

Trabajo clase

Estudiante

Carlos Andrés Castellanos Arias

Id:667626

Sesto Semestre

Profesor:

William Alexander Matallana Porras

UNIMINUTO - CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS

ZIPAQUIRÁ

Bases de datos masivas

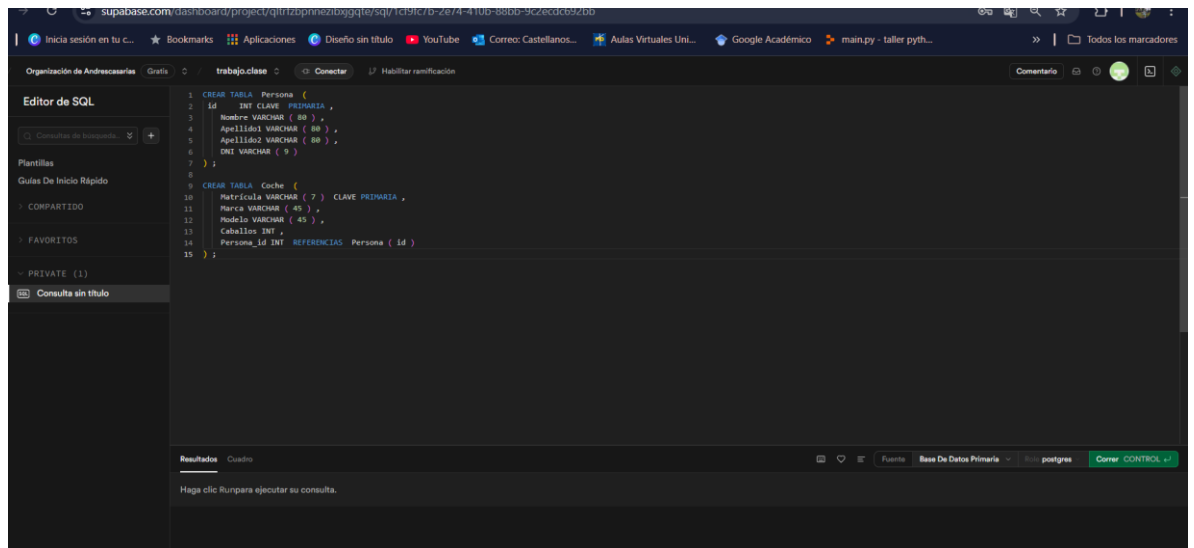
2025

Introducción

Objetivo

Creación tablas

Creamos las tablas en supabase



The screenshot shows the Supabase SQL Editor interface. The left sidebar contains the 'Editor de SQL' section with a search bar and a list of templates. The main area displays the SQL code for creating two tables: 'Persona' and 'Coche'. The 'Persona' table has columns for 'Id' (primary key), 'Nombre', 'Apellido1', 'Apellido2', and 'DNI'. The 'Coche' table has columns for 'Matricula' (primary key), 'Marca', 'Modelo', 'Caballos', and 'Persona_id' (foreign key to 'Persona'). The bottom of the interface shows the 'Resultados' section with a message: 'Haga clic Run para ejecutar su consulta.'

```
1 CREATE TABLE Persona (
2   Id INT CLAVE PRIMARIA ,
3   Nombre VARCHAR ( 80 ) ,
4   Apellido1 VARCHAR ( 80 ) ,
5   Apellido2 VARCHAR ( 80 ) ,
6   DNI VARCHAR ( 9 )
7 ) ;
8
9 CREATE TABLE Coche (
10  Matricula VARCHAR ( 7 ) CLAVE PRIMARIA ,
11  Marca VARCHAR ( 45 ) ,
12  Modelo VARCHAR ( 45 ) ,
13  Caballos INT ,
14  Persona_id INT REFERENCIAS Persona ( Id )
15 ) ;
```

Ingresos de datos por ia para cada tabla

CREAMOS TABLA Persona (id INT PRIMARY KEY, Nombre VARCHAR(80), Apellido1 VARCHAR(80), Apellido2 VARCHAR(80), DNI VARCHAR(9));
 CREAMOS TABLA Coche (Matricula VARCHAR(7) PRIMARY KEY, Marca VARCHAR(45), Modelo VARCHAR(45), Caballos INT, Persona_id INT REFERENCIAS Persona(id)); por favor dame 100 registros de cada tabla

Aquí tienes un ejemplo de cómo generar 100 registros de prueba para las tablas Persona y Coche que creaste. Puedes ejecutarlo en PostgreSQL o usarlo como base para poblar tu base de datos. Los datos son aleatorios y simulados.

1. INSERTOS para Persona (100 registros)

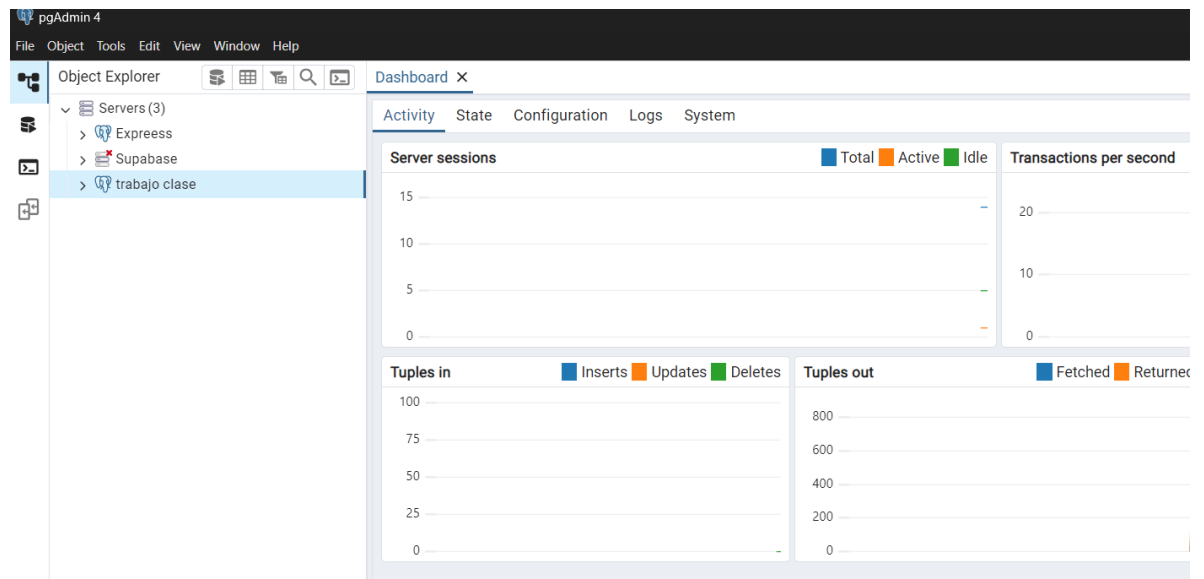
```

SQL
-- Copiar -- Editar

INSERT INTO Persona (id, Nombre, Apellido1, Apellido2, DNI) VALUES
(1, 'Luis', 'García', 'Pérez', '12345678A'),
(2, 'Ana', 'Martínez', 'López', '23456789B'),
(3, 'Carlos', 'Rodríguez', 'Gómez', '34567890C'),
(4, 'Marta', 'Sánchez', 'Díaz', '45678901D'),
(5, 'Pedro', 'Fernández', 'Ruiz', '56789012E'),
(6, 'Lucía', 'Moreno', 'Romero', '67890123F'),
(7, 'Jorge', 'Alonso', 'Serrano', '78901234G'),
(8, 'Sofía', 'Dominguez', 'Navarro', '89012345H'),
(9, 'Raúl', 'Torres', 'Gill', '90123456I'),
(10, 'Elena', 'Vargas', 'Cano', '01234567J'),
-- Continúa con los registros 11 al 100:

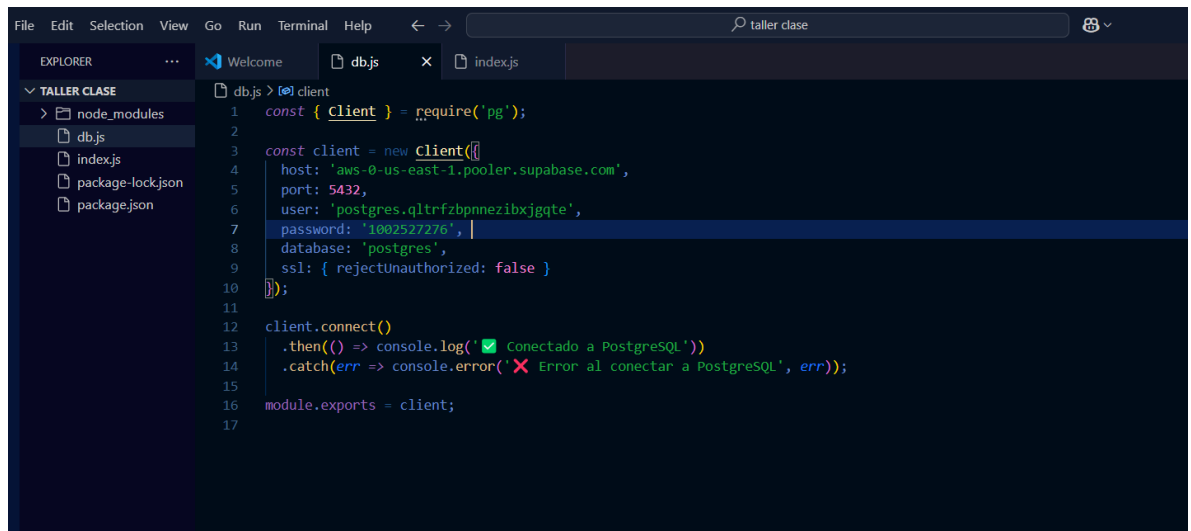
```

Conexión a pgadmin



Creación de la conexión entre supabase con visual studio

Donde tomamos los datos que tenemos en supabase para la conexión para que se conecten entre ellas



```
File Edit Selection View Go Run Terminal Help
taller clase

EXPLORER
TALLER CLASE
  node_modules
  db.js
  index.js
  package-lock.json
  package.json

db.js > client
1  const { Client } = require('pg');
2
3  const client = new Client({
4    host: 'aws-0-us-east-1.pooler.supabase.com',
5    port: 5432,
6    user: 'postgres.qltrfzbpnnezibxjgqte',
7    password: '1002527276',
8    database: 'postgres',
9    ssl: { rejectUnauthorized: false }
10  });
11
12  client.connect()
13    .then(() => console.log('✅ Conectado a PostgreSQL'))
14    .catch(err => console.error('❌ Error al conectar a PostgreSQL', err));
15
16  module.exports = client;
17
```

Importamos la librerías

```
// Importar librerías
const express = require('express');
const connection = require('./db');
```

Después creamos la app express y la middleware que nos permite almacenar datos en formato JSON y los datos por la url

```
// Middleware para parsear JSON y datos por URL
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

```

// Puerto del servidor
const PORT = 5000;

// RUTAS DE PRUEBA
app.get('/api/prueba', (req, res) => {
  res.send('Estoy respondiendo por la API');
});

app.get('/api/prueba2', (req, res) => {
  res.status(200).json({
    message: ' La API funciona bien',
    port: PORT,
    status: ' fue exitoso'
  });
});

```

```

// API PERSONA

// Guardar persona
app.post('/api/persona', (req, res) => {
  const { id, nombre, apellido1, apellido2, dni } = req.body;
  const query = 'INSERT INTO persona (id, nombre, apellido1, apellido2, dni) VALUES ($1, $2, $3, $4, $5)';

  connection.query(query, [id, nombre, apellido1, apellido2, dni], (error, result) => {
    if (error) {
      res.status(500).json({
        success: false,
        message: 'Error creando la persona',
        details: error.message
      });
    } else {
      res.status(201).json({ id, nombre, apellido1, apellido2, dni });
    }
  });
});

```

```

// Obtener todas las personas
app.get('/api/persona', (req, res) => {
  const query = 'SELECT * FROM persona';

  connection.query(query, (error, result) => {
    if (error) {
      res.status(500).json({
        success: false,
        message: "Error al recuperar los datos",
        details: error.message
      });
    } else {
      res.status(200).json({
        success: true,
        message: "Datos de la tabla persona",
        data: result.rows
      });
    }
  });
});
});

```

```

// Eliminar persona por ID
app.delete('/api/persona/:id', (req, res) => {
  const { id } = req.params;
  const query = 'DELETE FROM persona WHERE id = $1';

  connection.query(query, [id], (error, result) => {
    if (error) {
      res.status(500).json({
        success: false,
        message: "Error al eliminar la persona",
        details: error.message
      });
    } else if (result.rowCount === 0) {
      res.status(404).json({
        success: false,
        message: `No existe ninguna persona con id ${id}`,
      });
    } else {
      res.status(200).json({
        success: true,
        message: "Persona eliminada correctamente"
      });
    }
  });
});

```

```

94
95 // Actualizar persona por ID
96 app.put('/api/persona/:id', (req, res) => {
97   const { id } = req.params;
98   const { nombre, apellido1, apellido2, dni } = req.body;
99   const query = 'UPDATE persona SET nombre = $1, apellido1 = $2, apellido2 = $3, dni = $4 WHERE id = $5';
100
101   connection.query(query, [nombre, apellido1, apellido2, dni, id], (error, result) => {
102     if (error) {
103       res.status(500).json({
104         success: false,
105         message: 'Error al actualizar la persona',
106         details: error.message
107       });
108     } else if (result.rowCount === 0) {
109       res.status(404).json({
110         success: false,
111         message: `No se encontró ninguna persona con el ID ${id}`
112       });
113     } else {
114       res.status(200).json({
115         success: true,
116         message: 'Persona actualizada correctamente',
117         updated: {
118           id,
119           nombre,
120           apellido1,
121           apellido2,
122           dni
123         }
124       });
125     }
126   });
127 }

```

```

// API COCHE

// Crear coche
app.post('/api/coche', (req, res) => {
  const { matricula, marca, modelo, caballos, persona_id } = req.body;
  const query = 'INSERT INTO coche (matricula, marca, modelo, caballos, persona_id) VALUES ($1, $2, $3, $4, $5)';

  connection.query(query, [matricula, marca, modelo, caballos, persona_id], (error, result) => {
    if (error) {
      res.status(500).json({
        success: false,
        message: 'Error al crear el coche',
        details: error.message
      });
    } else {
      res.status(201).json({ matricula, marca, modelo, caballos, persona_id });
    }
  });
});

```

```

// Obtener todos los coches
app.get('/api/coche', (req, res) => {
  const query = 'SELECT * FROM coche';

  connection.query(query, (error, result) => {
    if (error) {
      res.status(500).json({
        success: false,
        message: "Error al recuperar los datos de coches",
        details: error.message
      });
    } else {
      res.status(200).json({
        success: true,
        message: "Datos de la tabla coche",
        data: result.rows
      });
    }
  });
});
});

// Eliminar coche por matrícula

```

```

// Eliminar coche por matrícula
app.delete('/api/coche/:matricula', (req, res) => {
  const { matricula } = req.params;
  const query = 'DELETE FROM coche WHERE matricula = $1';

  connection.query(query, [matricula], (error, result) => {
    if (error) {
      res.status(500).json({
        success: false,
        message: "Error al eliminar el coche",
        details: error.message
      });
    } else if (result.rowCount === 0) {
      res.status(404).json({
        success: false,
        message: `No se encuentra ningún coche con matrícula ${matricula}`,
      });
    } else {
      res.status(200).json({
        success: true,
        message: "Coche fue eliminado exitosamente "
      });
    }
  });
});
});

```



```

195 // Actualizar coche por matricula
196
197 app.put('/api/coche/:matricula', (req, res) => {
198   const { matricula } = req.params;
199   const { marca, modelo, caballos, persona_id } = req.body;
200   const query = 'UPDATE coche SET marca = $1, modelo = $2, caballos = $3, persona_id = $4 WHERE matricula = $5';
201
202   connection.query(query, [marca, modelo, caballos, persona_id, matricula], (error, result) => {
203     if (error) {
204       res.status(500).json({
205         success: false,
206         message: 'Error al actualizar el coche',
207         details: error.message
208       });
209     } else if (result.rowCount === 0) {
210       res.status(404).json({
211         success: false,
212         message: 'No se encontró ningún coche con la matricula que nos das ${matricula}'
213       });
214     } else {
215       res.status(200).json({
216         success: true,
217         message: 'Coche actualizado correctamente',
218         updated: {
219           matricula,
220           marca,
221           modelo,
222           caballos,
223           persona_id
224         }
225       });
226     }
227   });
228 }
229 );

```