Universidad Nacional Autónoma de México

Facultad de Ingeniería

Curso: Bases de Datos

Grupo: 01

Serie SQL

Alumno:
Ortiz Valles Joaquín Rafael

Profesor:
Ing. Fernando Arreola Franco

Diciembre 2025

1. Configuración y Scripts (Ejercicio 1)

Enunciado: Editar el archivo script_bd2.sql teniendo en cuenta las siguientes consideraciones:

- Para la tabla cliente, agregar PK y que el atributo estado tome por defecto el valor cdmx.
- Para la tabla artículo agregar una restricción que no permita ingresar un valor menor que 0 en el atributo precio, así como la PK.
- Para la tabla orden, agregar PK, permitir que el atributo fecha tome por defecto la hora del sistema y considerar que hay una relación 1:M entre la tabla cliente y orden, por lo que debe agregar la FK donde corresponda.

DDL corregido y para PostgreSQL:

```
-- 1. Crear base de datos y conectar
CREATE DATABASE datos_clase;
\c datos_clase

-- 2. Tabla Cliente
-- Agregar PK y default 'cdmx' en estado
CREATE TABLE cliente (
    id_Cliente varchar(13) PRIMARY KEY, -- PK Agregada
    nombre char(50),
    ap_Pat char(50) not null,
    ap_Mat char(50),
    estado varchar(25) DEFAULT 'cdmx'   -- Default agregado
);

-- 3. Tabla Articulo
-- Restricci n precio >= 0 y PK. Se cambia char a numeric.
CREATE TABLE articulo (
    num_Articulo int PRIMARY KEY,
    nombre_Articulo varchar(30) not null,
    precio numeric(10,2) not null CHECK (precio >= 0),
    categoria varchar(35) not null
);

-- 4. Tabla Orden
-- PK, default fecha sistema, Relaci n 1:M con Cliente
CREATE TABLE orden (
    id_Orden int PRIMARY KEY,
    fecha timestamp not null DEFAULT NOW(), -- Default hora sistema
    id_Cliente varchar(13),
    CONSTRAINT fk_cliente FOREIGN KEY (id_Cliente) REFERENCES
   cliente(id_Cliente)
);
```

```sql
-- 1. Crear base de datos y conectar
CREATE DATABASE serie_sql;

-- 2. Tabla Cliente
-- Agregar PK y default 'cdmx' en estado
CREATE TABLE cliente (
    id_Cliente varchar(13) PRIMARY KEY, -- PK Agregada
    nombre char(50),
    ap_Pat char(50) not null,
    ap_Mat char(50),
    estado varchar(25) DEFAULT 'cdmx'   -- Default agregado
);

-- 3. Tabla Articulo
-- Restricción precio >= 0 y PK
CREATE TABLE articulo (
    num_Articulo int PRIMARY KEY,       -- PK Agregada
    nombre_Articulo varchar(30) not null, -- Corregido "varrchar"
    precio numeric(10,2) not null CHECK (precio >= 0), -- Cambiado a numeric para validar >= 0
    categoria varchar(35) not null
);

-- 4. Tabla Orden
-- PK, default fecha sistema, Relación 1:M con Cliente
CREATE TABLE orden (
    id_Orden int PRIMARY KEY,           -- PK Agregada
    fecha timestamp not null DEFAULT NOW(), -- Default hora sistema
    id_Cliente varchar(13),             -- FK debe ser del mismo tipo que en tabla cliente
    CONSTRAINT fk_cliente FOREIGN KEY (id_Cliente) REFERENCES cliente(id_Cliente)
```

```
CREATE TABLE

Query returned successfully in 59 msec.
```

Figura 1: Creación de las tablas

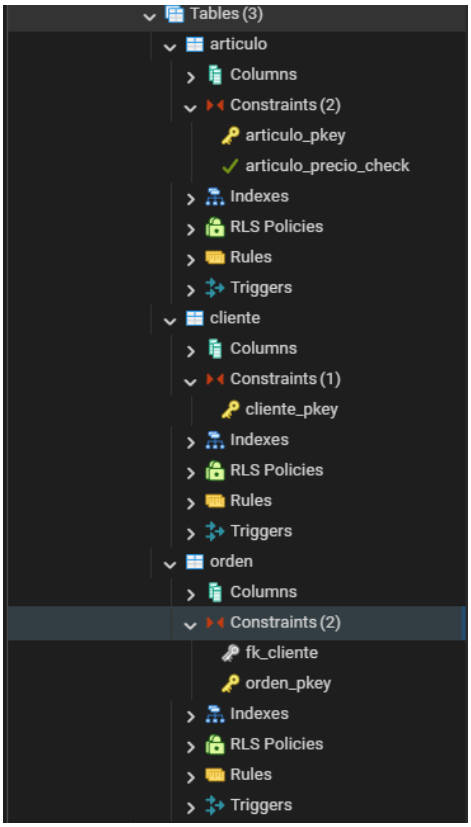


Figura 2: Tablas de nuestra base

2. Consultas de Vuelos (Ejercicios 2-13)

Nota: Todas las consultas incluyen la fecha del sistema y el usuario actual como se solicitó en las instrucciones generales.

Ejercicio 2

Enunciado: Indicar las ciudades que tienen más de un aeropuerto.

```sql
SELECT city, COUNT(*) as cantidad_aeropuertos, CURRENT_USER, NOW()
FROM aeropuertos
GROUP BY city
HAVING COUNT(*) > 1;
```

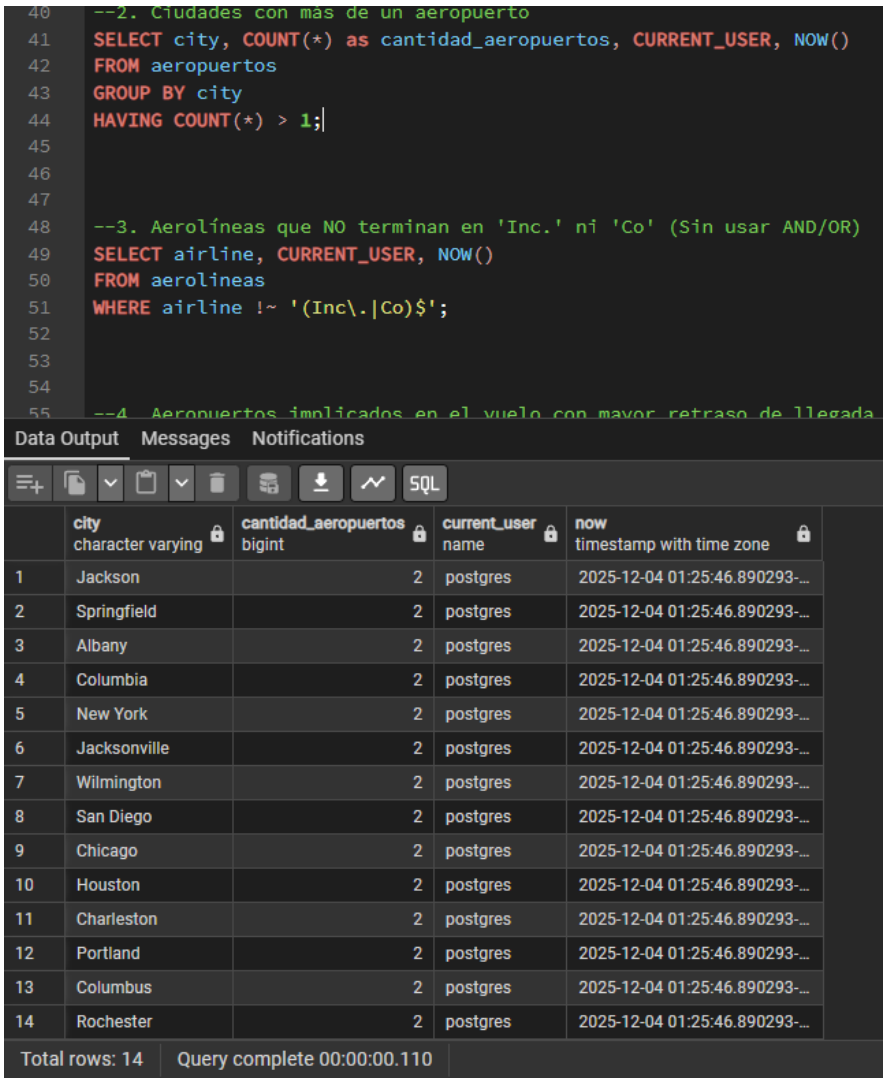


Figura 3: Ciudades con más de un aeropuerto

Ejercicio 3

Enunciado: Nombre de las aerolíneas que no terminan en Inc. ni en Co sin usar operadores AND y OR.

```sql
-- Se utiliza expresion regular para evitar el uso de AND/OR
SELECT airline, CURRENT_USER, NOW()
FROM aerolineas
WHERE airline !~ '(Inc\.|Co)$';
```

```sql
--3. Aerolíneas que NO terminan en 'Inc.' ni 'Co' (Sin usar AND/OR)
SELECT airline, CURRENT_USER, NOW()
FROM aerolineas
WHERE airline !~ '(Inc\.|Co)$';

--4. Aeropuertos implicados en el vuelo con mayor retraso de llegada
```

Data Output | Messages | Notifications

	airline character varying	current_user name	now timestamp with time zone
1	JetBlue Airways	postgres	2025-12-04 01:29:45.039087-...
2	Spirit Air Lines	postgres	2025-12-04 01:29:45.039087-...
3	Southwest Airlines Co.	postgres	2025-12-04 01:29:45.039087-...
4	Atlantic Southeast Airlin...	postgres	2025-12-04 01:29:45.039087-...
5	Virgin America	postgres	2025-12-04 01:29:45.039087-...

Total rows: 5 | Query complete 00:00:00.103

Figura 4: Aerolíneas que NO terminan en 'Inc.' ni 'Co'

Ejercicio 4

Enunciado: Indicar los nombres de los aeropuertos que estuvieron implicados en el vuelo que presentó el mayor retraso de llegada.

```sql
SELECT origin_airport, destination_airport, arrivale_delay,
   CURRENT_USER, NOW()
FROM vuelos
WHERE arrivale_delay = (SELECT MAX(arrivale_delay) FROM vuelos);
```

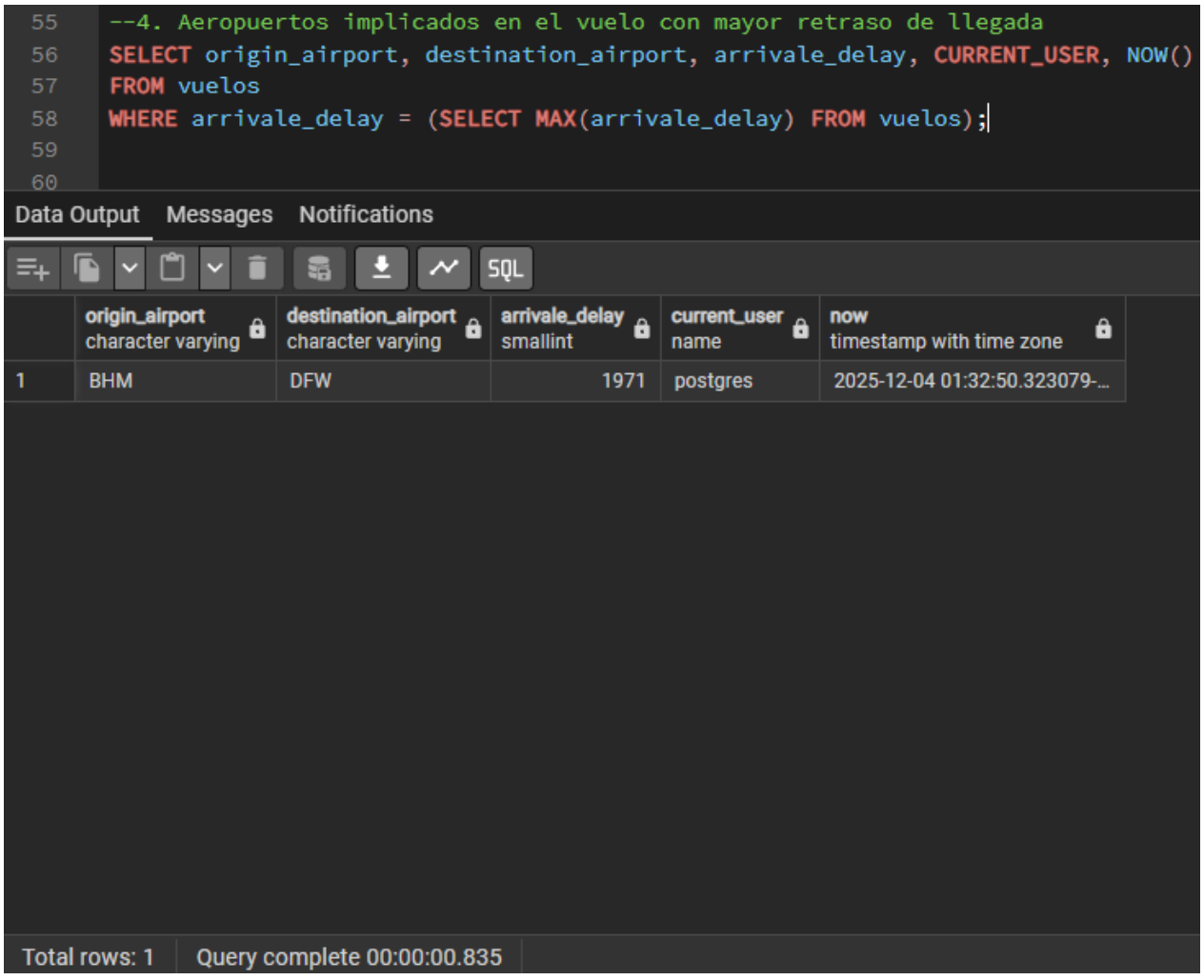


Figura 5: Aeropuertos implicados en el vuelo con mayor retraso de llegada

Ejercicio 5

Enunciado: Mostrar aquella categoría (tabla artículo) que tiene el precio mínimo. La información debe estar agrupada.

```sql
SELECT categoria, MIN(precio) as precio_minimo, CURRENT_USER, NOW()
FROM articulo
GROUP BY categoria
ORDER BY precio_minimo ASC
LIMIT 1;
```

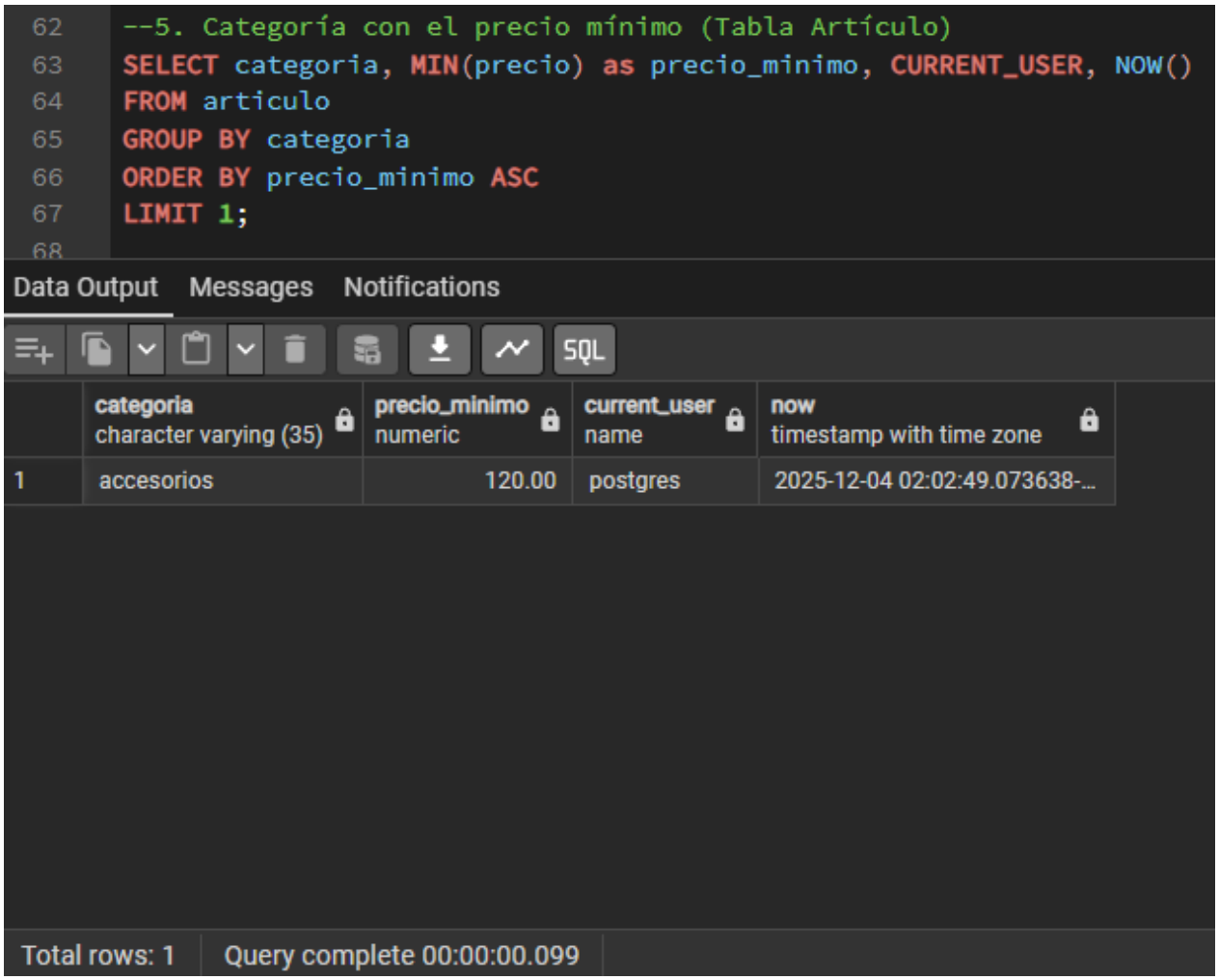


Figura 6: Categoría precio mínimo

Ejercicio 6

Enunciado: Se desea conocer el nombre de aquellas aerolíneas cuyo segundo caracter del iata_code termina en X ó 9. Debe incluirse una columna que muestre dicha terminación.

```sql
SELECT
    airline,
    iata_code,
```

```
    SUBSTRING(iata_code, 2, 1) as terminacion,
    CURRENT_USER, NOW()
FROM aerolineas
WHERE SUBSTRING(iata_code, 2, 1) SIMILAR TO '[X9]';
```

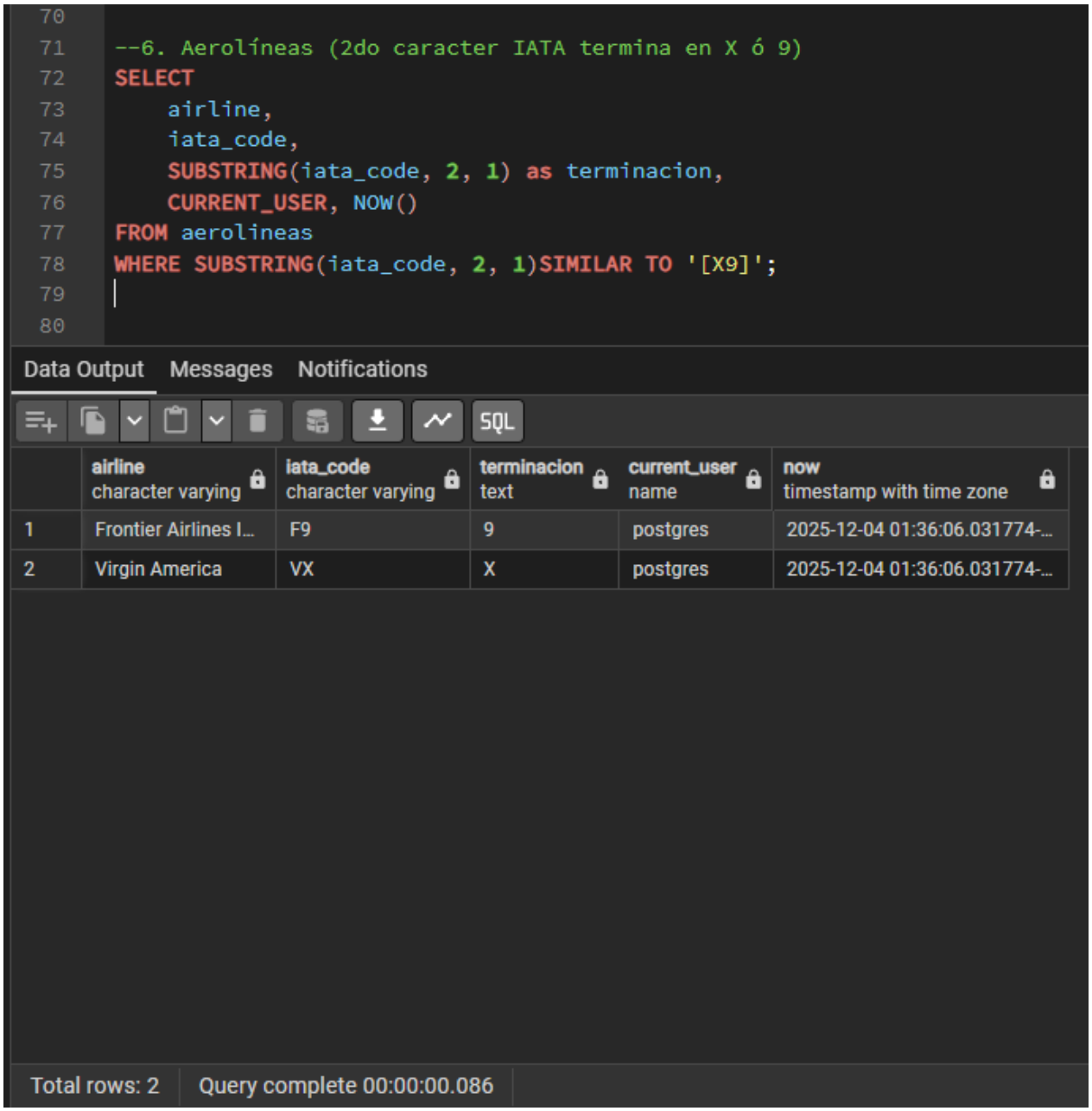
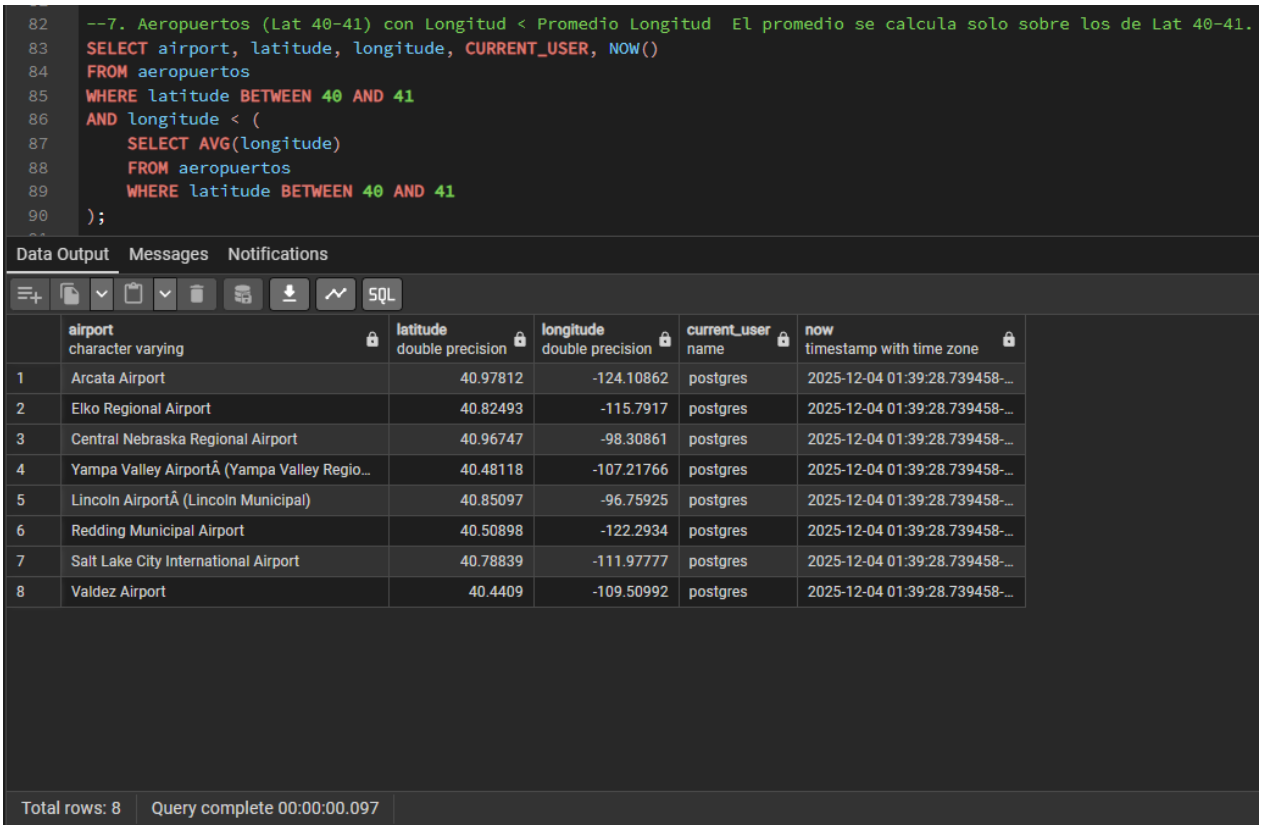


Figura 7: Aerolíneas (2do caracter IATA termina en X ó 9)

Ejercicio 7

Enunciado: Proporcionar el nombre de los aeropuertos cuya latitud se encuentre entre 40 y 41, y su longitud sea menor que el promedio de la longitud (tomando el promedio de aquellas observaciones cuya latitud esté entre 40 y 41).

```sql
SELECT airport, latitude, longitude, CURRENT_USER, NOW()
FROM aeropuertos
WHERE latitude BETWEEN 40 AND 41
AND longitude < (
    SELECT AVG(longitude)
    FROM aeropuertos
    WHERE latitude BETWEEN 40 AND 41
);
```



```sql
--7. Aeropuertos (Lat 40-41) con Longitud < Promedio Longitud  El promedio se calcula solo sobre los de Lat 40-41.
SELECT airport, latitude, longitude, CURRENT_USER, NOW()
FROM aeropuertos
WHERE latitude BETWEEN 40 AND 41
AND longitude < (
    SELECT AVG(longitude)
    FROM aeropuertos
    WHERE latitude BETWEEN 40 AND 41
);
```

Data Output | Messages | Notifications

	airport character varying	latitude double precision	longitude double precision	current_user name	now timestamp with time zone
1	Arcata Airport	40.97812	-124.10862	postgres	2025-12-04 01:39:28.739458-...
2	Elko Regional Airport	40.82493	-115.7917	postgres	2025-12-04 01:39:28.739458-...
3	Central Nebraska Regional Airport	40.96747	-98.30861	postgres	2025-12-04 01:39:28.739458-...
4	Yampa Valley AirportÂ (Yampa Valley Regio...	40.48118	-107.21766	postgres	2025-12-04 01:39:28.739458-...
5	Lincoln AirportÂ (Lincoln Municipal)	40.85097	-96.75925	postgres	2025-12-04 01:39:28.739458-...
6	Redding Municipal Airport	40.50898	-122.2934	postgres	2025-12-04 01:39:28.739458-...
7	Salt Lake City International Airport	40.78839	-111.97777	postgres	2025-12-04 01:39:28.739458-...
8	Valdez Airport	40.4409	-109.50992	postgres	2025-12-04 01:39:28.739458-...

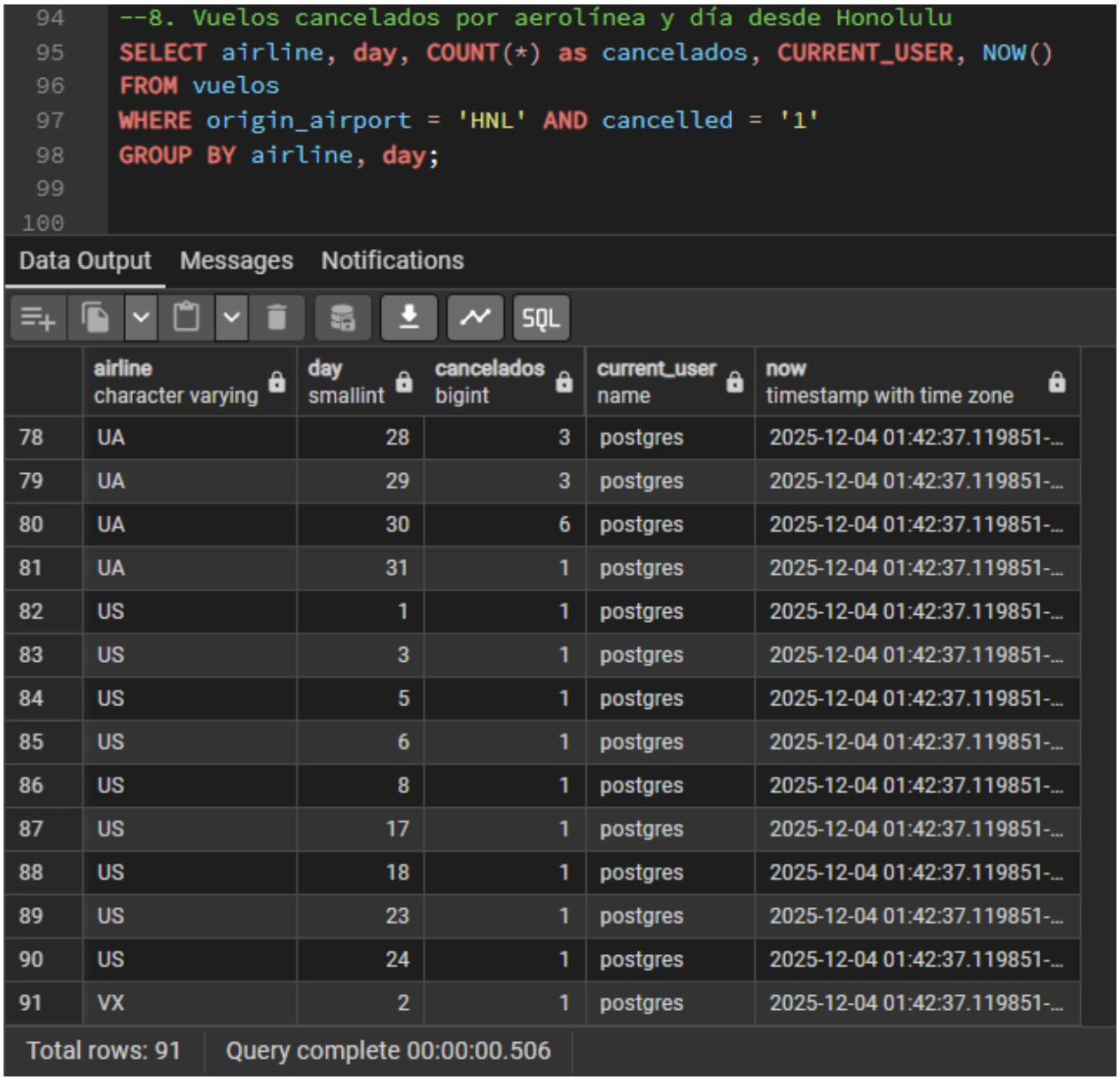
Total rows: 8 | Query complete 00:00:00.097

Figura 8: Latitud/Longitud

Ejercicio 8

Enunciado: ¿Cuántos aviones por aerolínea y día, fueron cancelados saliendo del aeropuerto de Honolulu?

```sql
SELECT airline, day, COUNT(*) as cancelados, CURRENT_USER, NOW()
FROM vuelos
WHERE origin_airport = 'HNL' AND cancelled = '1'
GROUP BY airline, day;
```



```sql
--8. Vuelos cancelados por aerolínea y día desde Honolulu
SELECT airline, day, COUNT(*) as cancelados, CURRENT_USER, NOW()
FROM vuelos
WHERE origin_airport = 'HNL' AND cancelled = '1'
GROUP BY airline, day;
```

Data Output | Messages | Notifications

	airline character varying	day smallint	cancelados bigint	current_user name	now timestamp with time zone
78	UA	28	3	postgres	2025-12-04 01:42:37.119851-...
79	UA	29	3	postgres	2025-12-04 01:42:37.119851-...
80	UA	30	6	postgres	2025-12-04 01:42:37.119851-...
81	UA	31	1	postgres	2025-12-04 01:42:37.119851-...
82	US	1	1	postgres	2025-12-04 01:42:37.119851-...
83	US	3	1	postgres	2025-12-04 01:42:37.119851-...
84	US	5	1	postgres	2025-12-04 01:42:37.119851-...
85	US	6	1	postgres	2025-12-04 01:42:37.119851-...
86	US	8	1	postgres	2025-12-04 01:42:37.119851-...
87	US	17	1	postgres	2025-12-04 01:42:37.119851-...
88	US	18	1	postgres	2025-12-04 01:42:37.119851-...
89	US	23	1	postgres	2025-12-04 01:42:37.119851-...
90	US	24	1	postgres	2025-12-04 01:42:37.119851-...
91	VX	2	1	postgres	2025-12-04 01:42:37.119851-...

Total rows: 91 | Query complete 00:00:00.506

Figura 9: Cancelados Honolulu

Ejercicio 9

Enunciado: Hacer un cross join entre la tabla cliente y la tabla aerolíneas.

```sql
-- Asumiendo permisos de lectura cruzada o dblink
SELECT c.nombre, a.airline, CURRENT_USER, NOW()
FROM datos_clase.public.cliente c
CROSS JOIN registro_vuelos.public.aerolineas a;
```

```sql
--9. Cross Join entre Cliente y Aerolíneas
SELECT
    c.nombre,
    a.airline,
    CURRENT_USER as usuario,
    NOW() as fecha
FROM cliente c
CROSS JOIN aerolineas_temp a;
```

Data Output | Messages | Notifications

	nombre character (50)	airline character varying (50)	usuario name	fecha timestamp with time zone
1	Jaime	American Airlines	postgres	2025-12-04 02:06:24.703068-...
2	Luisa	American Airlines	postgres	2025-12-04 02:06:24.703068-...
3	Mario	American Airlines	postgres	2025-12-04 02:06:24.703068-...
4	Jaime	Aeromexico	postgres	2025-12-04 02:06:24.703068-...
5	Luisa	Aeromexico	postgres	2025-12-04 02:06:24.703068-...
6	Mario	Aeromexico	postgres	2025-12-04 02:06:24.703068-...
7	Jaime	Delta Air Lines	postgres	2025-12-04 02:06:24.703068-...
8	Luisa	Delta Air Lines	postgres	2025-12-04 02:06:24.703068-...
9	Mario	Delta Air Lines	postgres	2025-12-04 02:06:24.703068-...

Total rows: 9 | Query complete 00:00:00.070

Figura 10: Cross Join

Ejercicio 10

Enunciado: Cantidad de vuelos cancelados por día.

```sql
SELECT year, month, day, COUNT(*) as total_cancelados, CURRENT_USER,
    NOW()
FROM vuelos
```

```
WHERE cancelled = '1'
GROUP BY year, month, day;
```

```
--10. Cantidad de vuelos cancelados por día
SELECT year, month, day, COUNT(*) as total_cancelados, CURRENT_USER, NOW()
FROM vuelos
WHERE cancelled = '1'
GROUP BY year, month, day;
```

Data Output | Messages | Notifications

	year smallint	month smallint	day smallint	total_cancelados bigint	current_user name	now timestamp with time zone
352	2015	12	18	168	postgres	2025-12-04 01:47:27.426482-...
353	2015	12	19	59	postgres	2025-12-04 01:47:27.426482-...
354	2015	12	20	119	postgres	2025-12-04 01:47:27.426482-...
355	2015	12	21	172	postgres	2025-12-04 01:47:27.426482-...
356	2015	12	22	141	postgres	2025-12-04 01:47:27.426482-...
357	2015	12	23	212	postgres	2025-12-04 01:47:27.426482-...
358	2015	12	24	308	postgres	2025-12-04 01:47:27.426482-...
359	2015	12	25	142	postgres	2025-12-04 01:47:27.426482-...
360	2015	12	26	392	postgres	2025-12-04 01:47:27.426482-...
361	2015	12	27	1158	postgres	2025-12-04 01:47:27.426482-...
362	2015	12	28	2177	postgres	2025-12-04 01:47:27.426482-...
363	2015	12	29	685	postgres	2025-12-04 01:47:27.426482-...
364	2015	12	30	281	postgres	2025-12-04 01:47:27.426482-...
365	2015	12	31	42	postgres	2025-12-04 01:47:27.426482-...

Total rows: 365 | Query complete 00:00:00.494

Figura 11: Cancelados por día

Ejercicio 11

Enunciado: Seleccionar el nombre de los aeropuertos cuya segunda letra del iata code sea K ó X, sin usar operadores AND, NOT u OR.

```
SELECT airport, iata_code, CURRENT_USER, NOW()
FROM aeropuertos
WHERE SUBSTRING(iata_code, 2, 1) ~ '[KX]';
```

```
116
117  --11. Aeropuertos (2da letra IATA es K ó X) sin AND/NOT/OR
118  SELECT airport, iata_code, CURRENT_USER, NOW()
119  FROM aeropuertos
120  WHERE SUBSTRING(iata_code, 2, 1) ~ '[KX]';
121
122
123
```

Data Output | Messages | Notifications

SQL

	airport character varying	iata_code character varying	current_user name	now timestamp with time zone
1	King Salmon Airport	AKN	postgres	2025-12-04 01:49:47.768006-...
2	Elko Regional Airport	EKO	postgres	2025-12-04 01:49:47.768006-...
3	General Mitchell International Airport	MKE	postgres	2025-12-04 01:49:47.768006-...
4	Muskegon County Airport	MKG	postgres	2025-12-04 01:49:47.768006-...
5	Will Rogers World Airport	OKC	postgres	2025-12-04 01:49:47.768006-...
6	Rock Springs-Sweetwater County Airport	RKS	postgres	2025-12-04 01:49:47.768006-...
7	Texarkana Regional AirportÂ (Webb Fie...	TXK	postgres	2025-12-04 01:49:47.768006-...

Total rows: 7 | Query complete 00:00:00.119

Figura 12: Aeropuertos K o X

Ejercicio 12

Enunciado: Indicar el nombre(s) de la aerolínea cuya distancia de vuelo es la mayor.

```sql
SELECT al.airline, v.distance, CURRENT_USER, NOW()
FROM vuelos v
JOIN aerolineas al ON v.airline = al.iata_code
ORDER BY distance DESC
LIMIT 1;
```

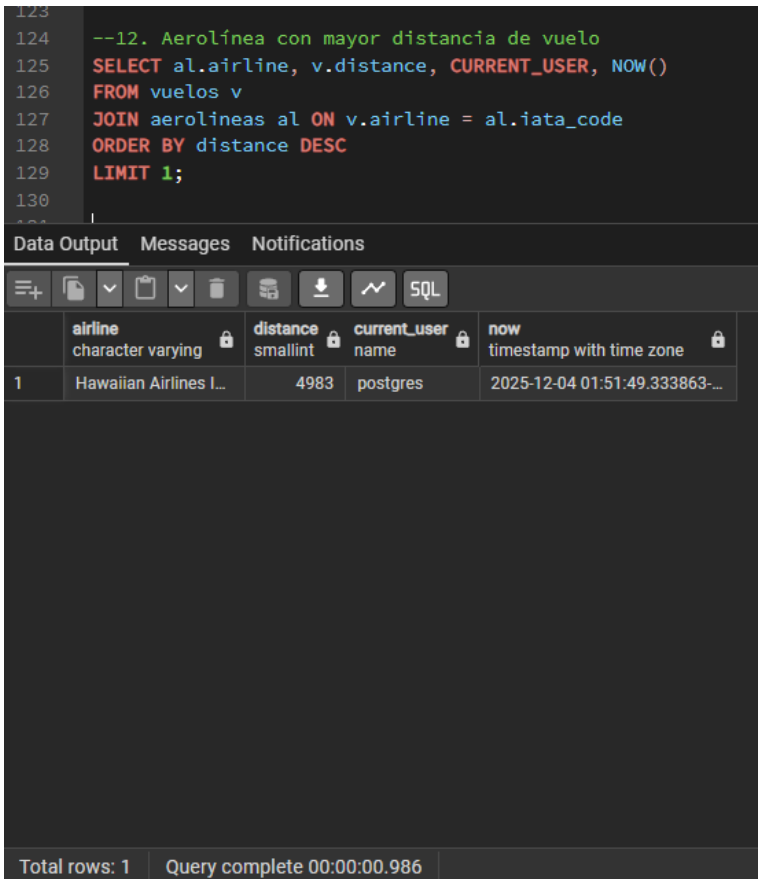


Figura 13: Aerolínea mayor distancia

Ejercicio 13

Enunciado: Indicar el nombre del aeropuerto de origen donde se presentó el mayor tiempo de vuelo.

```sql
SELECT origin_airport, air_time, CURRENT_USER, NOW()
FROM vuelos
WHERE air_time = (SELECT MAX(air_time) FROM vuelos);
```

```
132
133   --13. Aeropuerto de origen con mayor tiempo de vuelo
134   SELECT origin_airport, air_time, CURRENT_USER, NOW()
135   FROM vuelos
136   WHERE air_time = (SELECT MAX(air_time) FROM vuelos);
```

Data Output | Messages | Notifications

SQL

	origin_airport character varying	air_time smallint	current_user name	now timestamp with time zone
1	JFK	690	postgres	2025-12-04 01:53:14.563804-...
2	JFK	690	postgres	2025-12-04 01:53:14.563804-...

Total rows: 2 | Query complete 00:00:00.743

Figura 14: Aeropuerto mayor tiempo vuelo

3. Modelado Entidad-Relación (Ejercicio 14)

Enunciado: Partiendo del MER proporcionado (Votante, Partido, Candidato), generar el mapeo a la representación intermedia de MR y generar el DDL. Para las FK debe establecerse la restricción cascade para borrado y actualización.
Solución - Modelo Relacional:

- **Partido** (siglas, nombre, año_fundacion)
- **Candidato** (rfc, nombre, años_trayectoria, siglas_partido_fk)
- **Candidato_Emails** (rfc_candidato, email) [Atributo multivalorado]
- **Votante** (id, nombre, dirección, edad, fecha_nacimiento, rfc_candidato_fk)

Solución - DDL:

```
CREATE TABLE Partido (
    siglas varchar(10) PRIMARY KEY,
    nombre varchar(100),
    anio_fundacion int
);

CREATE TABLE Candidato (
    rfc varchar(13) PRIMARY KEY,
    nombre varchar(100),
    anios_trayectoria int,
    siglas_partido varchar(10),
    CONSTRAINT fk_partido FOREIGN KEY (siglas_partido)
        REFERENCES Partido(siglas)
        ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Candidato_Emails (
    rfc_candidato varchar(13),
    email varchar(100),
    PRIMARY KEY (rfc_candidato, email),
    CONSTRAINT fk_email_cand FOREIGN KEY (rfc_candidato)
        REFERENCES Candidato(rfc)
        ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Votante (
    id int PRIMARY KEY,
    nombre varchar(100),
    direccion varchar(150),
    edad int,
    fecha_nacimiento date,
    rfc_candidato_voto varchar(13),
```

```
    CONSTRAINT fk_voto FOREIGN KEY (rfc_candidato_voto)
        REFERENCES Candidato(rfc)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Query | Query History

```
CREATE TABLE Partido (
    siglas varchar(10) PRIMARY KEY,
    nombre varchar(100),
    año_fundacion int
);

CREATE TABLE Candidato (
    rfc varchar(13) PRIMARY KEY,
    nombre varchar(100),
    años_trayectoria int,
    siglas_partido varchar(10),
    CONSTRAINT fk_partido FOREIGN KEY (siglas_partido)
        REFERENCES Partido(siglas)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Tabla para atributo multivalorado 'emails'
CREATE TABLE Candidato_Emails (
    rfc_candidato varchar(13),
    email varchar(100),
    PRIMARY KEY (rfc_candidato, email),
    CONSTRAINT fk_email_cand FOREIGN KEY (rfc_candidato)
        REFERENCES Candidato(rfc)
        ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Votante (
    id int PRIMARY KEY,
    nombre varchar(100),
    direccion varchar(150),
    edad int,
    fecha_nacimiento date,
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 53 msec.
```

Total rows: | Query complete 00:00:00.053

Figura 15: script DDL

4. Procedimientos Almacenados (Ejercicio 15)

Enunciado: Genere una función o procedure que reciba el nombre de una categoría, itere sobre los registros (usando cursores), seleccione nombre y precio, y además incluya el precio mínimo y máximo de la categoría.

Solución:

```
CREATE OR REPLACE FUNCTION reporte_categoria(categoria_input varchar
    )
RETURNS void AS $$
DECLARE
    min_precio numeric;
    max_precio numeric;
    rec_articulo RECORD;
    cur_articulos CURSOR FOR
        SELECT nombre_Articulo, precio
        FROM articulo WHERE categoria = categoria_input;
BEGIN
    -- 1. Obtener Min y Max de la categor a
    SELECT MIN(precio), MAX(precio) INTO min_precio, max_precio
    FROM articulo WHERE categoria = categoria_input;

    RAISE NOTICE 'Reporte para categor a: %', categoria_input;
    RAISE NOTICE 'Rango Global: % - %', min_precio, max_precio;

    -- 2. Abrir cursor e iterar
    OPEN cur_articulos;
    LOOP
        FETCH cur_articulos INTO rec_articulo;
        EXIT WHEN NOT FOUND;

        RAISE NOTICE 'Articulo: %, Precio: %',
            rec_articulo.nombre_Articulo, rec_articulo.precio;
    END LOOP;
    CLOSE cur_articulos;
END;
$$ LANGUAGE plpgsql;
```

Query | Query History

```
64      -- 1. Obtener Min y Max de la categoría
65      SELECT MIN(precio), MAX(precio) INTO min_precio, max_precio
66      FROM articulo
67      WHERE categoria = categoria_input;
68
69      -- Cabecera del reporte
70      RAISE NOTICE 'Reporte para categoría: %', categoria_input;
71      RAISE NOTICE 'Precio Mínimo Global: %, Precio Máximo Global: %', min_precio, max_precio;
72      RAISE NOTICE '--------------------------------------------------';
73
74      -- 2. Abrir cursor e iterar
75      OPEN cur_articulos;
76
77      LOOP
78          FETCH cur_articulos INTO rec_articulo;
79          EXIT WHEN NOT FOUND;
80
81          --Imprimir registro + estadísticos
82          RAISE NOTICE 'Articulo: %, Precio: % (Rango Cat: % - %)',
83              rec_articulo.nombre_Articulo,
84              rec_articulo.precio,
85              min_precio,
86              max_precio;
87      END LOOP;
88
89      CLOSE cur_articulos;
90  END;
91  $$ LANGUAGE plpgsql;
92
```

Data Output | Messages | Notifications

```
NOTICE:  Reporte para categoría: ropa
NOTICE:  Precio Mínimo Global: 450.00, Precio Máximo Global: 650.00
NOTICE:  --------------------------------------------------
NOTICE:  Articulo: Bermuda, Precio: 500.00 (Rango Cat: 450.00 - 650.00)
NOTICE:  Articulo: Jeans, Precio: 650.00 (Rango Cat: 450.00 - 650.00)
NOTICE:  Articulo: Falda, Precio: 450.00 (Rango Cat: 450.00 - 650.00)

Successfully run. Total query runtime: 70 msec.
1 rows affected.
```

Total rows: 1 | Query complete 00:00:00.070

Figura 16: Procedimiento Almacenado

Referencias

[1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Hoboken, NJ, USA: Pearson, 2016.

[2] PostgreSQL Global Development Group, "PostgreSQL 16.1 Documentation," [Online]. Available: https://www.postgresql.org/docs/. [Accessed: Dec. 03, 2025].

[3] C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston, MA, USA: Addison-Wesley, 2003.

[4] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. New York, NY, USA: McGraw-Hill Education, 2019.