

Bases de datos – Parcial 2

Jorge Esteban Herrera Jimenez – 833060

Corporación Universitaria Minuto de Dios

Bases de Datos Masivas

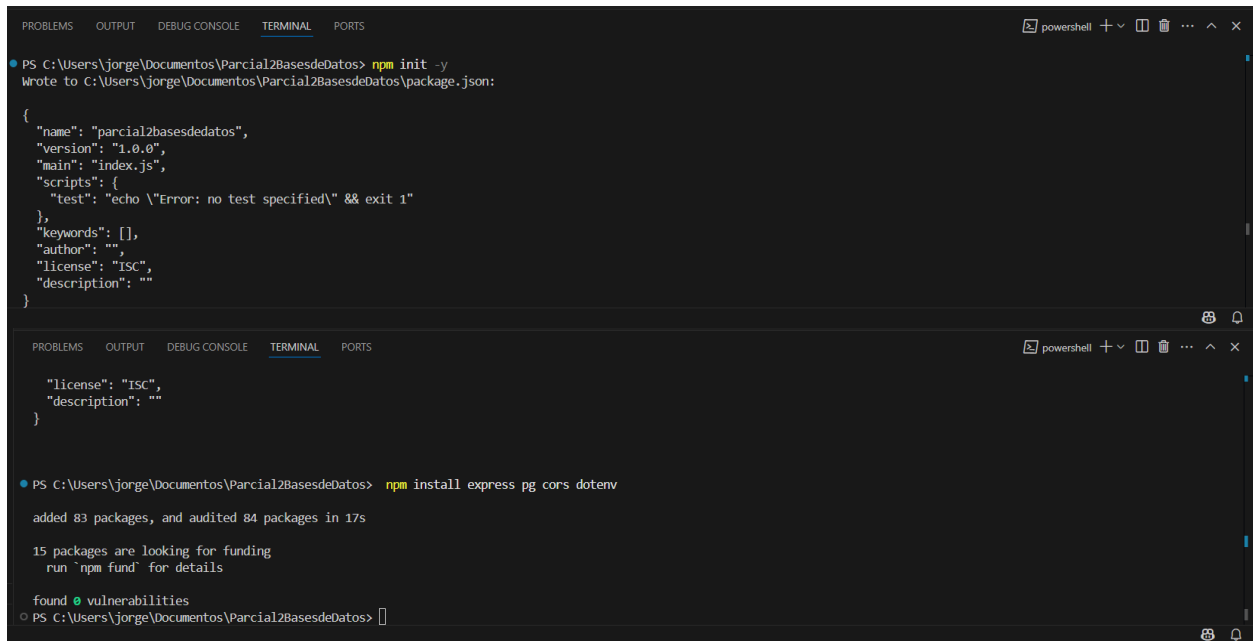
Ing. De Sistemas

Ing. William Alexander Matallana Porras

2025

1. Primeramente, vamos a crear una carpeta en la ruta que deseemos para llevar a cabo nuestro proyecto, una vez abierto vamos a colocar los siguientes comandos en la terminal:

- `npm init -y`
- `npm install express pg cors dotenv`



The image shows two screenshots of a PowerShell terminal window. The first screenshot shows the command `npm init -y` being executed, which creates a `package.json` file. The output shows the contents of the generated `package.json` file, including the project name `parcial2basesdedatos`, version `1.0.0`, and a test script. The second screenshot shows the command `npm install express pg cors dotenv` being executed, which installs the specified dependencies. The output shows that 83 packages were added and audited, and 15 packages are looking for funding.

```
PS C:\Users\jorge\Documentos\Parcial2BasesdeDatos> npm init -y
Wrote to C:\Users\jorge\Documentos\Parcial2BasesdeDatos\package.json:

{
  "name": "parcial2basesdedatos",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

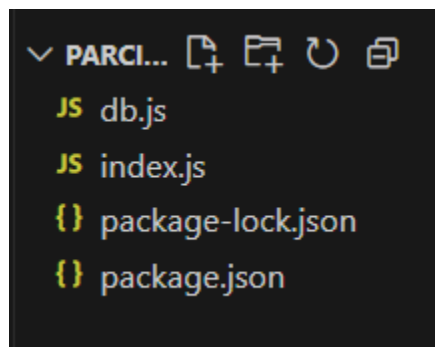
PS C:\Users\jorge\Documentos\Parcial2BasesdeDatos> npm install express pg cors dotenv

added 83 packages, and audited 84 packages in 17s

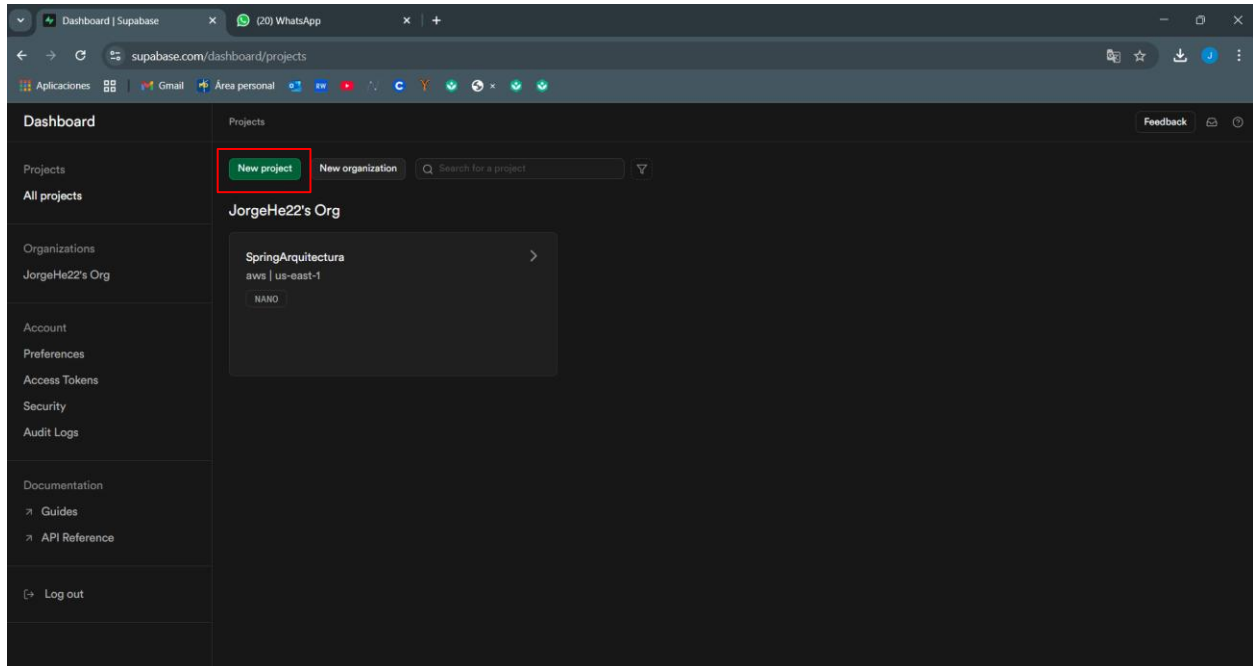
15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\jorge\Documentos\Parcial2BasesdeDatos>
```

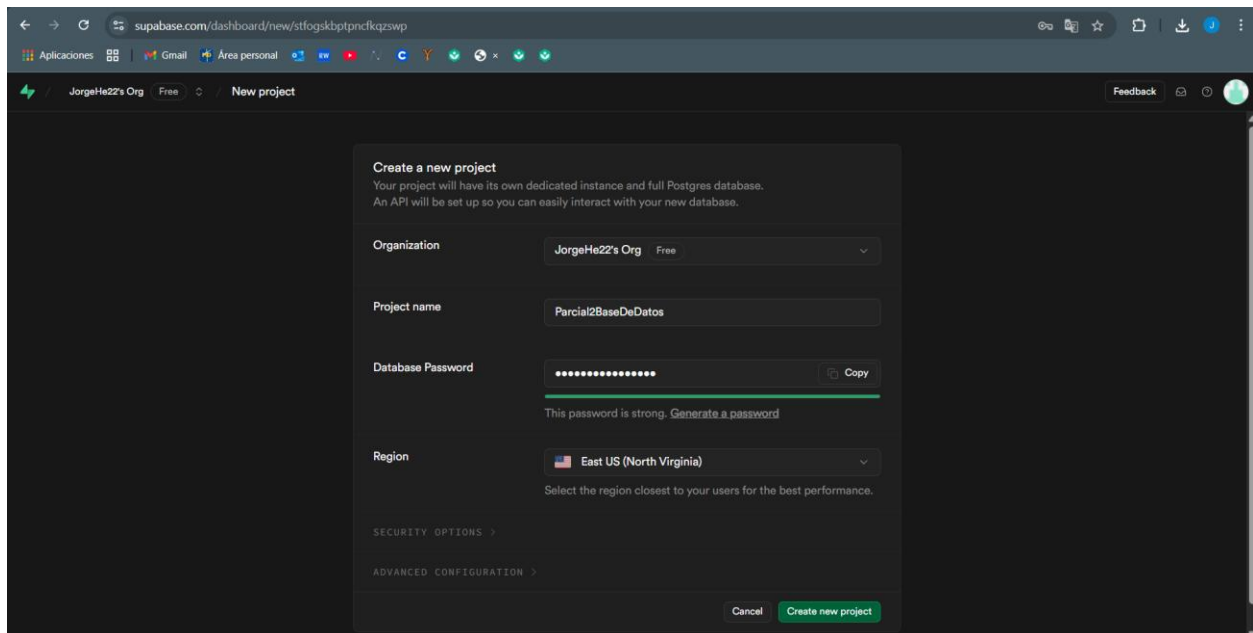
2. Ahora vamos a crear las siguientes archivos adicionales que vamos a necesitar para llevar a cabo nuestro trabajo:



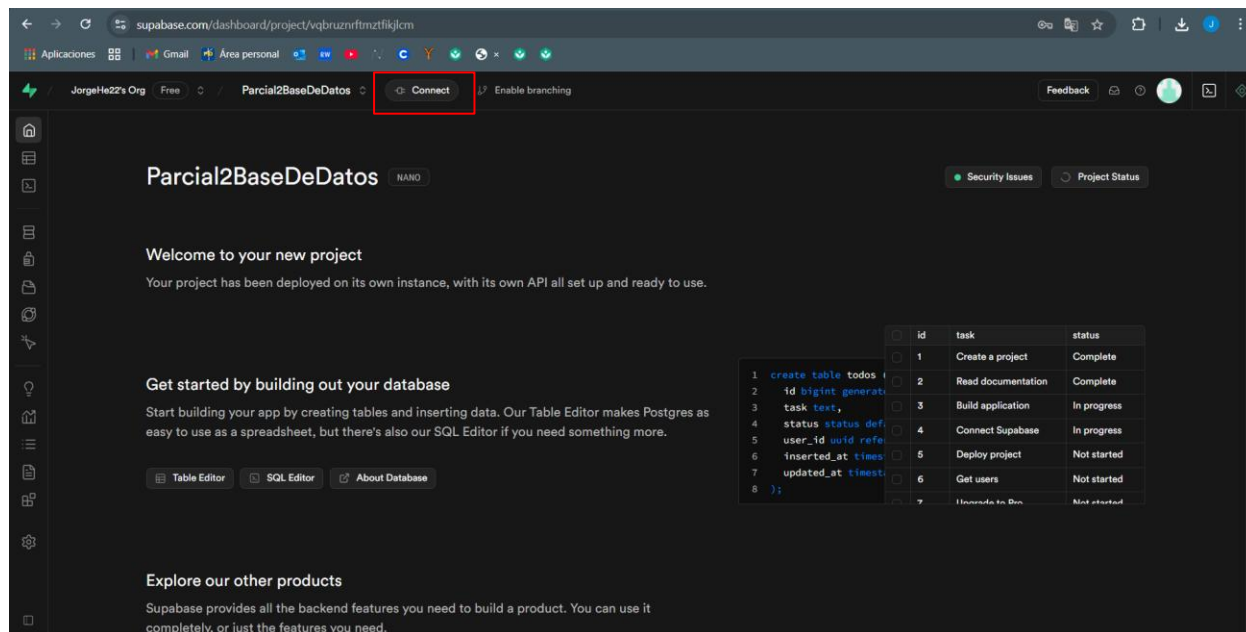
3. **Creación en supabase:** Ahora vamos a dirigirnos a la pagina de supabase y vamos a crear un nuevo proyecto:



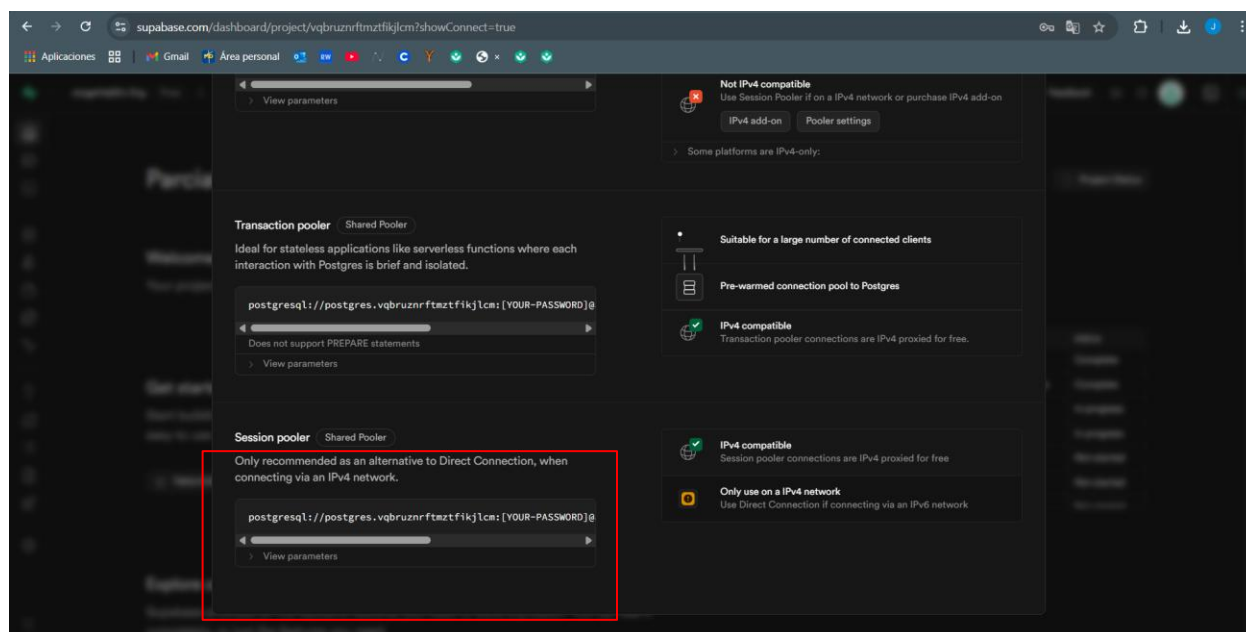
Seguidamente ingresamos la información requerida de manera que quede así:



Luego de haber creado el proyecto, nos vamos al apartado de “**Connect**” para empezar a realizar la conexión de la base de datos a pgAdmin4.



Después de darle clic en “**Connect**” nos abre una ventana que nos ubica en la sección de “**Connection Sting**” en donde vamos a bajar hasta el apartado de “**Session Poler**” y copiamos la información, la cual usaremos seguidamente en pgAdmin4.

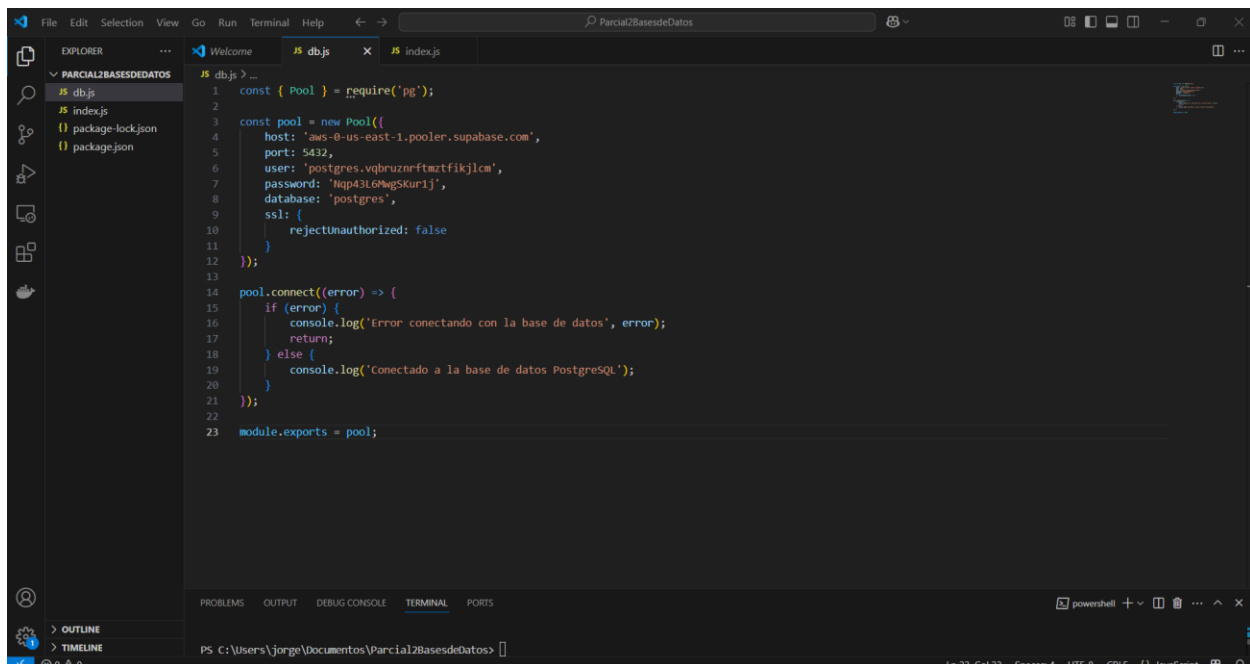


4. **Conexión en visual studio:** Para conectar el supabase con nuestro proyecto en visual studio vamos al archivo que creamos anteriormente “db.js” y aca vamos a poner el siguiente código en el cual es importante resaltar que los apartados de host, user y password, vamos a poner la contraseña con la cual creamos el proyecto y el link que copiamos en el apartado de connect en supabase lo vamos a dividir de la siguiente manera

```
postgres://postgres.vqbuznrftmztfikjlc:[YOUR-PASSWORD]@aws-0-us-east-1.pooler.supabase.com:5432/postgres
```

- **Host name/address:** Aca vamos a colocar la parte que esta subrayada de amarillo.
- **Username:** Acá vamos a colocar la parte que esta subrayada de azul.
- **Password:** Acá vamos a colocar la contraseña que usamos para crear nuestro proyecto

Teniendo en cuenta esto, debería quedar de la siguiente manera:

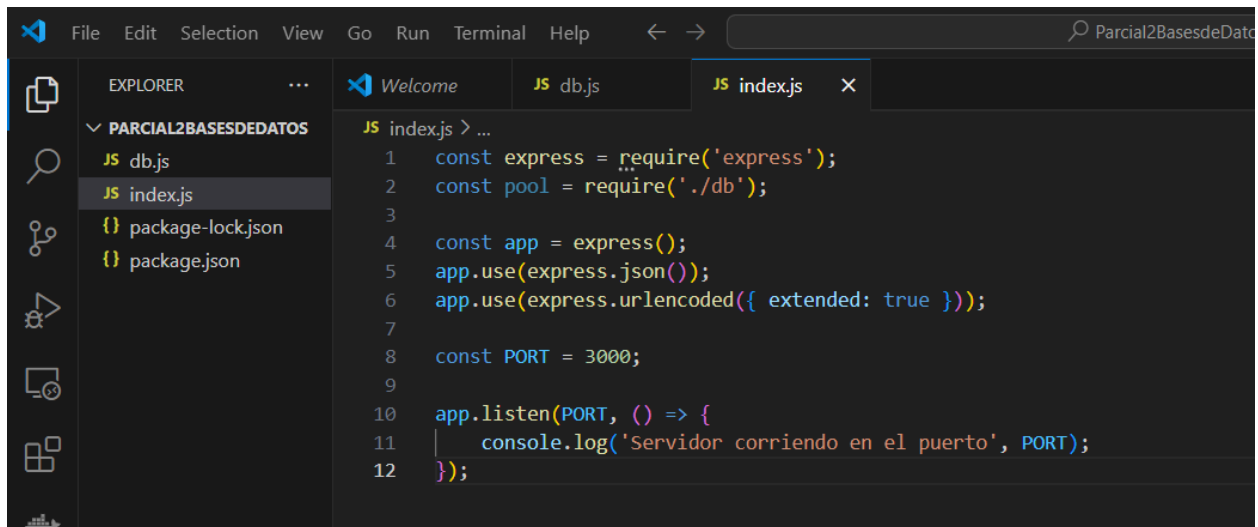


```

1  const { Pool } = require('pg');
2
3  const pool = new Pool({
4    host: 'aws-0-us-east-1.pooler.supabase.com',
5    port: 5432,
6    user: 'postgres.vqbuznrftmztfikjlc',
7    password: 'Nqp43l6Wg5kur1j',
8    database: 'postgres',
9    ssl: {
10     rejectUnauthorized: false
11   }
12 });
13
14 pool.connect((error) => {
15   if (error) {
16     console.log('Error conectando con la base de datos', error);
17     return;
18   } else {
19     console.log('Conectado a la base de datos PostgreSQL');
20   }
21 });
22
23 module.exports = pool;

```

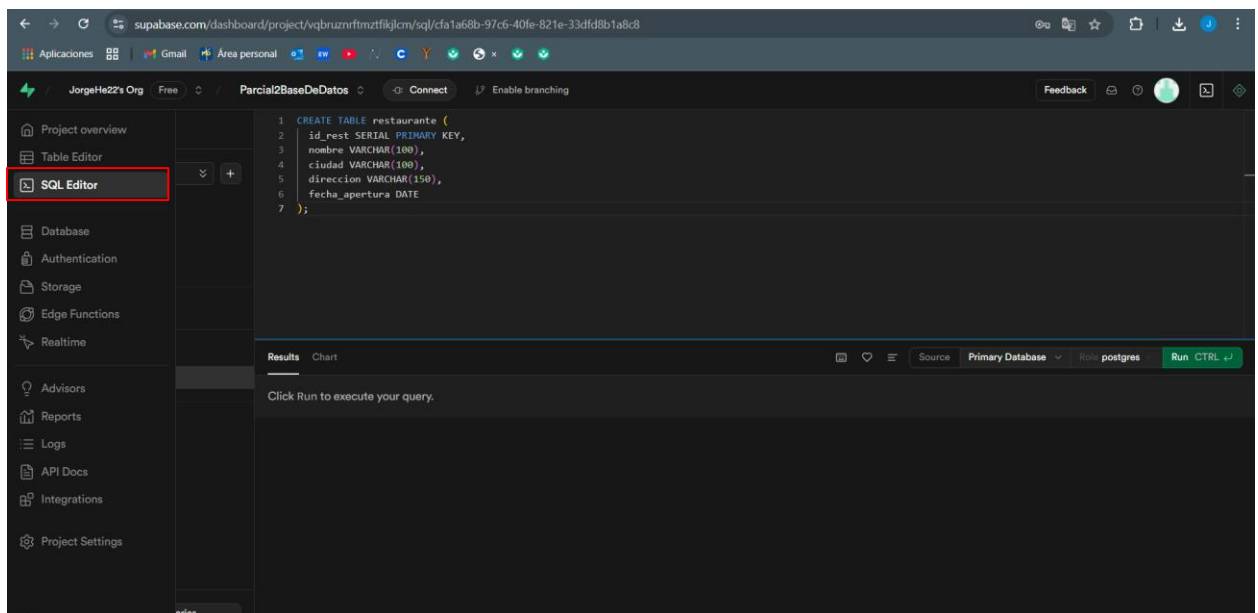
Ahora en el index vamos a hacer la configuración básica inicial de nuestra API en express, por lo cual nos debe quedar de la siguiente manera



```

1  const express = require('express');
2  const pool = require('./db');
3
4  const app = express();
5  app.use(express.json());
6  app.use(express.urlencoded({ extended: true }));
7
8  const PORT = 3000;
9
10 app.listen(PORT, () => {
11   console.log('Servidor corriendo en el puerto', PORT);
12 });
  
```

5. **Creación de tablas:** Para crear las tablas vamos a usar supabase, una vez en la pagina vamos a la parte izquierda y seleccionamos la parte de “SQL Editor” y aca ponemos los códigos para la creación de las tablas según se requieran:



```







1 CREATE TABLE restaurante (
2   id_rest SERIAL PRIMARY KEY,
3   nombre VARCHAR(100),
4   ciudad VARCHAR(100),
5   direccion VARCHAR(150),
6   fecha_apertura DATE
7 );
  
```

Y así sucesivamente dependiendo la cantidad de tablas que necesitamos.

```
1 CREATE TABLE empleado (  
2   id_empleado SERIAL PRIMARY KEY,  
3   nombre VARCHAR(100),  
4   rol VARCHAR(50),  
5   id_rest INT,  
6   FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest) ON DELETE CASCADE  
7 );
```

Results Chart Export ▾

Success. No rows returned

Parcial2BaseDeDatos  Connect  Enable branching Feedback    

```
1 CREATE TABLE producto (  
2   id_prod SERIAL PRIMARY KEY,  
3   nombre VARCHAR(100),  
4   precio NUMERIC(10,2)  
5 );
```

Results Chart Export ▾

Success. No rows returned

```
1 CREATE TABLE pedido (  
2   id_pedido SERIAL PRIMARY KEY,  
3   fecha DATE,  
4   id_rest INT,  
5   total NUMERIC(10,2),  
6   FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest) ON DELETE CASCADE  
7 );
```

Results Chart Export ▾

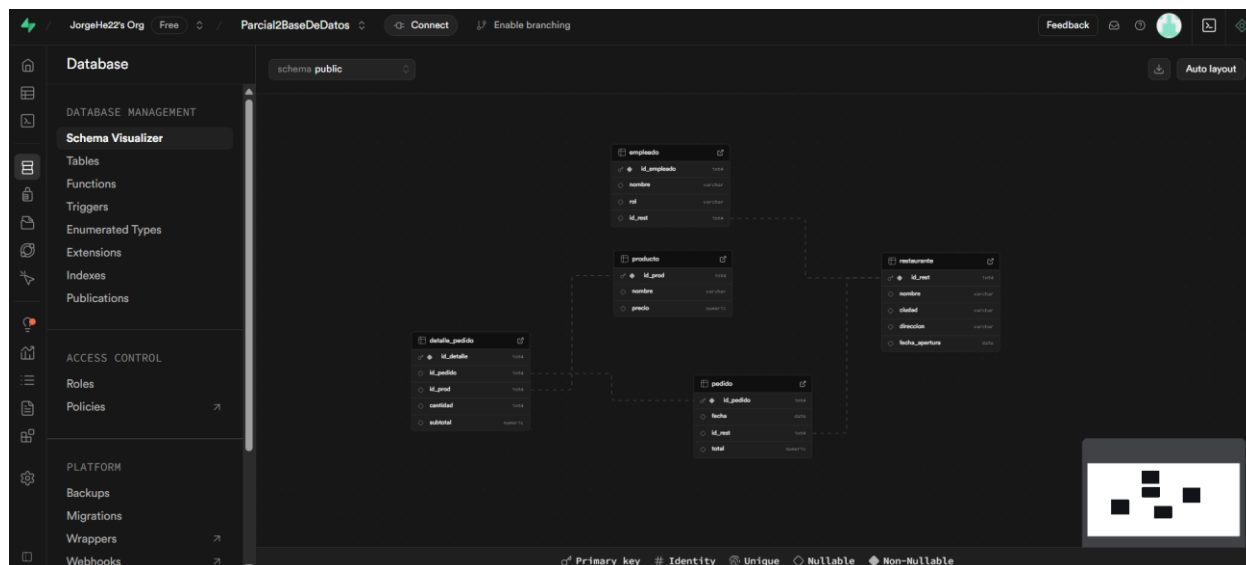
Success. No rows returned

```
1 CREATE TABLE detalle_pedido (  
2   id_detalle SERIAL PRIMARY KEY,  
3   id_pedido INT,  
4   id_prod INT,  
5   cantidad INT,  
6   subtotal NUMERIC(10,2),  
7   FOREIGN KEY (id_pedido) REFERENCES pedido(id_pedido) ON DELETE CASCADE,  
8   FOREIGN KEY (id_prod) REFERENCES producto(id_prod) ON DELETE CASCADE  
9 );
```

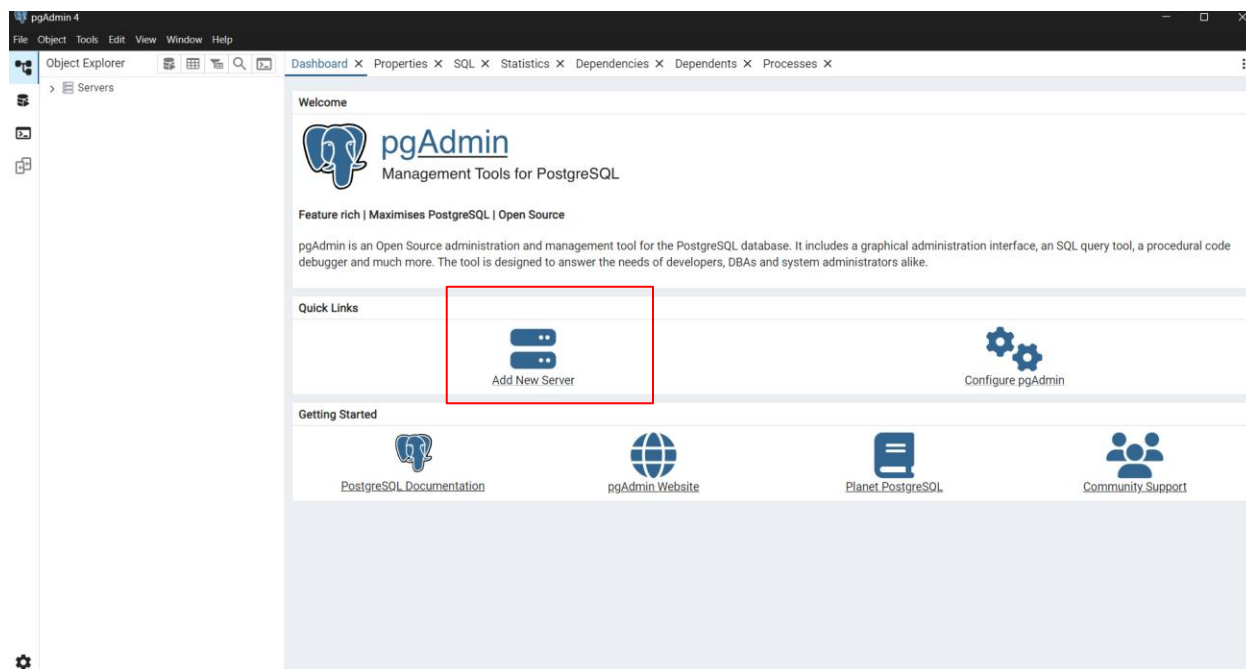
Results Chart Export ▾

Success. No rows returned

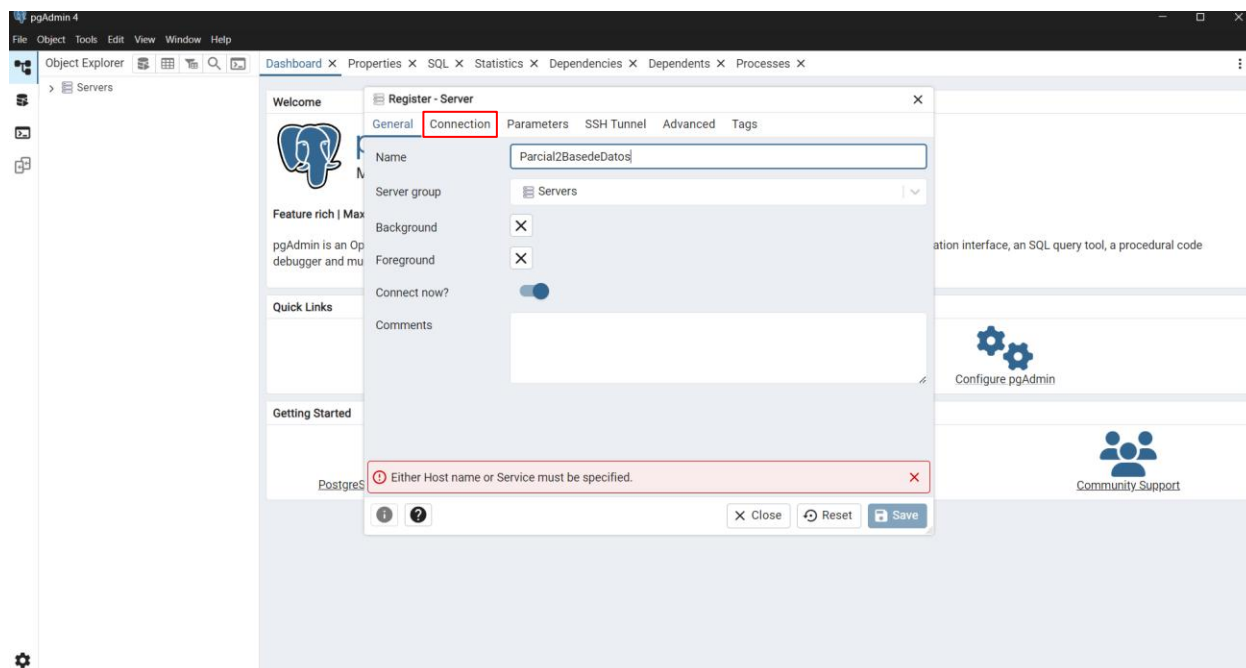
Una vez creadas las tablas, nos vamos a dirigir ahora a la parte de “database” y podremos ver la conexión de nuestras tablas:



6. Conexión en pgAdmin: Una vez abierto pgAdmin vamos a la parte de “Add new server”



Luego le colocamos un nombre y seguidamente nos dirigimos al apartado de “**Connection**”

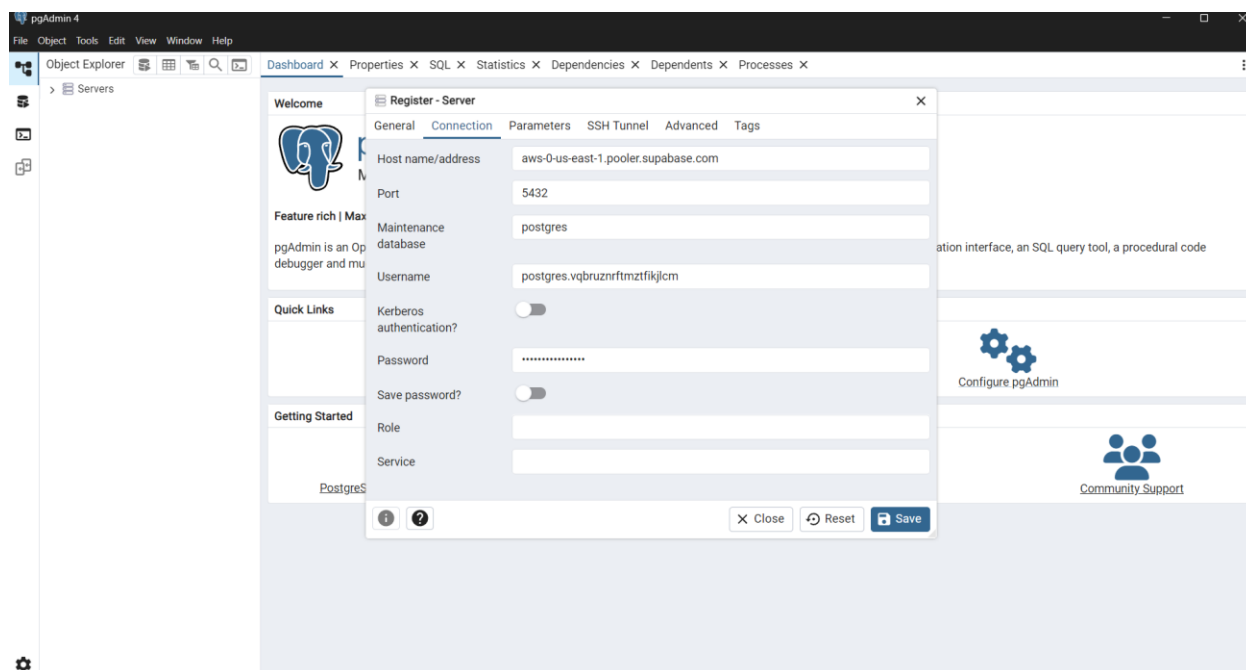


Luego, en el apartado de “**Connection**” encontramos 3 apartados los cuales son: Host name/address, username, y password. Acá lo que vamos a colocar en cada sección tendrá que ver con los pasos anteriores, el link copiado y la contraseña creada, lo cual vamos a dividir de la siguiente manera:

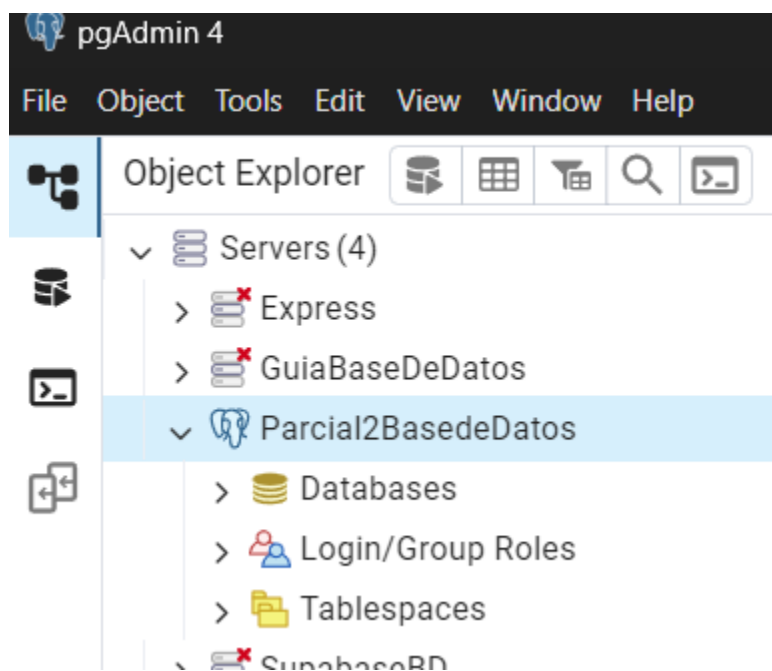
postgres://postgres.vqbruznrftmztfikjlc:[YOUR-PASSWORD]@aws-0-us-east-1.pooler.supabase.com:5432/postgres

- **Host name/address:** Aca vamos a colocar la parte que esta subrayada de amarillo.
- **Username:** Acá vamos a colocar la parte que esta subrayada de azul.
- **Password:** Acá vamos a colocar la contraseña que usamos para crear nuestro proyecto

Teniendo en cuenta esto, debería quedar de la siguiente manera:



Después de poner todo en su respectivo campo le damos “**Save**” y debe salir de la siguiente manera:



7. **Registro de datos en supabase:** Para agregar los registros que se nos solicita vamos a ir al menú de la parte izquierda y damos clic en “SQL Editor” aca vamos a poner los respectivos scripts para agregar los 50 datos a cada tabla:

```

1 INSERT INTO restaurante (id_rest, nombre, ciudad, direccion, fecha_apertura) VALUES
2 (1, 'Robertson-Alvarado', 'East Andrew', '444 Price Via Suite 006', '2018-03-24'),
3 (2, 'Nelson and Sons', 'Morgantown', '397 Zimmerman Trace Apt. 160', '2022-05-14'),
4 (3, 'Johnson-Lee', 'Mckenzie-mouth', '1531 White Knolls', '2023-12-08'),
5 (4, 'Davis PLC', 'Davisstad', '05698 Simmons Shores', '2018-03-01'),
6 (5, 'Williams, Allen and French', 'Toddhaven', '106 Ross Circle Suite 229', '2024-01-21'),
7 (6, 'Cooke LLC', 'Coreyview', '63829 Brown Harbor Suite 116', '2025-03-05'),
8 (7, 'Hall, Wright and Jenkins', 'South Ginebury', '8019 Oneal Flats Suite 341', '2017-03-15'),
9 (8, 'Soto Inc', 'Lake Melissaland', '654 Bautista Mount', '2023-05-09'),
10 (9, 'Bartlett, Bradley and Walker', 'Hornmouth', '34337 Carlos Circle', '2022-01-02'),
11 (10, 'Huber PLC', 'Greenemouth', '467 Sonya Course', '2024-06-18'),
12 (11, 'Long and Sons', 'Julianville', '8019 Monica Knolls Suite 612', '2020-12-28'),
13 (12, 'Taylor-George', 'Berrybury', '2888 Nichols Orchard Apt. 630', '2025-01-08'),
14 (13, 'Gamble-Reed', 'Parkerton', '67468 Horn Burgs', '2019-04-06'),
15 (14, 'Ollion-Villegas', 'Delgademouth', '9158 Rojas Oval Apt. 267', '2018-12-16'),
16 (15, 'Baker-Walsh', 'West Joshua', '60367 David Port', '2015-06-06'),
17 (16, 'Anderson-Thomas', 'East Jasontown', '42623 Hurst Track', '2017-04-08'),
18 (17, 'Hansen, Powell and Williamson', 'Port Brent', '03231 Richard Lock', '2019-01-24'),
19 (18, 'Myers-Johnson', 'East Johnmouth', '3735 Perez Via', '2018-03-11'),
20 (19, 'Olson, Lambert and Smith', 'Port Teresa', '415 Julia Oval', '2021-12-15'),
21 (20, 'Miller, Burton and Hobbs', 'Stonefort', '928 Robin Fields Suite 679', '2019-01-03'),
22 (21, 'Shelton LLC', 'Frazierfurt', '7756 Morris Ranch', '2023-12-11'),
23 (22, 'Jackson PLC', 'New Joseph', '13248 Evans Keys', '2021-11-18'),
24 (23, 'Edwards and Sons', 'Kyleside', '8426 Tyler Spurs Suite 053', '2021-03-06'),
25 (24, 'Ibarra-Dodson', 'West Megan', '2844 Melissa Branch', '2016-04-29'),
26 (25, 'Young, Scott and Wood', 'Snyderport', '5998 Baker Circles', '2023-02-16'),

```

Results Chart Export

Success. No rows returned

View running queries

```

1 INSERT INTO empleado (id_empleado, nombre, rol, id_rest) VALUES
2 (1, 'Casey Webster', 'Cocinero', 11),
3 (2, 'Michelle Thomas', 'Administrador', 37),
4 (3, 'Timothy Smith', 'Administrador', 28),
5 (4, 'Jacqueline Thomas', 'Mesero', 20),
6 (5, 'Amanda Gilbert', 'Cocinero', 31),
7 (6, 'Vincent Owens', 'Administrador', 38),
8 (7, 'Jane Carter', 'Administrador', 15),
9 (8, 'Jay Gibson', 'Administrador', 36),
10 (9, 'Jose Garcia', 'Administrador', 12),
11 (10, 'Luke Sharp', 'Administrador', 1),
12 (11, 'Kenneth Murphy', 'Mesero', 49),
13 (12, 'Matthew Smith', 'Cocinero', 39),
14 (13, 'Brian Bailey', 'Cocinero', 47),
15 (14, 'Matthew Hill', 'Mesero', 12),
16 (15, 'Joseph Allen', 'Mesero', 29),
17 (16, 'Anthony Stewart', 'Cocinero', 15),
18 (17, 'Kenneth Simon', 'Administrador', 32),
19 (18, 'Daniel Parker', 'Administrador', 40),
20 (19, 'Michael Anderson', 'Cocinero', 25),
21 (20, 'William Curtis', 'Administrador', 6),
22 (21, 'Laura Williams', 'Administrador', 23),
23 (22, 'Matthew Johnson', 'Administrador', 29),
24 (23, 'James Wilson', 'Administrador', 6),
25 (24, 'Kelsey Stevens', 'Repartidor', 21),
26 (25, 'John Rodgers', 'Administrador', 43),

```

Results Chart Export

Success. No rows returned

View running queries

SQL Editor

Search queries...

Templates

Quickstarts

SHARED

FAVORITES

PRIVATE (1)

Restaurant Information

View running queries

```
1 INSERT INTO producto (id_prod, nombre, precio) VALUES
2 (1, 'Get', 13.48),
3 (2, 'Two', 25.83),
4 (3, 'Really', 48.13),
5 (4, 'Community', 7.29),
6 (5, 'Open', 27.07),
7 (6, 'Community', 18.88),
8 (7, 'Everybody', 19.52),
9 (8, 'Standard', 12.07),
10 (9, 'Though', 23.09),
11 (10, 'Less', 44.51),
12 (11, 'Notice', 19.90),
13 (12, 'Effect', 45.23),
14 (13, 'Leg', 11.84),
15 (14, 'Well', 24.81),
16 (15, 'Participant', 30.56),
17 (16, 'Team', 23.93),
18 (17, 'Dog', 24.81),
19 (18, 'Son', 6.34),
20 (19, 'Evidence', 43.97),
21 (20, 'Affect', 37.37),
22 (21, 'Simple', 35.07),
23 (22, 'Local', 27.36),
24 (23, 'Less', 45.91),
25 (24, 'Send', 44.15),
26 (25, 'Ability', 14.88),
```

Results Chart Export

Success. No rows returned

0 row

SQL Editor

Search queries...

Templates

Quickstarts

SHARED

FAVORITES

PRIVATE (1)

Restaurant Information

View running queries

```
1 INSERT INTO pedido (id_pedido, fecha, id_rest, total) VALUES
2 (1, '2025-01-25', 20, 0.0),
3 (2, '2024-12-30', 46, 0.0),
4 (3, '2025-01-12', 35, 0.0),
5 (4, '2024-08-07', 44, 0.0),
6 (5, '2024-10-19', 13, 0.0),
7 (6, '2024-07-10', 37, 0.0),
8 (7, '2024-10-16', 48, 0.0),
9 (8, '2025-01-22', 38, 0.0),
10 (9, '2024-07-22', 20, 0.0),
11 (10, '2024-10-25', 30, 0.0),
12 (11, '2024-07-06', 17, 0.0),
13 (12, '2024-06-16', 28, 0.0),
14 (13, '2025-03-22', 50, 0.0),
15 (14, '2025-02-18', 16, 0.0),
16 (15, '2024-11-25', 50, 0.0),
17 (16, '2024-09-18', 30, 0.0),
18 (17, '2024-05-18', 36, 0.0),
19 (18, '2025-04-14', 46, 0.0),
20 (19, '2024-05-02', 46, 0.0),
21 (20, '2024-08-21', 30, 0.0),
22 (21, '2025-03-09', 9, 0.0),
23 (22, '2024-09-18', 6, 0.0),
24 (23, '2024-09-07', 38, 0.0),
25 (24, '2024-05-31', 19, 0.0),
26 (25, '2025-04-22', 24, 0.0),
```

Results Chart Export

Success. No rows returned

0 row

SQL Editor

Search queries...

Templates

Quickstarts

SHARED

FAVORITES

PRIVATE (1)

Restaurant Information

View running queries

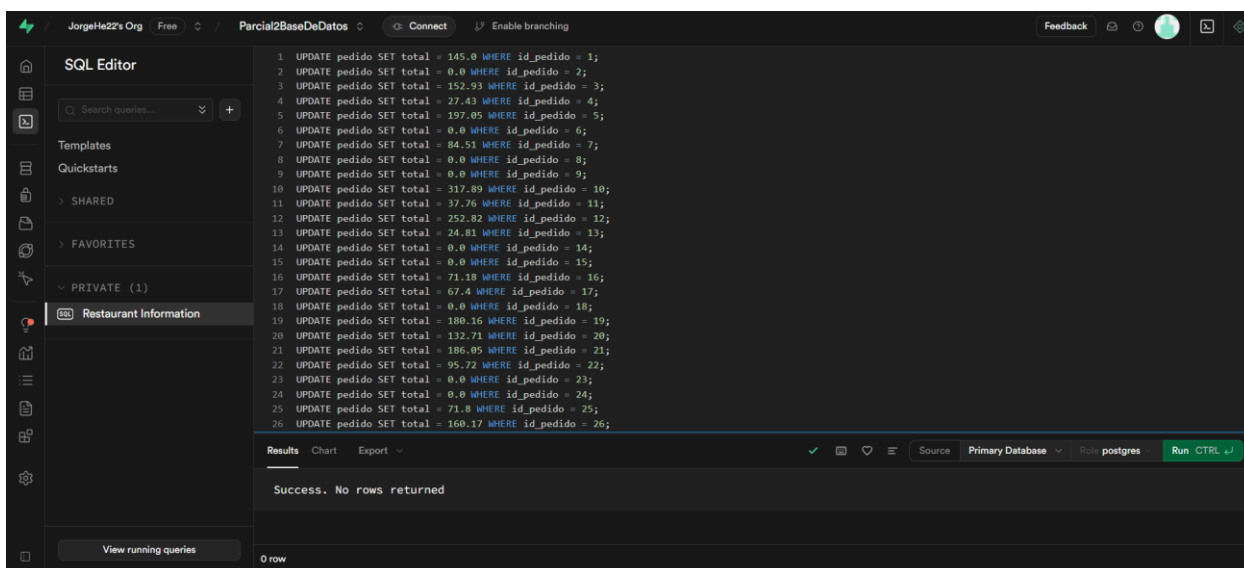
```
1 INSERT INTO detalle_pedido (id_detalle, id_pedido, id_prod, cantidad, subtotal) VALUES
2 (1, 10, 38, 3, 84.39),
3 (2, 37, 5, 1, 27.07),
4 (3, 26, 38, 5, 140.65),
5 (4, 30, 17, 2, 49.62),
6 (5, 29, 39, 3, 106.77),
7 (6, 1, 46, 4, 145.0),
8 (7, 33, 43, 4, 42.16),
9 (8, 30, 36, 2, 29.12),
10 (9, 21, 19, 3, 131.91),
11 (10, 31, 44, 5, 102.95),
12 (11, 10, 21, 2, 70.14),
13 (12, 7, 30, 3, 84.51),
14 (13, 5, 42, 4, 115.72),
15 (14, 46, 41, 5, 182.45),
16 (15, 42, 10, 3, 19.02),
17 (16, 10, 6, 1, 18.88),
18 (17, 34, 25, 1, 14.88),
19 (18, 36, 22, 4, 109.44),
20 (19, 10, 47, 4, 72.24),
21 (20, 10, 47, 4, 72.24),
22 (21, 21, 5, 2, 54.14),
23 (22, 17, 1, 5, 67.4),
24 (23, 4, 47, 1, 18.06),
25 (24, 19, 48, 4, 180.16),
26 (25, 44, 18, 4, 25.36),
```

Results Chart Export

Success. No rows returned

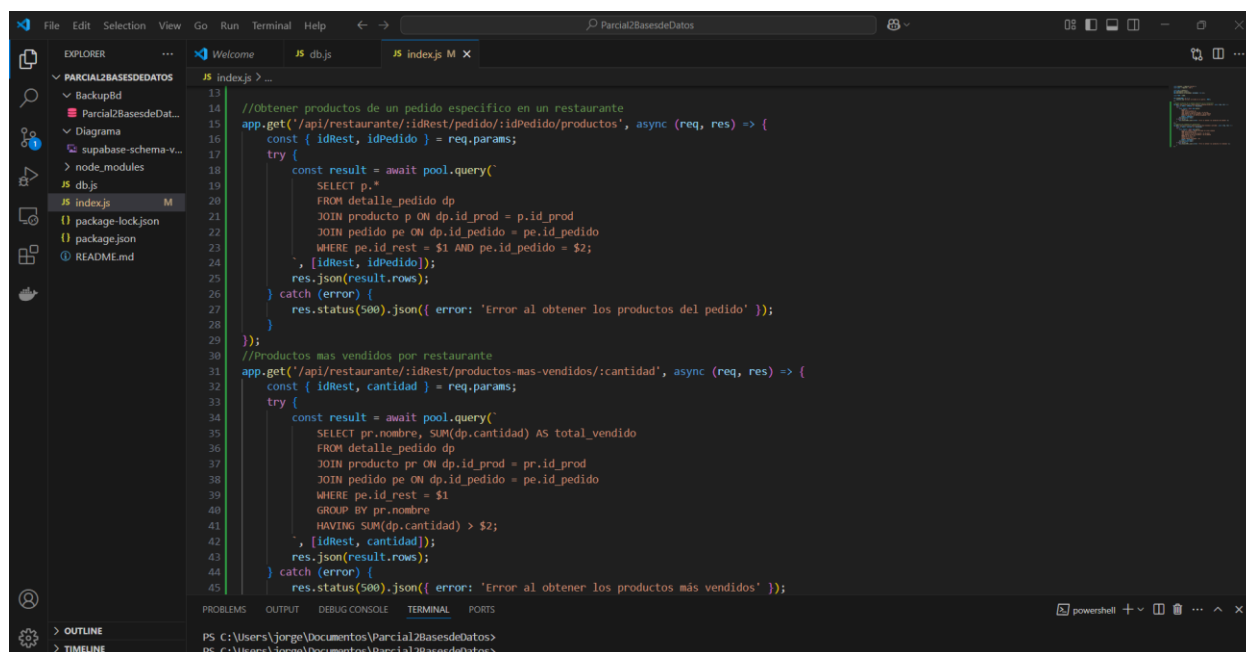
0 row

Ahora vamos a actualizar los totales de pedidos para sincronizar el campo “total” en la tabla “pedido” con la suma de los subtotales calculados en la tabla “detalle_pedido”



8. Consultas nativas para cada tabla:

- Tabla restaurante:



```

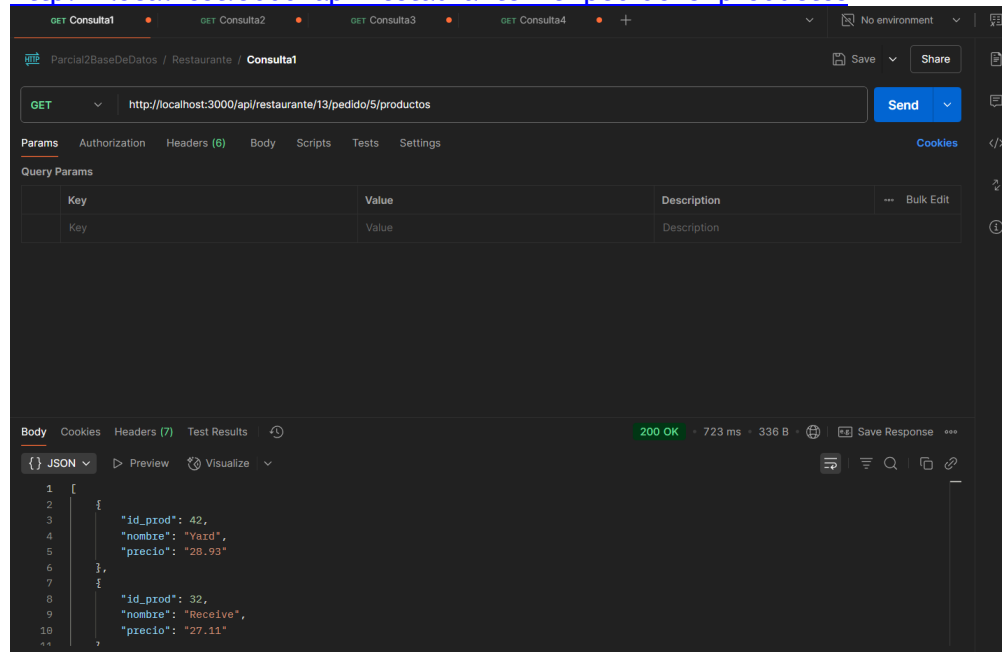
47 app.get('/api/restaurante/:idRest/productos-mas-vendidos/:cantidad', async (req, res) => {
48 });
49 //Total de ventas por restaurante
50 app.get('/api/restaurante/ventas', async (req, res) => {
51   try {
52     const result = await pool.query(
53       SELECT r.id_rest, r.nombre, SUM(pe.total) AS total_ventas
54       FROM restaurante r
55       JOIN pedido pe ON r.id_rest = pe.id_rest
56       GROUP BY r.id_rest, r.nombre;
57     );
58     res.json(result.rows);
59   } catch (error) {
60     res.status(500).json({ error: 'Error al obtener el total de ventas' });
61   }
62 });
63 //Pedidos por fecha y restaurante
64 app.get('/api/restaurante/:idRest/pedidos/:fecha', async (req, res) => {
65   const { idRest, fecha } = req.params;
66   try {
67     const result = await pool.query(
68       SELECT *
69       FROM pedido
70       WHERE id_rest = $1 AND fecha = $2;
71     );
72     res.json(result.rows);
73   } catch (error) {
74     res.status(500).json({ error: 'Error al obtener pedidos por fecha' });
75   }
76 });
77 //Empleados por rol en un restaurante
78 app.get('/api/restaurante/:idRest/empleados/:rol', async (req, res) => {
79   const { idRest, rol } = req.params;
80   try {
81     const result = await pool.query(
82       SELECT *
83       FROM empleado
84       WHERE id_rest = $1 AND rol = $2;
85     );
86     res.json(result.rows);
87   } catch (error) {
88     res.status(500).json({ error: 'Error al obtener empleados por rol' });
89   }
90 });

```

Consultas Nativas:

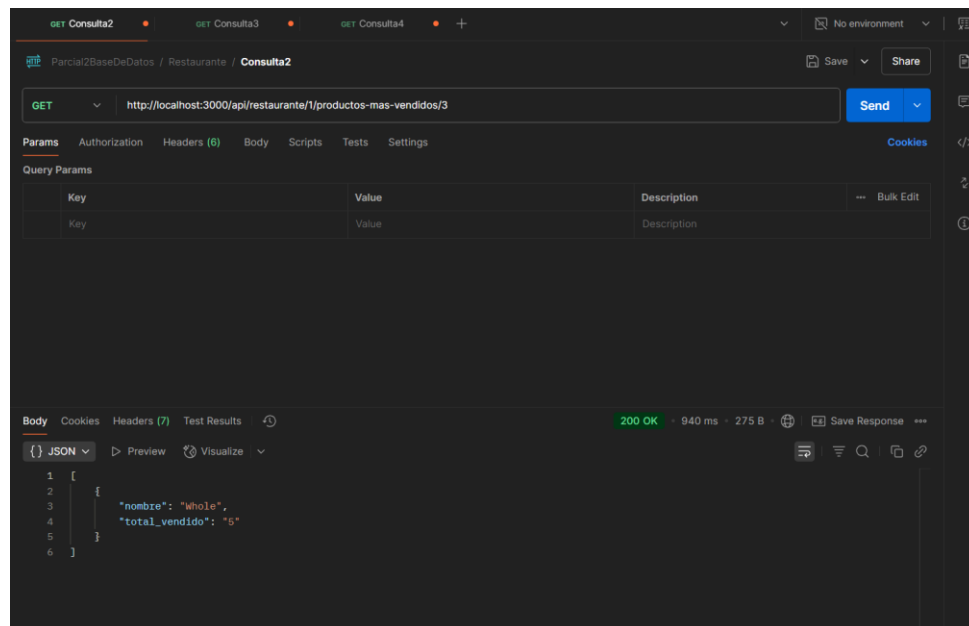
Obtener productos de un pedido específico en un restaurante:

- <http://localhost:3000/api/restaurante/13/pedido/5/productos>



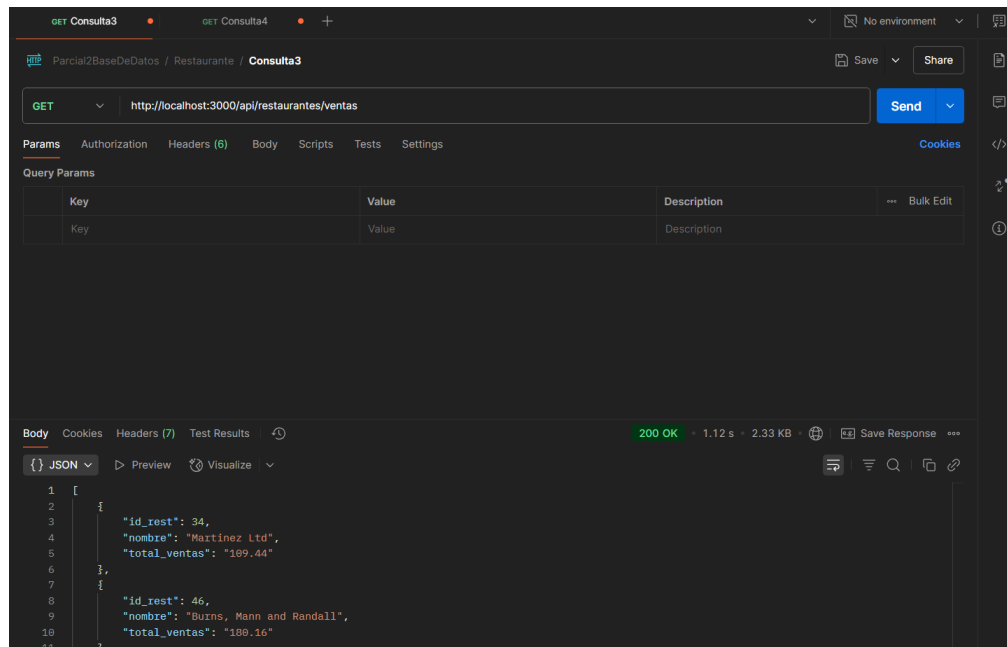
Productos más vendidos por restaurante:

- <http://localhost:3000/api/restaurante/1/productos-mas-vendidos/3>



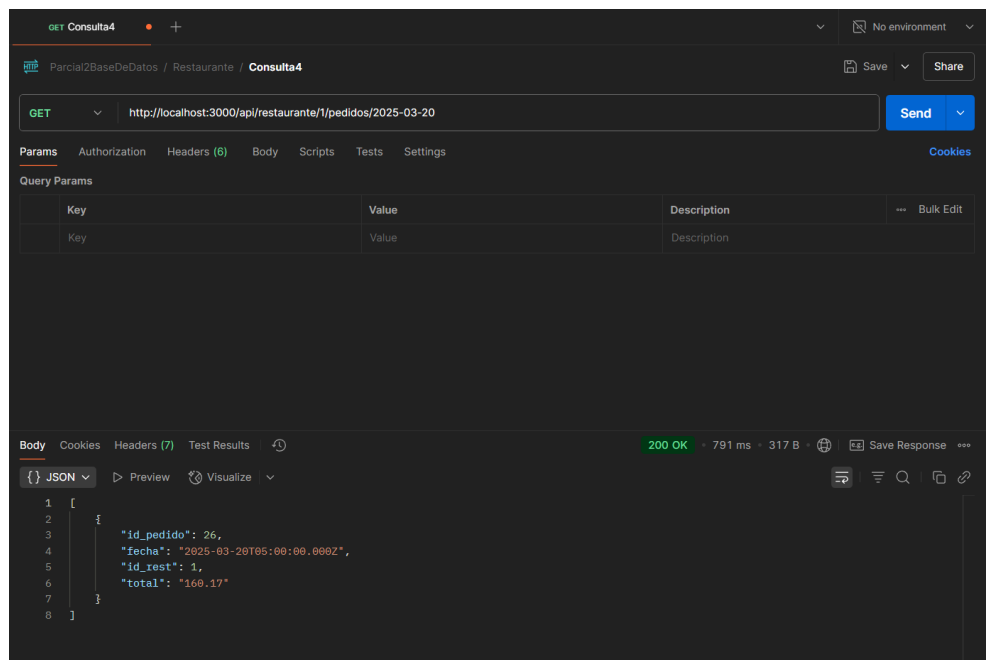
Total de ventas por restaurante:

- <http://localhost:3000/api/restaurantes/ventas>



Pedidos por fecha y restaurante:

- <http://localhost:3000/api/restaurante/1/pedidos/2025-03-20>



Empleados por rol en un restaurante :

- <http://localhost:3000/api/restaurante/11/empleados/Cocinero>

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:3000/api/restaurante/11/empleados/Cocinero`
- Status:** 200 OK
- Response Time:** 1.05 s
- Response Size:** 309 B
- Response Body (JSON):**

```
[
  {
    "id_empleado": 1,
    "nombre": "Casey Webster",
    "rol": "Cocinero",
    "id_rest": 11
  }
]
```

The interface also includes tabs for Params, Authorization, Headers (6), Body, Scripts, Tests, and Settings. The Body tab is currently selected, showing the JSON response.