

## M5\_Evaluación del Portafolio [Actividad Evaluada]

## M5\_Evaluación del Portafolio [Actividad Evaluada]

## Evaluación de portafolio

## Instrucciones

En función de tu proyecto personal previamente establecido, deberás implementar clase a clase las diferentes tecnologías y competencias técnicas adquiridas a lo largo del curso.

Recuerda que este proyecto irá directamente al registro de evidencia de tu portafolio, el cual deberá demostrar el dominio, competencias técnicas y diferentes habilidades relacionadas con la gestión de bases de datos relacionales.

## Requerimientos Funcionales Mínimos Esperados

1. **Distinguir las características, rol y elementos fundamentales de una base de datos relacional para la gestión de la información en una organización.**
  - Describir los componentes básicos de una base de datos relacional: tablas, registros, campos, claves primarias y foráneas.
  - Explicar cómo se gestionan y almacenan los datos en tablas y cómo se establece la relación entre ellas para satisfacer necesidades organizacionales.
  - Ejemplo: Crear una tabla de clientes y otra de pedidos, relacionándolas por una clave foránea.
2. **Utilizar Lenguaje Estructurado de Consultas (SQL) para la obtención de información que satisface los requerimientos planteados a partir de un modelo de datos dado.**
  - Desarrollar consultas SQL para obtener información específica de las tablas, utilizando cláusulas como **SELECT**, **WHERE**, **JOIN**, **GROUP BY**, entre otras.
  - Ejemplo: Crear una consulta que obtenga todos los pedidos realizados por un cliente específico.
3. **Utilizar lenguaje de manipulación de datos (DML) para la modificación de los datos existentes en una base de datos dando solución a un problema planteado.**
  - Implementar consultas de inserción (**INSERT**), actualización (**UPDATE**) y eliminación (**DELETE**) para modificar los datos dentro de las tablas.
  - Ejemplo: Crear una consulta que actualice la dirección de un cliente en la base de datos o elimine un pedido que no fue procesado.

4. **Implementar estructuras de datos relacionales utilizando lenguaje de definición de datos (DDL) a partir de un modelo de datos para la creación y mantención de las definiciones de los objetos de una base de datos.**
  - Utilizar el lenguaje DDL para crear, modificar y eliminar tablas, índices y otros objetos dentro de una base de datos.
  - Ejemplo: Crear una tabla para almacenar información de empleados, con las columnas correspondientes como nombre, salario y fecha de ingreso.
5. **Elaborar un modelo de datos de acuerdo a los estándares de modelamiento para resolver un problema de baja complejidad.**
  - Crear un diagrama entidad-relación (ER) para representar el modelo de datos antes de implementarlo en una base de datos.
  - Ejemplo: Crear un modelo de datos para una tienda en línea, que incluya entidades como productos, clientes, pedidos y métodos de pago, y sus respectivas relaciones.

## Entrega

### 1. Repositorio en GitHub:

- Subir todos los archivos SQL y diagramas de modelado de datos en carpetas organizadas por tema.
- Incluir un archivo **README.md** en cada carpeta explicando el propósito del código y cómo ejecutar las consultas.
- Realizar al menos tres commits documentando los cambios realizados en el código.

## Estado de la entrega

Estado de la entrega	Todavía no se han realizado envíos
Estado de la calificación	Sin calificar
Última modificación	-

## DESARROLLO M5\_EVALUACIÓN DE PORTAFOLIO

### Gestión de Contratos de Servicios

Este portafolio corresponde al “**Módulo 5: Fundamentos de bases de datos relacionales**” del el **curso de Desarrollo de Aplicaciones Full Stack JavaScript**. En él se presenta la evidencia requerida según los criterios de evaluación para demostrar el dominio en el diseño, implementación y consulta de bases de datos relacionales en PostgreSQL.

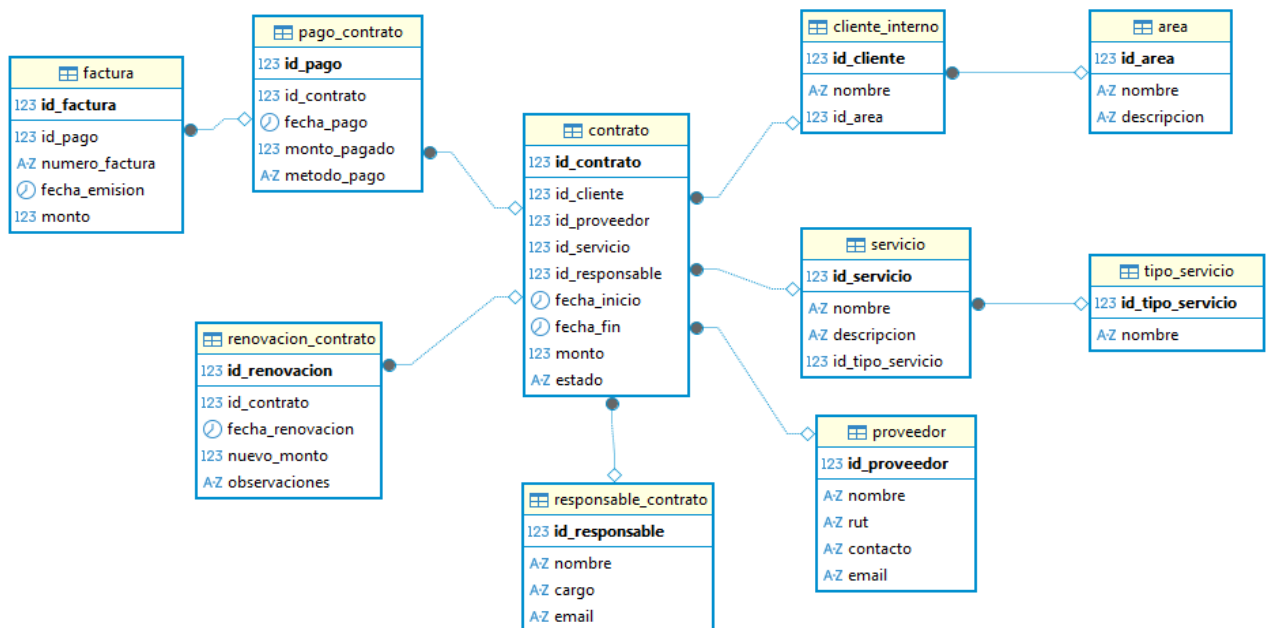
#### Punto 1: Fundamentos de Bases de Datos Relacionales

- **Rol:** Una base de datos relacional organiza la información en tablas relacionadas entre sí, garantizando integridad, consistencia y acceso seguro a los datos.
- **Componentes básicos:**
  - ✓ **Tabla:** estructura que agrupa datos en filas y columnas.
  - ✓ **Registro:** fila que representa una instancia de dato.
  - ✓ **Campo:** columna que define una propiedad (ejemplo: nombre, fecha).
  - ✓ **Clave primaria:** identifica de manera única a cada registro.
  - ✓ **Clave foránea:** crea relaciones entre tablas.
- **Gestión:** los datos se almacenan en tablas, se relacionan mediante claves, y se gestionan con instrucciones SQL (DDL, DML, SELECT).

#### Punto 2: Modelo de Datos (ERD)

- El sistema de gestión de contratos contempla entidades principales: clientes internos, proveedores, tipos de servicio, servicios, responsable de contrato, renovaciones y facturas. Estas se relacionan mediante claves foráneas, siguiendo un modelo entidad-relación normalizado.

Archivo relacionado: ERD GestionContratos.png



### Punto 3: Definición de Tablas (DDL)

- Se crearon las tablas mediante sentencias DDL, asegurando la definición de claves primarias y foráneas. Ejemplo: la tabla CONTRATO tiene PK en 'id' y FK hacia CLIENTE\_INTERNO, PROVEEDOR, SERVICIO y RESPONSABLE\_CONTRATO.

Archivo relacionado: 01\_ddl.sql

### Punto 4: Manipulación de Datos (DML)

- Se insertaron datos de prueba representativos, permitiendo simular contratos activos, pagos realizados y diferentes tipos de servicios.

Archivo relacionado: 02\_dml.sql

### Punto 5: Consultas SQL, Funciones y Procedimientos

- Se realizaron consultas SQL para obtener información de:
  - ✓ Listado de contratos vigentes.
  - ✓ Pagos asociados a un contrato.
  - ✓ Totales contratados por proveedor.
  - ✓ Contratos vigentes por área.
  - ✓ Se incluyen funciones y procedimientos:
    - fn\_total\_pagado\_contrato(contrato\_id INT) -> devuelve el total pagado de un contrato.
    - sp\_contratos\_area(area\_id INT) -> lista contratos vigentes de un área específica.

Archivo relacionado: 03\_queries.sql

### Evidencia Práctica

- Al ejecutar las consultas se espera obtener resultados que vinculen correctamente las tablas, por ejemplo: listar contratos con nombre del proveedor y servicio contratado. Si se intenta insertar un contrato con un proveedor inexistente, la FK evita la operación, demostrando la integridad referencial.
- Otros ejemplos ejecutados:
  1. Obtener contratos vigentes de TI: CALL sp\_contratos\_area(2);
  2. Total pagado de un contrato: SELECT fn\_total\_pagado\_contrato(1);
  3. Listado de contratos con pagos y facturas:

```
SELECT    c.id_contrato AS contrato, ci.nombre AS cliente, p.nombre AS proveedor,
          f.numero_factura, pc.monto_pagado
FROM      contrato c
JOIN      cliente_interno ci ON ci.id_cliente = c.id_cliente
JOIN      proveedor p      ON p.id_proveedor = c.id_proveedor
JOIN      pago_contrato pc  ON pc.id_contrato = c.id_contrato
JOIN      factura f        ON f.id_pago     = pc.id_pago;
```

- Esto demuestra integridad referencial, relaciones entre tablas permitiendo con ello lo siguiente:
  - Evitar duplicidad de información.
  - Mantener consistencia en los datos.
  - Realizar consultas complejas que unan información de diferentes tablas (por ejemplo, "todos los pedidos de un cliente con sus productos y cantidades").
- En resumen, las bases de datos relacionales permiten organizar, relacionar y consultar datos de manera eficiente, adaptándose a las necesidades de la organización y facilitando la gestión de información confiable.

## DESARROLLO DE CÓDIGO SQL:

### a) Definición de Tablas (DDL): 01\_ddl.sql

-- Creación de Base de Datos

**CREATE DATABASE** GestionContratos;

-- Tablas principales existentes

```
CREATE TABLE area (
  id_area      SERIAL PRIMARY KEY,
  nombre       VARCHAR(100) NOT NULL,
  descripción  VARCHAR(60)
);
```

```
CREATE TABLE cliente_interno (
  id_cliente   SERIAL PRIMARY KEY,
  nombre       VARCHAR(60) NOT NULL,
  id_area      INT REFERENCES area(id_area)
);
```

```
CREATE TABLE proveedor (
  id_proveedor SERIAL PRIMARY KEY,
  nombre       VARCHAR(60) NOT NULL,
  rut          VARCHAR(10) UNIQUE NOT NULL,
  contacto     VARCHAR(60),
  email        VARCHAR(60)
);
```

```
CREATE TABLE tipo_servicio (
  id_tipo_servicio SERIAL PRIMARY KEY,
  nombre           VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE servicio (
  id_servicio   SERIAL PRIMARY KEY,
  nombre        TEXT NOT NULL,
  descripción    TEXT,
  id_tipo_servicio INT REFERENCES tipo_servicio(id_tipo_servicio)
);
```

```

CREATE TABLE responsable_contrato (
  id_responsable SERIAL PRIMARY KEY,
  nombre VARCHAR(60) NOT NULL,
  cargo VARCHAR(100),
  email VARCHAR(60)
);

CREATE TABLE contrato (
  id_contrato SERIAL PRIMARY KEY,
  id_ INT REFERENCES cliente_interno(id_cliente),
  id_proveedor INT REFERENCES proveedor(id_proveedor),
  id_servicio INT REFERENCES servicio(id_servicio),
  id_responsable INT REFERENCES responsable_contrato(id_responsable),
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL,
  monto NUMERIC(12,2) NOT NULL,
  estado TEXT NOT NULL CHECK (estado IN ('Vigente','Finalizado','Suspendido'))
);

CREATE TABLE pago_contrato (
  id_pago SERIAL PRIMARY KEY,
  id_contrato INT REFERENCES contrato(id_contrato),
  fecha_pago DATE NOT NULL,
  monto_pagado NUMERIC(12,2) NOT NULL,
  metodo_pago TEXT
);

CREATE TABLE renovacion_contrato (
  id_renovacion SERIAL PRIMARY KEY,
  id_contrato INT REFERENCES contrato(id_contrato),
  fecha_renovacion DATE NOT NULL,
  nuevo_monto NUMERIC(12,2),
  observaciones TEXT
);

CREATE TABLE factura (
  id_factura SERIAL PRIMARY KEY,
  id_pago INT REFERENCES pago_contrato(id_pago),
  numero_factura VARCHAR(50) UNIQUE NOT NULL,
  fecha_emision DATE NOT NULL,
  monto NUMERIC(12,2) NOT NULL
);

```

**b) Manipulación de Datos (DML): 02\_dml.sql**

```

-- Areas
INSERT INTO area (nombre, descripcion) VALUES
('RRHH', 'Recursos Humanos'),
('TI' , 'Tecnologías de la Información'),
('SGDI' , 'Oficina de Documentación'),
('DFZA' , 'Departamento de Finanzas');

```

– Clientes internos

```
INSERT INTO cliente_interno (nombre, id_area) VALUES
('Departamento RRHH' , 1),
('Departamento TI' , 2),
('Registratura' , 3),
('Sección Contabilidad', 4);
```

– Proveedores

```
INSERT INTO proveedor (nombre, rut, contacto, email) VALUES
('Limpio S.A.' , '76123456-7' , 'José Fuentes' , 'contacto@limpio.cl'),
('Computación S.A.' , '77987654-3' , 'Pedro Sepúlveda' , 'conctado@compu.cl'),
('Defontana S.A.' , '78456123-1' , 'Luis Troncoso' , 'conctado@defontana.cl'),
('Jardines Verdes Ltda.' , '79789890-1' , 'Gonzalo Molina' , 'conctado@jardines.cl');
```

– Tipo de servicios

```
INSERT INTO tipo_servicio (nombre) VALUES
('Aseo'),
('Informática'),
('Contabilidades'),
('Jardinería');
```

– Servicios

```
INSERT INTO servicio (nombre, descripcion, id_tipo_servicio) VALUES
('Aseo Oficina', 'Limpieza general de oficinas' , 1),
('Arriendo PCs' , 'Arriendo de PCs, Notebook' , 2),
('Auditorias' , 'Mantenimiento de áreas verdes' , 3),
('Areas Verdes', 'Mantenimiento de áreas verdes' , 4);
```

– Responsable de contratos

```
INSERT INTO responsable_contrato (nombre, cargo, email) VALUES
('Alejandro González', 'Coordinador TI' , 'alejandro.gonzalez@empresa.cl'),
('Irene Silva' , 'Jefe Finanzas' , 'irene.silva@empresa.cl'),
('José Cifuentes' , 'Cooridnador RRHH' , 'jose.cifuentes@empresa.cl');
```

– Contratos

```
INSERT INTO contrato (id_cliente, id_proveedor, id_servicio, id_responsable, fecha_inicio,
fecha_fin, monto, estado) VALUES
(1, 1, 1, 1, '2025-01-01', '2025-12-31', 1500000, 'Vigente'),
(2, 1, 1, 1, '2025-01-01', '2025-12-31', 2500000, 'Vigente'),
(3, 2, 2, 2, '2025-03-01', '2025-09-30', 800000, 'Vigente');
```

– Pagos

```
INSERT INTO pago_contrato (id_contrato, fecha_pago, monto_pagado, metodo_pago)
VALUES
(1, '2025-02-01', 125000, 'Transferencia'),
(2, '2025-03-01', 125000, 'Transferencia'),
(3, '2025-04-15', 200000, 'Cheque');
```

– Renovaciones

```
INSERT INTO renovacion_contrato (id_contrato, fecha_renovacion, nuevo_monto,
observaciones) VALUES
(1, '2025-07-01', 1600000, 'Ajuste de tarifas'),
(2, '2025-09-01', 850000, 'Extensión temporal');
```

-- Facturas

```
INSERT INTO factura (id_pago, numero_factura, fecha_emision, monto) VALUES
(1, 'FAC-1001', '2025-02-02', 125000),
(2, 'FAC-1002', '2025-03-02', 125000),
(3, 'FAC-2001', '2025-04-16', 200000);
```

### c) Consultas SQL, Funciones y Procedimientos: 03\_queries.sql

-- Consultas básicas

```
SELECT c.id_cliente, ci.nombre AS cliente, p.nombre AS proveedor, s.nombre AS servicio,
c.fecha_inicio, c.fecha_fin, c.monto
FROM contrato c
JOIN cliente_interno ci ON ci.id_cliente = c.id_cliente
JOIN proveedor p ON p.id_proveedor = c.id_proveedor
JOIN servicio s ON s.id_servicio = c.id_servicio
WHERE c.estado = 'Vigente';
```

contrato(+) 1 X

SELECT c.id\_cliente, ci.nombre AS cliente, p.nombre AS proveedor, s.nombre AS servicio, c.fecha\_inicio, c.fecha\_fin, c.monto

	123 id_cliente	AZ cliente	AZ proveedor	AZ servicio	fecha_inicio	fecha_fin	123 monto
1	2	Departamento TI	Limpio S.A.	Aseo Oficina	2025-01-01	2025-12-31	2.500.000
2	1	Departamento RRHH	Limpio S.A.	Aseo Oficina	2025-01-01	2025-12-31	1.500.000
3	3	Registratura	Computación S.A.	Arrindo PCs	2025-03-01	2025-09-30	800.000

-- Actualizar contacto de un proveedor

```
UPDATE proveedor SET contacto = 'José Cifuentes', email='jcfuentes@limpio.cl'
WHERE id_proveedor=1;
```

-- Registros antes del cambio

proveedor 1 X

select \* from Proveedor

	123 id_proveedor	AZ nombre	AZ rut	AZ contacto	AZ email
1	1	Limpio S.A.	76123456-7	José Fuentes	contacto@limpio.cl

-- Registro actualizado

proveedor 1 X

select \* from PROVEEDOR order by ID\_PROVI

	123 id_proveedor	AZ nombre	AZ rut	AZ contacto	AZ email
1	1	Limpio S.A.	76123456-7	José Cifuentes	jcfuentes@limpio.cl



-- Eliminar registro 3 de la tabla area

```
DELETE area WHERE id_rea =3;
```

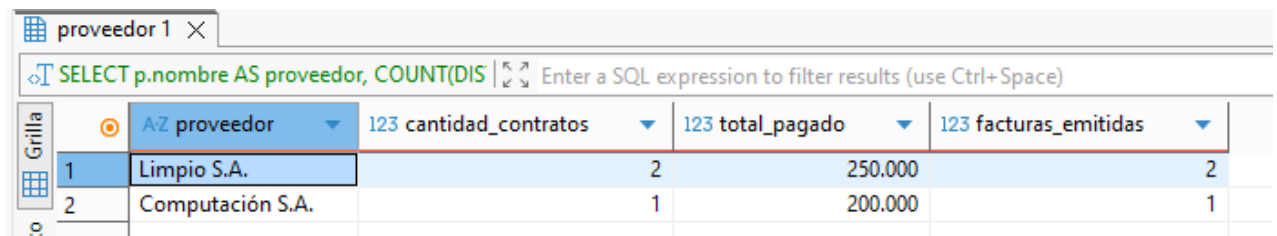
-- También se puede efectuar en cascada para ello debemos es definir las foreign keys con  
-- ON DELETE CASCADE.

-- Eso asegura que cuando borras en la tabla padre, los registros hijos se eliminen  
-- automáticamente.

```
ALTER TABLE cliente_interno
DROP CONSTRAINT IF EXISTS fk_area_cliente,
ADD CONSTRAINT fk_area_cliente
    FOREIGN KEY (area_id) REFERENCES area(id)
    ON DELETE CASCADE;
```

-- Consulta: Total pagado por proveedor con detalle de facturas

```
SELECT p.nombre AS proveedor,
       COUNT(DISTINCT c.id_contrato) AS cantidad_contratos,
       SUM(pc.monto_pagado) AS total_pagado,
       COUNT(DISTINCT f.id_factura) AS facturas_emitidas
FROM proveedor p
JOIN contrato c ON c.id_proveedor = p.id_proveedor
JOIN pago_contrato pc ON pc.id_contrato = c.id_contrato
LEFT JOIN factura f ON f.id_pago = pc.id_pago
WHERE c.estado = 'Vigente'
GROUP BY p.nombre
ORDER BY total_pagado DESC;
```



	AZ proveedor	123 cantidad_contratos	123 total_pagado	123 facturas_emitidas
1	Limpio S.A.	2	250.000	2
2	Computación S.A.	1	200.000	1

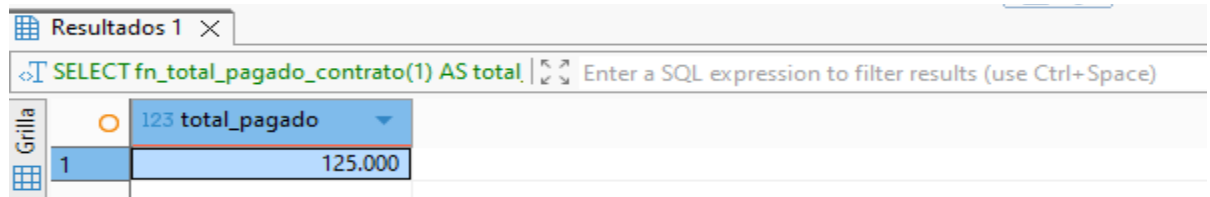
### Utilización de funciones y procedimientos almacenados.

-- Consultar total pagado por contrato

```
CREATE OR REPLACE FUNCTION fn_total_pagado_contrato(cid INT)
RETURNS NUMERIC AS $$
DECLARE
    total NUMERIC;
BEGIN
    SELECT SUM(monto_pagado) INTO total FROM pago_contrato WHERE id_contrato = cid;
    RETURN COALESCE(total,0);
END;
$$ LANGUAGE plpgsql;
```

-- Invocar function:

```
SELECT fn_total_pagado_contrato(1) AS total_pagado;
```



Resultados 1	
SELECT fn_total_pagado_contrato(1) AS total   Enter a SQL expression to filter results (use Ctrl+Space)	
Grilla	total_pagado
1	125.000

-- Procedimiento para listar contratos de un área

```
CREATE OR REPLACE PROCEDURE sp_contratos_area(aid INT)
```

```
LANGUAGE plpgsql
```

```
AS $$
```

```
DECLARE
```

```
    rec RECORD;
```

```
BEGIN
```

```
    RAISE NOTICE 'Contratos vigentes del área %:', aid;
```

```
    FOR rec IN
```

```
        SELECT c.id_contrato AS contrato,  
               ci.nombre    AS cliente,  
               s.nombre     AS servicio,  
               c.monto
```

```
        FROM contrato c
```

```
        JOIN cliente_interno ci ON ci.id_cliente = c.id_cliente
```

```
        JOIN servicio s       ON s.id_servicio = c.id_servicio
```

```
        WHERE ci.id_area = aid AND c.estado = 'Vigente'
```

```
    LOOP
```

```
        RAISE NOTICE 'Contrato %- Cliente: %, Servicio: %, Monto: %',  
                       rec.contrato, rec.cliente, rec.servicio, rec.monto;
```

```
    END LOOP;
```

```
END;
```

```
$$;
```

-- Llamado procedimiento:

```
CALL sp_contratos_area(2);
```

<postgres> 01\_ddl.sql

<postgres> 02\_dml.sql

\*public

<postgres> 03\_queries.sql

49

ci.nombre AS cliente,

50

s.nombre AS servicio,

51

c.monto

52

FROM contrato c

53

JOIN cliente\_interno ci ON ci.id\_cliente = c.id\_cliente

54

JOIN servicio s ON s.id\_servicio = c.id\_servicio

55

WHERE ci.id\_area = aid AND c.estado = 'Vigente'

56

LOOP

57

RAISE NOTICE 'Contrato % - Cliente: %, Servicio: %, Mont

58

rec.contrato, rec.cliente, rec.servicio, re

59

END LOOP;

60

END;

61

\$\$\$;

62

....

63

64

-- llamado procedimiento:

65

CALL sp\_contratos\_area(2);

66

67

68

Salida

Enter a part of a message to search for here

Contratos vigentes del área 2:

Contratos vigentes del área 2:

Contratos vigentes del área 2:

Contratos vigentes del área 2:

Contratos vigentes del área 2:

Contrato 2 - Cliente: Departamento TI, Servicio: Aseo Oficina

Contratos vigentes del área 2:

Contrato 2 - Cliente: Departamento TI, Servicio: Aseo Oficina

Estadísticas 1

Name	Value
Updated Rows	0
Execute time	0.001s
Start time	Fri Sep 05 23:55:25 CLT 2025
Finish time	Fri Sep 05 23:55:25 CLT 2025
Query	CALL sp_contratos_area(2)