c.id IS NULL;
```