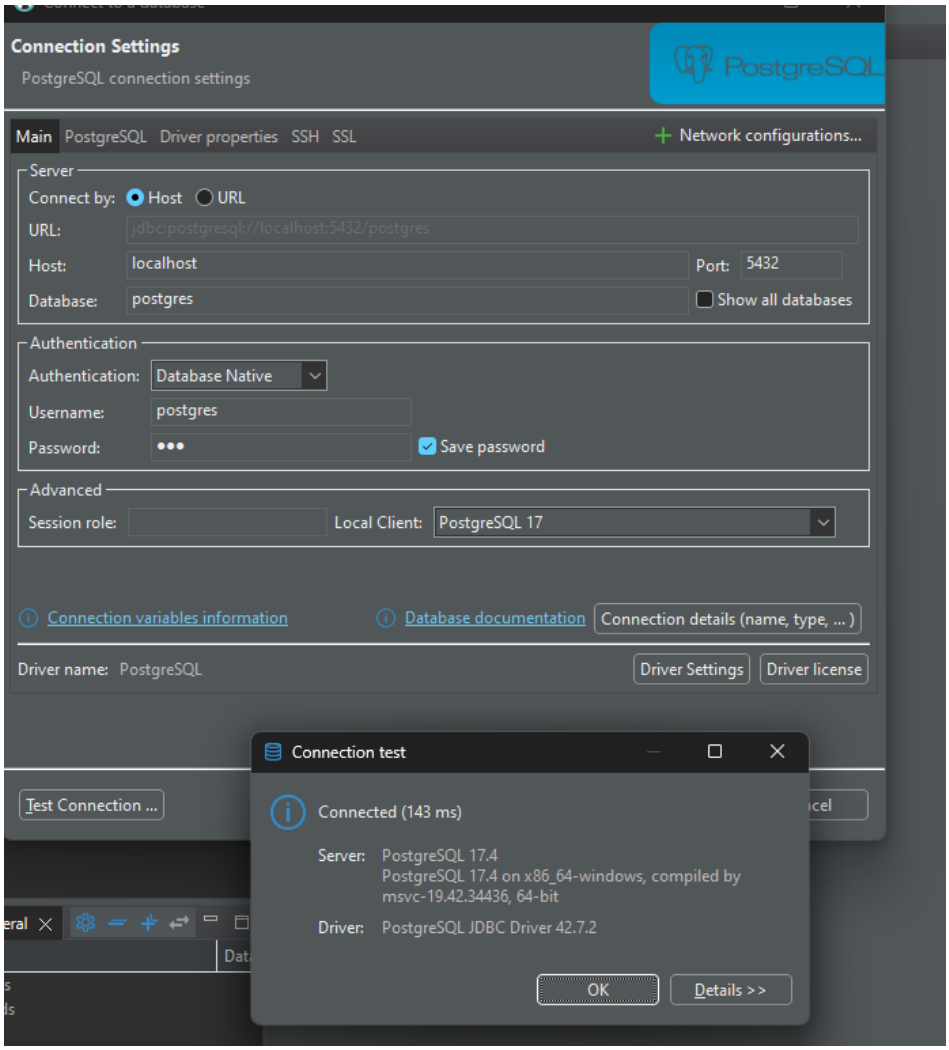
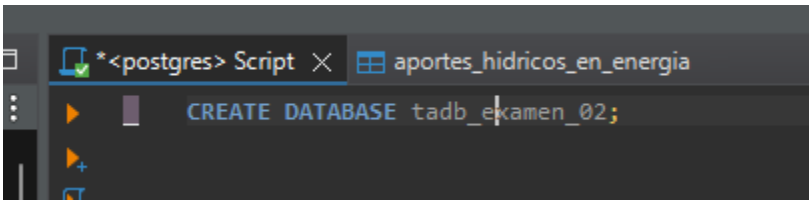
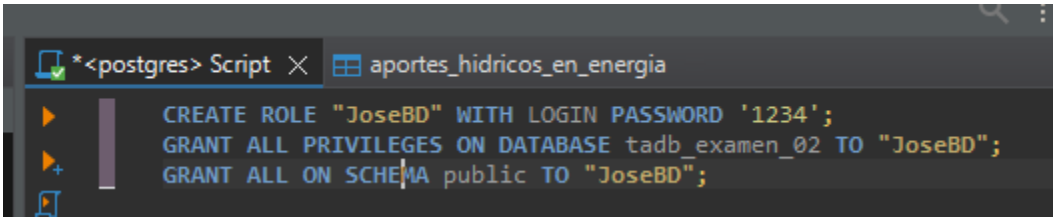
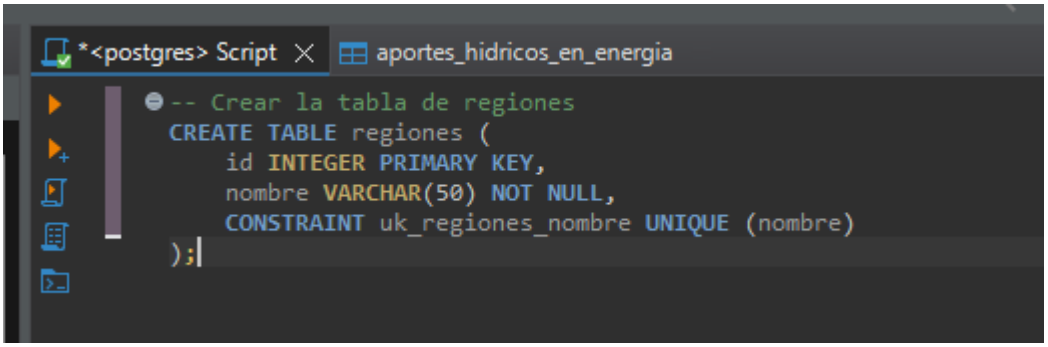
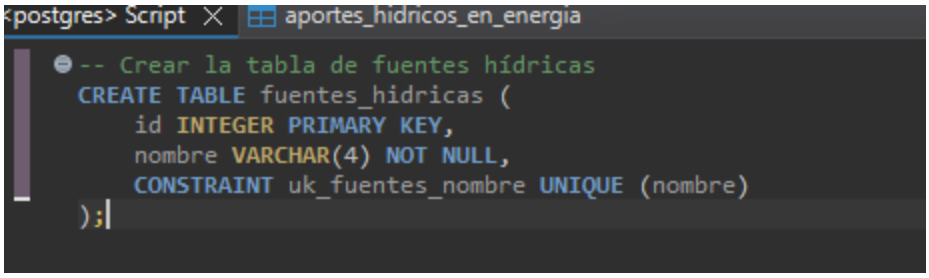
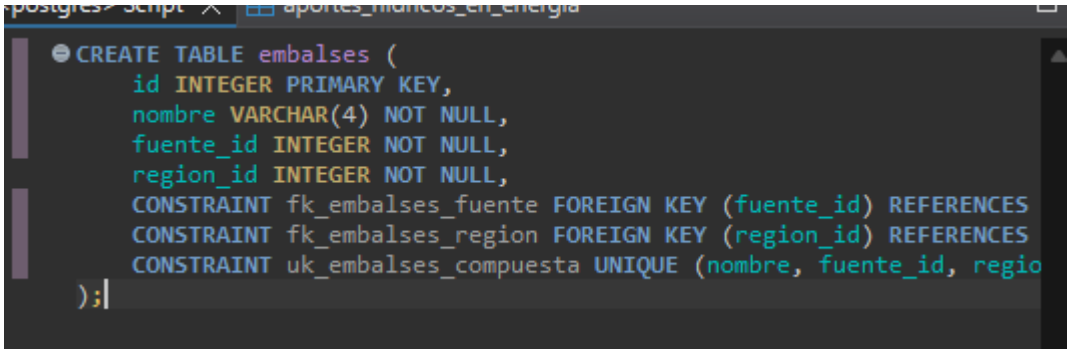

Connection Settings
PostgreSQL connection settings
PostgreSQL
Main PostgreSQL Driver properties SSH SSL
Network configurations...
Server
Connect by: Host URL
URL: jdbc:postgresql://localhost:5432/postgres
Host: localhost
Port: 5432
Database: postgres
Show all databases
Authentication
Authentication: Database Native
Username: postgres
Password: •••
Save password
Advanced
Session role:
Local Client: PostgreSQL 17
Connection variables information
Database documentation
Connection details (name, type, ...)
Driver name: PostgreSQL
Driver Settings
Driver license
Test Connection ...
Connection test
Connected (143 ms)
Server: PostgreSQL 17.4
PostgreSQL 17.4 on x86_64-windows, compiled by msvc-19.42.34436, 64-bit
Driver: PostgreSQL JDBC Driver 42.7.2
OK
Details >>

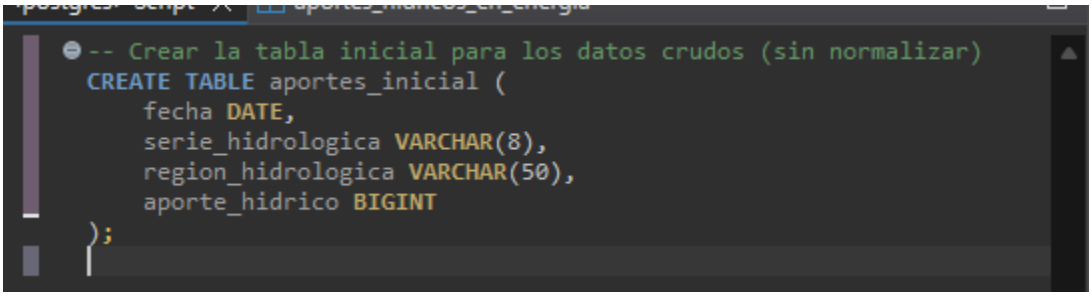

*<postgres> Script
aportes_hidricos_en_energia
CREATE DATABASE tadb_examen_02;

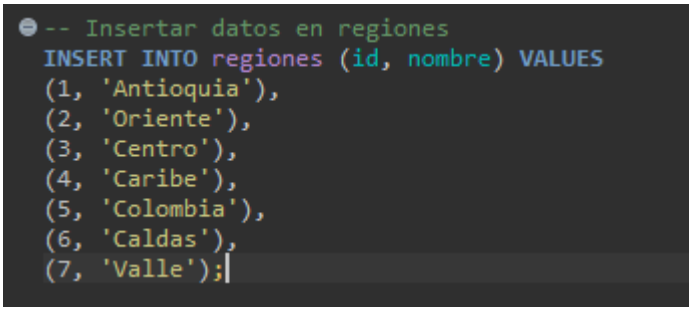

*<postgres> Script
aportes_hidricos_en_energia
CREATE ROLE "JoseBD" WITH LOGIN PASSWORD '1234';
GRANT ALL PRIVILEGES ON DATABASE tadb_examen_02 TO "JoseBD";
GRANT ALL ON SCHEMA public TO "JoseBD";

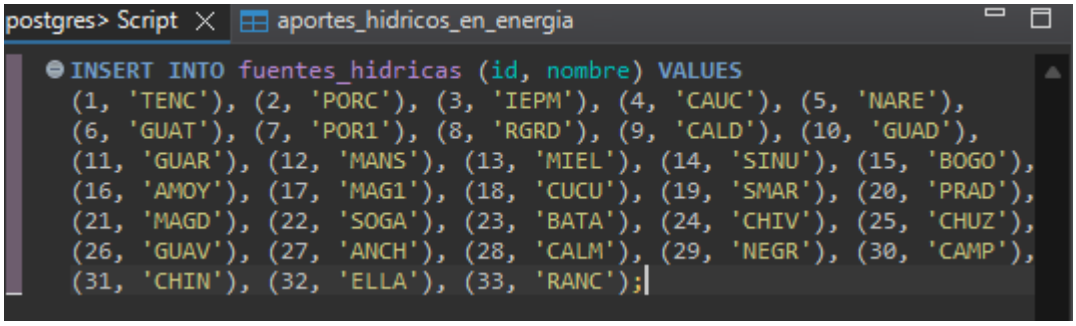

*<postgres> Script
aportes_hidricos_en_energia
-- Crear la tabla de regiones
CREATE TABLE regiones (
id INTEGER PRIMARY KEY,
nombre VARCHAR(50) NOT NULL,
CONSTRAINT uk_regiones_nombre UNIQUE (nombre)
);


postgres> Script
aportes_hidricos_en_energia
-- Crear la tabla de fuentes hídricas
CREATE TABLE fuentes_hidricas (
id INTEGER PRIMARY KEY,
nombre VARCHAR(4) NOT NULL,
CONSTRAINT uk_fuentes_nombre UNIQUE (nombre)
);


CREATE TABLE embalses (
id INTEGER PRIMARY KEY,
nombre VARCHAR(4) NOT NULL,
fuente_id INTEGER NOT NULL,
region_id INTEGER NOT NULL,
CONSTRAINT fk_embalses_fuente FOREIGN KEY (fuente_id) REFERENCES
CONSTRAINT fk_embalses_region FOREIGN KEY (region_id) REFERENCES
CONSTRAINT uk_embalses_compuesta UNIQUE (nombre, fuente_id, regio
);


-- Crear la tabla inicial para los datos crudos (sin normalizar)
CREATE TABLE aportes_inicial (
fecha DATE,
serie_hidrologica VARCHAR(8),
region_hidrologica VARCHAR(50),
aporte_hidrico BIGINT
);


-- Insertar datos en regiones
INSERT INTO regiones (id, nombre) VALUES
(1, 'Antioquia'),
(2, 'Oriente'),
(3, 'Centro'),
(4, 'Caribe'),
(5, 'Colombia'),
(6, 'Caldas'),
(7, 'Valle');


postgres> Script
aportes_hidricos_en_energia
INSERT INTO fuentes_hidricas (id, nombre) VALUES
(1, 'TENC'), (2, 'PORC'), (3, 'IEPM'), (4, 'CAUC'), (5, 'NARE'),
(6, 'GUAT'), (7, 'POR1'), (8, 'RGRD'), (9, 'CALD'), (10, 'GUAD'),
(11, 'GUAR'), (12, 'MANS'), (13, 'MIEL'), (14, 'SINU'), (15, 'BOGO'),
(16, 'AMOY'), (17, 'MAG1'), (18, 'CUCU'), (19, 'SMAR'), (20, 'PRAD'),
(21, 'MAGD'), (22, 'SOGA'), (23, 'BATA'), (24, 'CHIV'), (25, 'CHUZ'),
(26, 'GUAV'), (27, 'ANCH'), (28, 'CALM'), (29, 'NEGR'), (30, 'CAMP'),
(31, 'CHIN'), (32, 'ELLA'), (33, 'RANC');

```sql
INSERT INTO embalses (id, nombre, fuente_id, region_id) VALUES
(1, 'BOCA', 1, 1), (2, 'CLLR', 2, 1), (3, 'DESV', 3, 1),
(4, 'ITUA', 4, 1), (5, 'MIRF', 1, 1), (6, 'PENO', 5, 1),
(7, 'PLAY', 6, 1), (8, 'PP-2', 7, 1), (9, 'PP-3', 2, 1),
(10, 'PUNC', 6, 1), (11, 'QUBR', 8, 1), (12, 'RGR2', 8, 1),
(13, 'SLOR', 5, 1), (14, 'SMIG', 9, 1), (15, 'TRON', 10, 1),
(16, 'DESV', 11, 6), (17, 'DESV', 12, 6), (18, 'PTEH', 13, 6),
(19, 'URR1', 14, 4), (20, 'ALIC', 15, 3), (21, 'AMOY', 16, 3),
(22, 'BETA', 17, 3), (23, 'CUCU', 18, 3), (24, 'DESV', 19, 3),
(25, 'EMBA', 20, 3), (26, 'QUIM', 21, 3), (27, 'SOGA', 22, 3),
(28, 'DESV', 23, 2), (29, 'DESV', 24, 2), (30, 'EMBA', 25, 2),
(31, 'EMBA', 26, 2), (32, 'ESME', 23, 2), (33, 'ALTO', 27, 7),
(34, 'BAJO', 27, 7), (35, 'CAL1', 28, 7), (36, 'FLR2', 4, 7),
(37, 'SALV', 4, 7), (38, 'ESCM', 29, 1), (39, 'CAME', 30, 6),
(40, 'CAME', 31, 6), (41, 'ESME', 30, 6), (42, 'ESTR', 32, 6),
(43, 'SANF', 33, 6), (44, 'GUAV', 26, 2);
```

```sql
CREATE TABLE aportes (
    id SERIAL PRIMARY KEY,
    fecha DATE NOT NULL,
    embalse_id INTEGER NOT NULL,
    region_id INTEGER NOT NULL,
    aporte_hidrico BIGINT NOT NULL,
    CONSTRAINT fk_aportes_embalse FOREIGN KEY (embalse_id) REFERENCES
    CONSTRAINT fk_aportes_region FOREIGN KEY (region_id) REFERENCES r
);
```

```sql
SELECT COUNT(*) FROM aportes_inicial;
SELECT COUNT(*) FROM aportes;
SELECT * FROM aportes LIMIT 10;
```

lts 1 | Results 1 (2) | aportes 1 (3)

CT COUNT(*) FROM a | *Data filter is not supported*

123 id	fecha	123 embalse_id	123
1	2023-01-01	1	1
2	2023-01-01	2	1
3	2023-01-01	3	1
4	2023-01-01	4	1
5	2023-01-01	5	1
6	2023-01-01	6	1
7	2023-01-01	7	1
8	2023-01-01	8	1
9	2023-01-01	9	1
10	2023-01-01	10	1

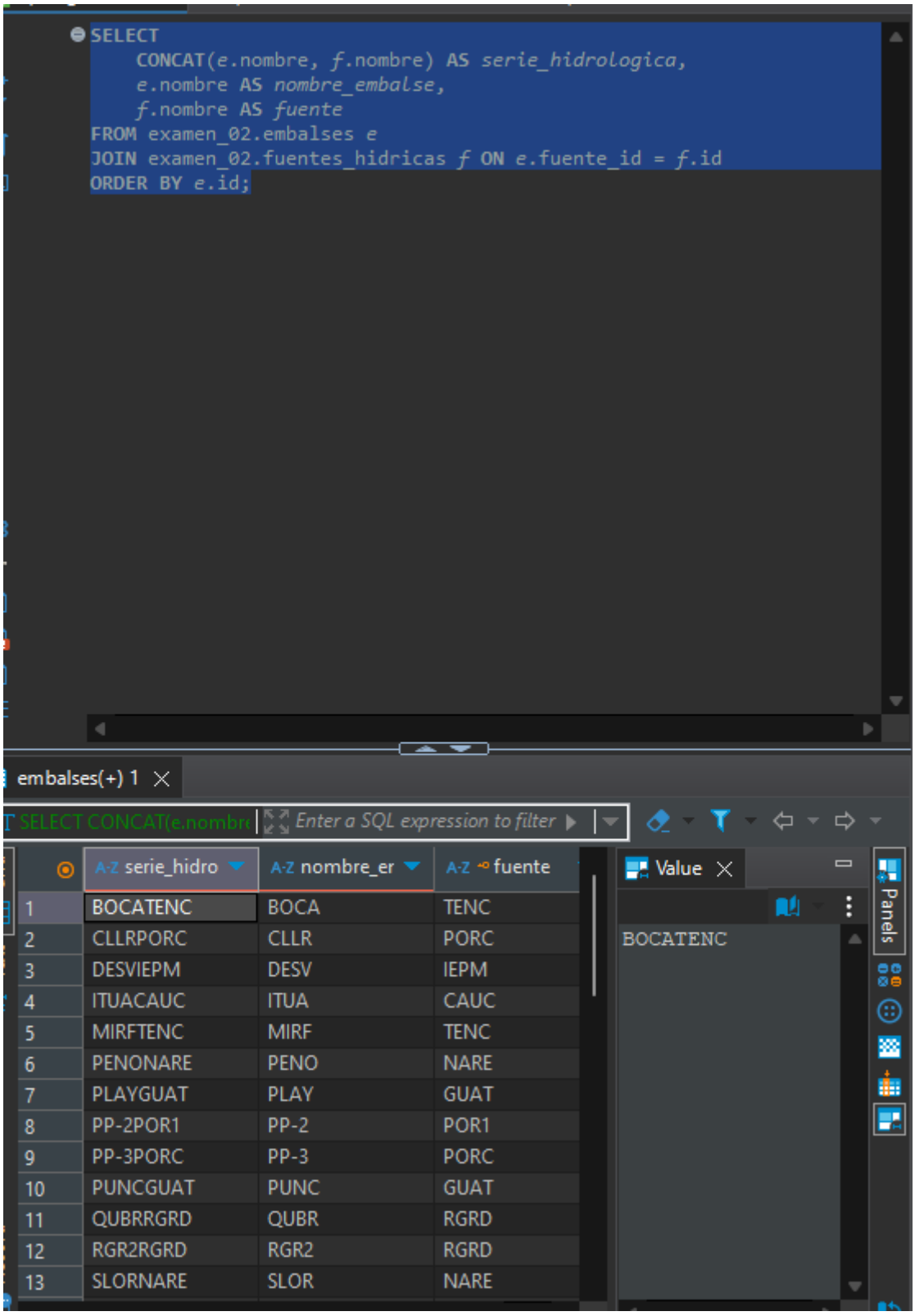
```sql
SELECT COUNT(*) FROM aportes;
SELECT * FROM aportes LIMIT 10;
```

sults 1 | aportes 1 (2) ×

LECT COUNT(*) FROM a | *Data filter is not supported*

	123 id	fecha	123 embalse_id	123
	1	2023-01-01	1	1
	2	2023-01-01	2	1
	3	2023-01-01	3	1
	4	2023-01-01	4	1
	5	2023-01-01	5	1
	6	2023-01-01	6	1
	7	2023-01-01	7	1
	8	2023-01-01	8	1
	9	2023-01-01	9	1
0	10	2023-01-01	10	1

```sql
INSERT INTO examen_02.aportes (fecha, embalse_id, region_id, aporte_h
SELECT
    ai.fecha,
    e.id AS embalse_id,
    r.id AS region_id,
    ai.aporte_hidrico
FROM examen_02.aportes_inicial ai
JOIN examen_02.embalses e ON
    SUBSTRING(ai.serie_hidrologica FROM 1 FOR 4) = e.nombre
    AND (
        SELECT f.id
        FROM examen_02.fuentes_hidricas f
        WHERE f.nombre = SUBSTRING(ai.serie_hidrologica FROM 5 FOR 4)
    ) = e.fuente_id
JOIN examen_02.regiones r ON ai.region_hidrologica = r.nombre;
```

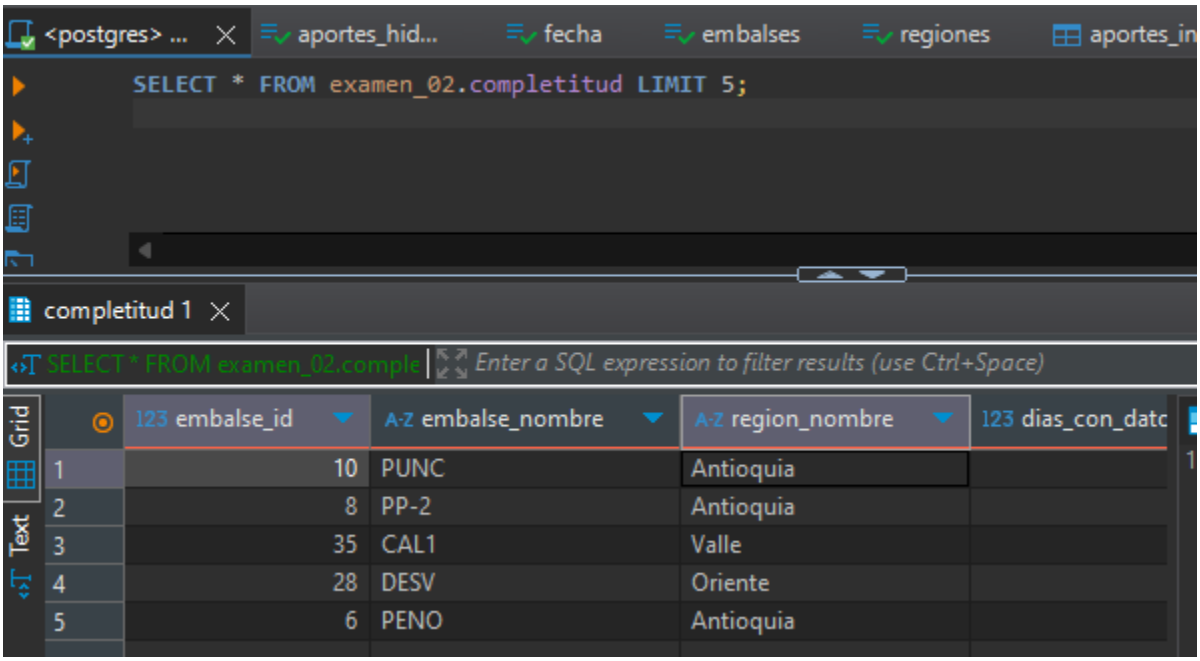

SELECT
CONCAT(e.nombre, f.nombre) AS serie_hidrologica,
e.nombre AS nombre_embalse,
f.nombre AS fuente
FROM examen_02.embalses e
JOIN examen_02.fuentes_hidricas f ON e.fuente_id = f.id
ORDER BY e.id;
embalses(+) 1
SELECT CONCAT(e.nombre
Enter a SQL expression to filter
serie_hidro
nombre_er
fuente
1 BOCATENC BOCA TENC
2 CLLRPORC CLLR PORC
3 DESVIEPM DESV IEPM
4 ITUACAUC ITUA CAUC
5 MIRFTENC MIRF TENC
6 PENONARE PENO NARE
7 PLAYGUAT PLAY GUAT
8 PP-2POR1 PP-2 POR1
9 PP-3PORC PP-3 PORC
10 PUNCGUAT PUNC GUAT
11 QUBRRGRD QUBR RGRD
12 RGR2RGRD RGR2 RGRD
13 SLORNARE SLOR NARE
Value
BOCATENC
Panels

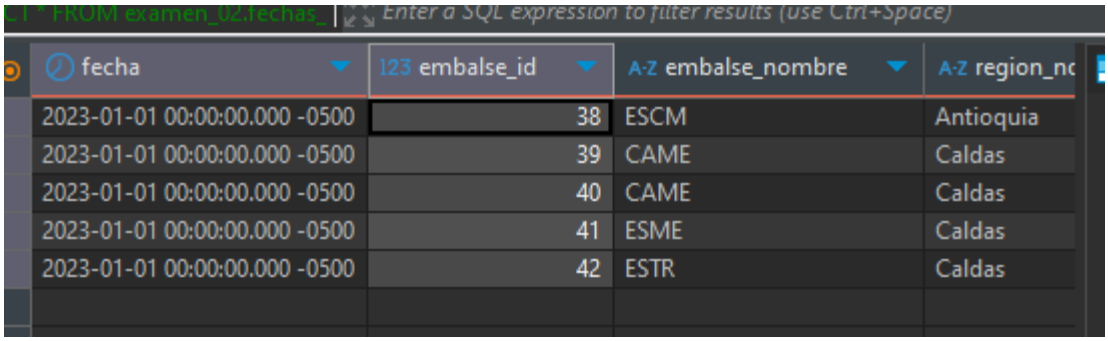
```sql
-- Crear una tabla de calendario con todas las fechas entre 2023-01-01 y 2024-12-31
CREATE TABLE calendario AS
SELECT generate_series('2023-01-01'::date, '2024-12-31'::date, '1 day'::interval) AS fecha;

-- Crear una tabla con todas las combinaciones esperadas de fechas y embalses
CREATE TABLE combinaciones_esperadas AS
SELECT c.fecha, e.id AS embalse_id
FROM calendario c
CROSS JOIN embalses e;
```

```sql
-- Resumen de completitud por embalse
SELECT
    e.id AS embalse_id,
    e.nombre AS embalse_nombre,
    r.nombre AS region_nombre,
    COALESCE(COUNT(a.id), 0) AS dias_con_datos,
    731 AS dias_totales,
    (COALESCE(COUNT(a.id), 0) * 100.0 / 731) AS porcentaje_completitud
INTO completitud
FROM embalses e
JOIN regiones r ON e.region_id = r.id
LEFT JOIN aportes a ON e.id = a.embalse_id
GROUP BY e.id, e.nombre, r.nombre
ORDER BY porcentaje_completitud DESC;
```

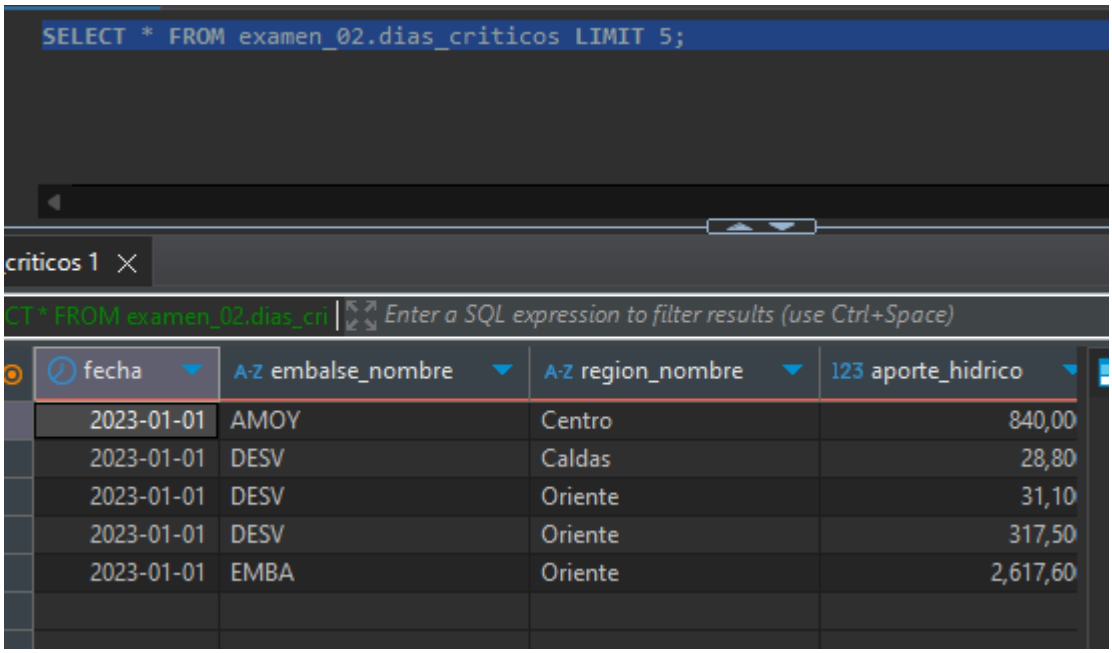
```sql
-- Calcular el umbral mínimo (percentil 10) por embalse
CREATE TABLE umbrales AS
SELECT
    embalse_id,
    percentile_cont(0.1) WITHIN GROUP (ORDER BY aporte_hidrico) AS umbral_minimo
FROM aportes
GROUP BY embalse_id;

-- Identificar días con aportes por debajo del umbral
SELECT
    a.fecha,
    e.nombre AS embalse_nombre,
    r.nombre AS region_nombre,
    a.aporte_hidrico,
    u.umbral_minimo
INTO dias_criticos
FROM aportes a
JOIN embalses e ON a.embalse_id = e.id
JOIN regiones r ON a.region_id = r.id
JOIN umbrales u ON a.embalse_id = u.embalse_id
WHERE a.aporte_hidrico < u.umbral_minimo
ORDER BY a.fecha, e.nombre;
```

```sql
-- Agrupar por períodos consecutivos
WITH periodos_criticos AS (
    SELECT
        a.fecha,
        e.nombre AS embalse_nombre,
        r.nombre AS region_nombre,
        a.aporte_hidrico,
        u.umbral_minimo,
        SUM(CASE WHEN a.aporte_hidrico < u.umbral_minimo THEN 1 ELSE 0 END)
            OVER (PARTITION BY e.id ORDER BY a.fecha) AS grupo
    FROM aportes a
    JOIN embalses e ON a.embalse_id = e.id
    JOIN regiones r ON a.region_id = r.id
    JOIN umbrales u ON a.embalse_id = u.embalse_id
)
SELECT
    embalse_nombre,
    region_nombre,
    MIN(fecha) AS inicio_periodo,
    MAX(fecha) AS fin_periodo,
    COUNT(*) AS dias_criticos
INTO periodos_criticos
FROM periodos_criticos
WHERE aporte_hidrico < umbral_minimo
GROUP BY embalse_nombre, region_nombre, grupo
ORDER BY inicio_periodo;
```

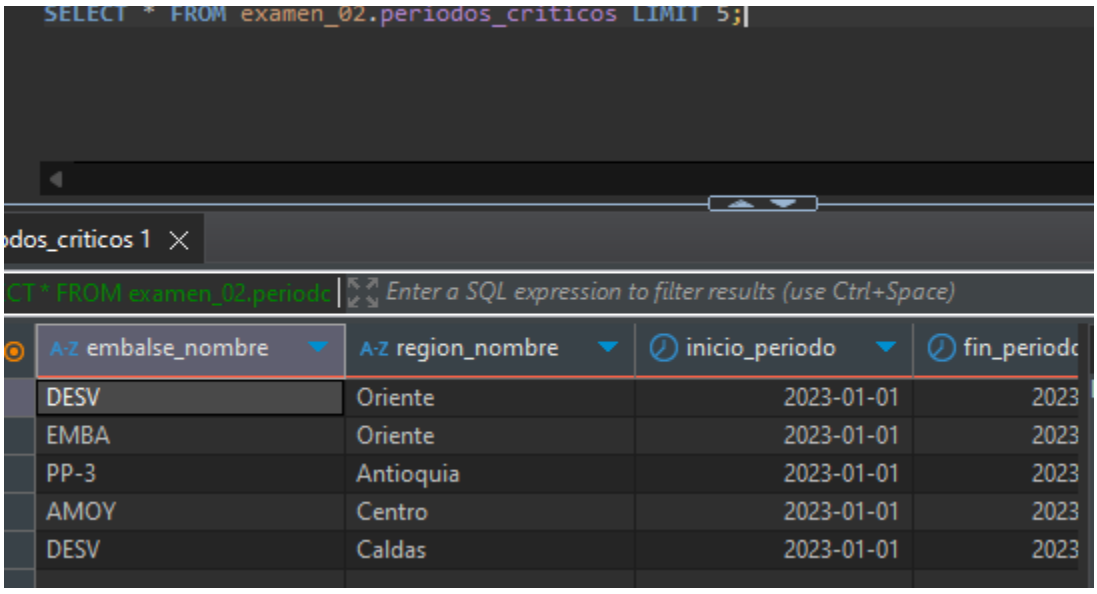




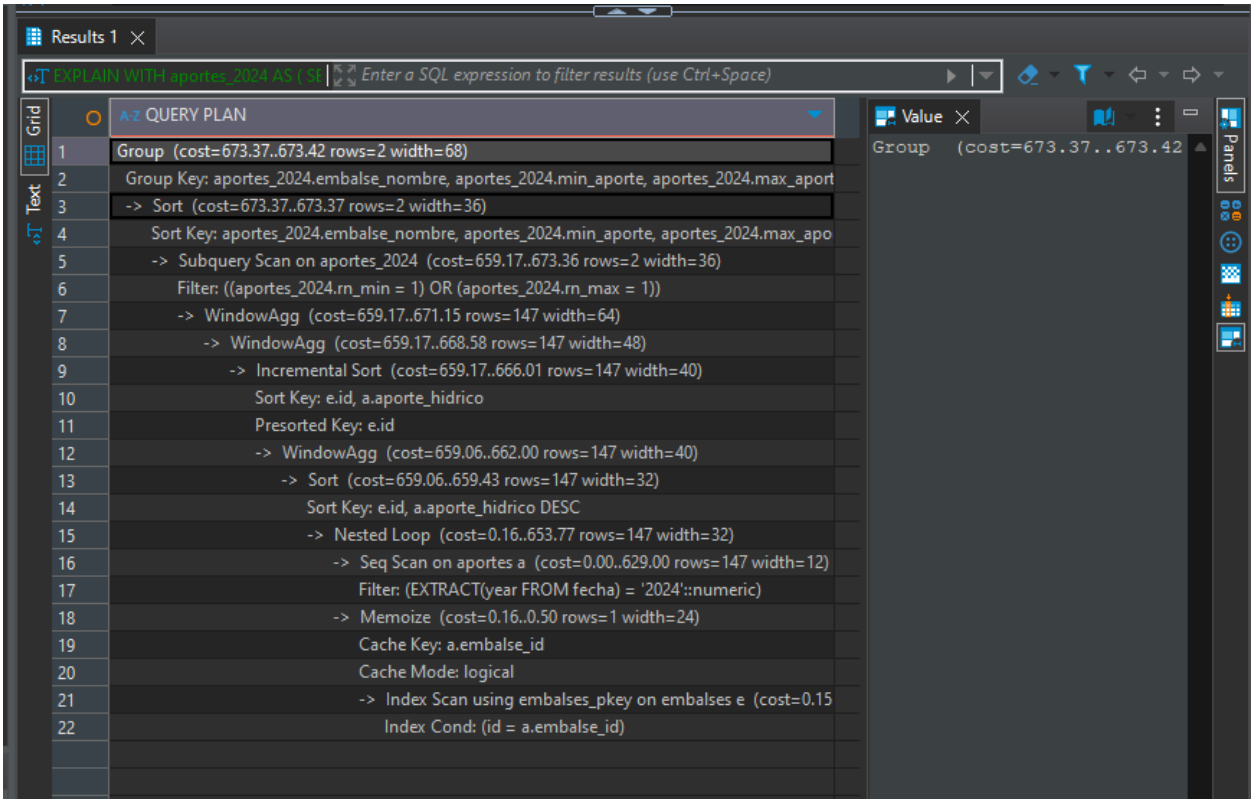
fecha	embalse_id	embalse_nombre	region_nc
2023-01-01 00:00:00.000 -0500	38	ESCM	Antioquia
2023-01-01 00:00:00.000 -0500	39	CAME	Caldas
2023-01-01 00:00:00.000 -0500	40	CAME	Caldas
2023-01-01 00:00:00.000 -0500	41	ESME	Caldas
2023-01-01 00:00:00.000 -0500	42	ESTR	Caldas



fecha	embalse_nombre	region_nombre	aporte_hidrico
2023-01-01	AMOY	Centro	840,00
2023-01-01	DESV	Caldas	28,80
2023-01-01	DESV	Oriente	31,10
2023-01-01	DESV	Oriente	317,50
2023-01-01	EMBA	Oriente	2,617,60



embalse_nombre	region_nombre	inicio_periodo	fin_periodc
DESV	Oriente	2023-01-01	2023
EMBA	Oriente	2023-01-01	2023
PP-3	Antioquia	2023-01-01	2023
AMOY	Centro	2023-01-01	2023
DESV	Caldas	2023-01-01	2023


Results 1
EXPLAIN WITH aportes_2024 AS (SE
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
QUERY PLAN
Value
Panels
1 Group (cost=673.37..673.42 rows=2 width=68)
2 Group Key: aportes_2024.embalse_nombre, aportes_2024.min_aporte, aportes_2024.max_aport
3 -> Sort (cost=673.37..673.37 rows=2 width=36)
4 Sort Key: aportes_2024.embalse_nombre, aportes_2024.min_aporte, aportes_2024.max_apo
5 -> Subquery Scan on aportes_2024 (cost=659.17..673.36 rows=2 width=36)
6 Filter: ((aportes_2024.rn_min = 1) OR (aportes_2024.rn_max = 1))
7 -> WindowAgg (cost=659.17..671.15 rows=147 width=64)
8 -> WindowAgg (cost=659.17..668.58 rows=147 width=48)
9 -> Incremental Sort (cost=659.17..666.01 rows=147 width=40)
10 Sort Key: e.id, a.aporte_hidrico
11 Presorted Key: e.id
12 -> WindowAgg (cost=659.06..662.00 rows=147 width=40)
13 -> Sort (cost=659.06..659.43 rows=147 width=32)
14 Sort Key: e.id, a.aporte_hidrico DESC
15 -> Nested Loop (cost=0.16..653.77 rows=147 width=32)
16 -> Seq Scan on aportes a (cost=0.00..629.00 rows=147 width=12)
17 Filter: (EXTRACT(year FROM fecha) = '2024'::numeric)
18 -> Memoize (cost=0.16..0.50 rows=1 width=24)
19 Cache Key: a.embalse_id
20 Cache Mode: logical
21 -> Index Scan using embalses_pkey on embalses e (cost=0.15
22 Index Cond: (id = a.embalse_id)
Group (cost=673.37..673.42