HITO 1 DEL 2º TRIMESTRE DE BASE DE DATOS

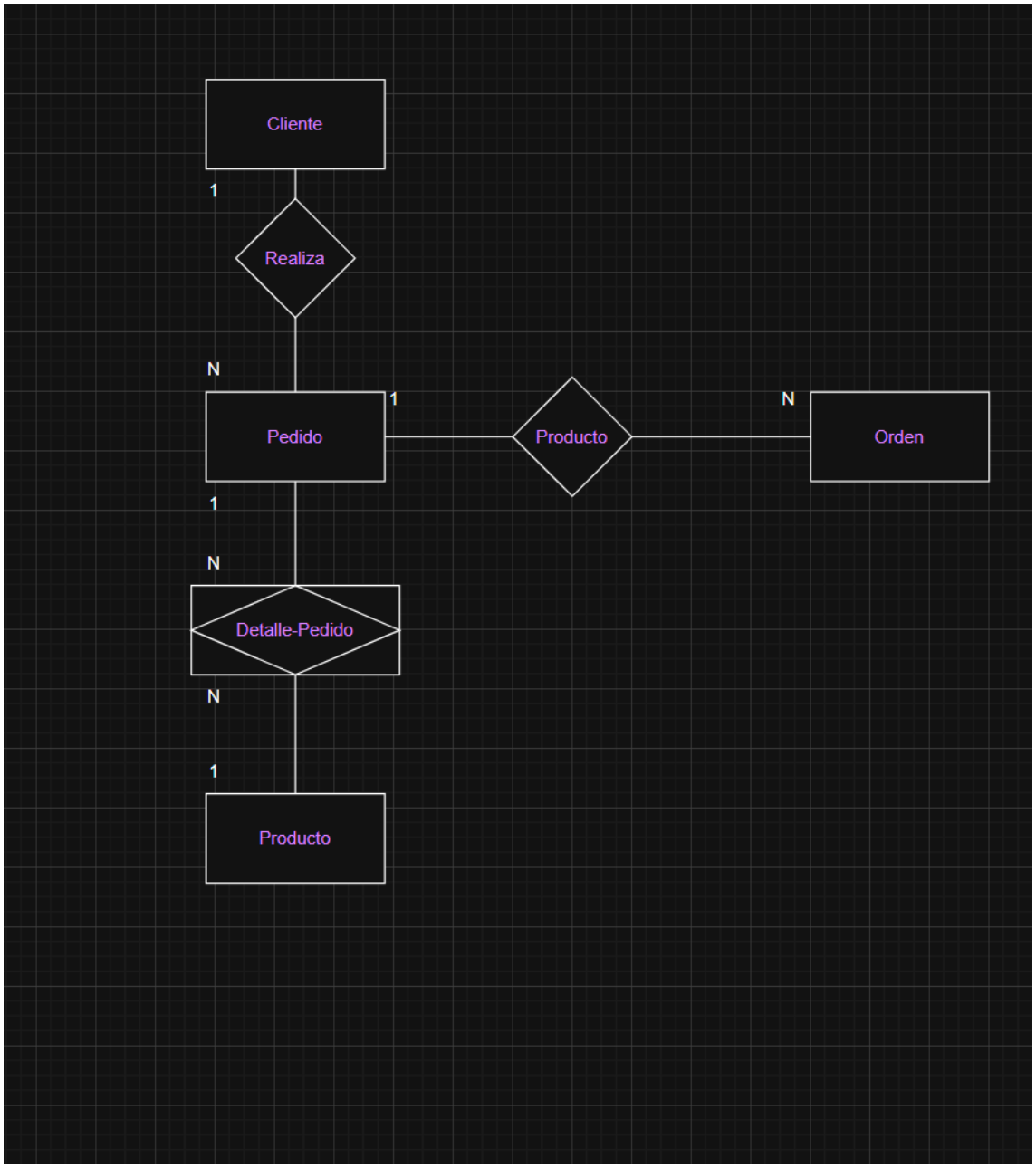
Alejandro Cortés Díaz

DISEÑO DE APLICACIONES MULTIPLATAFORMA – BASE DE DATOS 31/01/2025

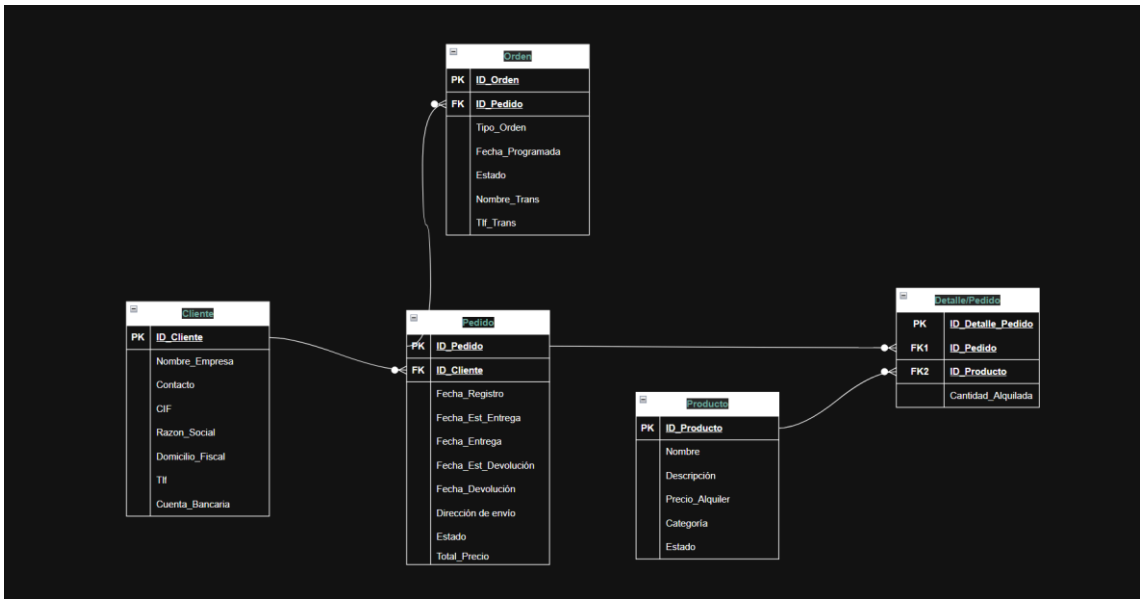
Índice

DESARROLLO

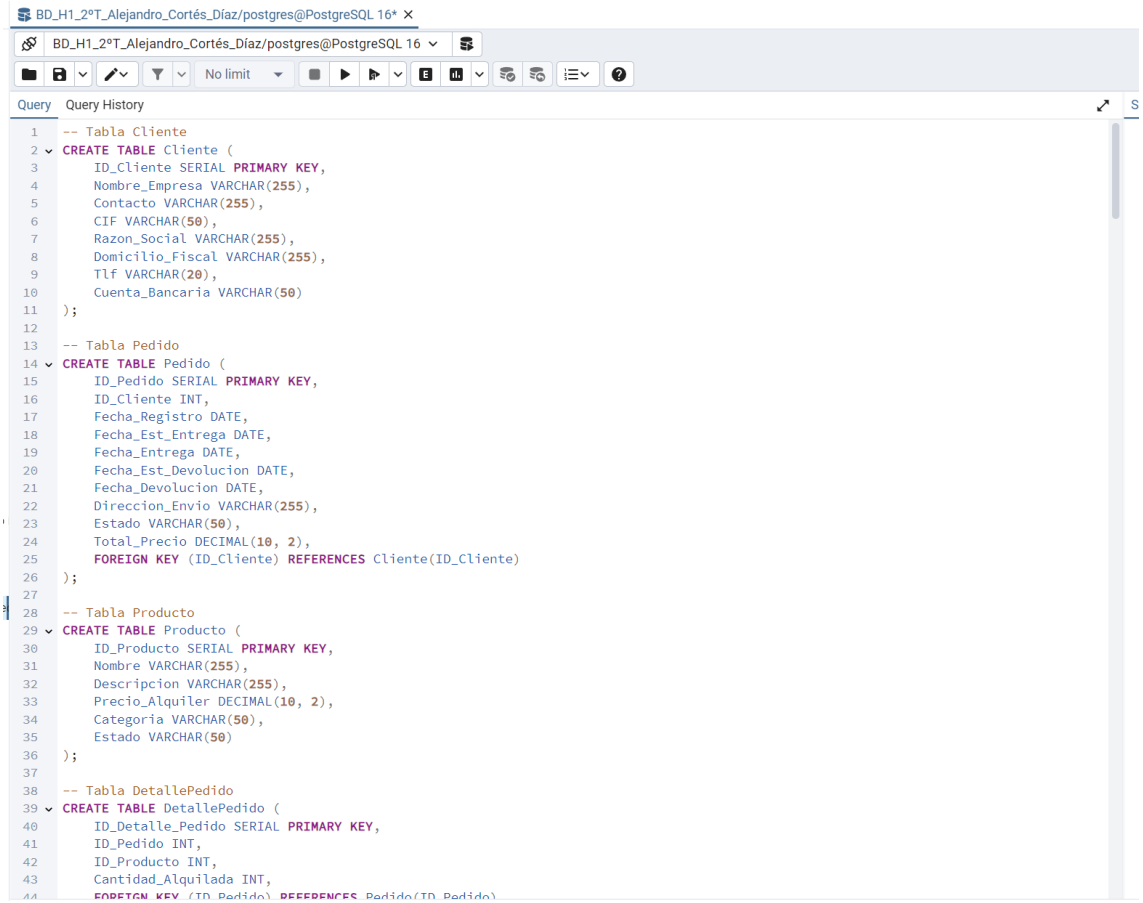
1. Realiza el diseño conceptual mediante un diagrama de Entidad-Relación. Puedes apoyarte en alguna herramienta de diagramado tipo Draw.IO o similar.



2. Realiza el diseño lógico estableciendo correctamente las claves principales y claves extranjeras para asegurar la integridad referencial.



3. Realiza el diseño físico de la base de datos mediante las sentencias SQL tipo CREATE que consideres necesarias.




```sql
-- Tabla DetallePedido
CREATE TABLE DetallePedido (
    ID_Detalle_Pedido SERIAL PRIMARY KEY,
    ID_Pedido INT,
    ID_Producto INT,
    Cantidad_Alquilada INT,
    FOREIGN KEY (ID_Pedido) REFERENCES Pedido(ID_Pedido),
    FOREIGN KEY (ID_Producto) REFERENCES Producto(ID_Producto)
);

-- Tabla Orden
CREATE TABLE Orden (
    ID_Orden SERIAL PRIMARY KEY,
    ID_Pedido INT,
    Tipo_Orden VARCHAR(50),
    Fecha_Programada DATE,
    Estado VARCHAR(50),
    Nombre_Trans VARCHAR(255),
    Tlf_Trans VARCHAR(20),
    FOREIGN KEY (ID_Pedido) REFERENCES Pedido(ID_Pedido)
);
```

Ver

4. Introduce datos de prueba suficientes para comprobar la eficiencia del Sistema y ejecuta las primeras selects para realizar una primera comprobación del buen funcionamiento de la BD.

```
INSERT INTO Cliente (Nombre_Empresa, Contacto, CIF, Razon_Social, Domicilio_Fiscal, Tlf, Cuenta_Bancaria)
VALUES
('Empresa A', 'Juan Pérez', 'CIF12345A', 'Empresa A S.A.', 'Calle Mayor 1', '600123456', 'ES1234567890123456789012'),
('Empresa B', 'María López', 'CIF67890B', 'Empresa B S.L.', 'Calle Menor 2', '600654321', 'ES9876543210987654321098'),
('Empresa C', 'Carlos Ruiz', 'CIF54321C', 'Empresa C S.A.', 'Calle Central 3', '600789123', 'ES3456789012345678901234'),
('Empresa D', 'Ana Martín', 'CIF98765D', 'Empresa D S.L.', 'Calle Norte 4', '600987654', 'ES4567890123456789012345'),
('Empresa E', 'Pedro Gómez', 'CIF45678E', 'Empresa E S.A.', 'Calle Este 5', '600456789', 'ES5678901234567890123456'),
('Empresa F', 'Laura Sánchez', 'CIF87654F', 'Empresa F S.L.', 'Calle Oeste 6', '600321987', 'ES6789012345678901234567'),
('Empresa G', 'Luis García', 'CIF23456G', 'Empresa G S.A.', 'Calle Sur 7', '600654123', 'ES7890123456789012345678'),
('Empresa H', 'Marta Díaz', 'CIF76543H', 'Empresa H S.L.', 'Calle Nueva 8', '600987321', 'ES8901234567890123456789'),
('Empresa I', 'Elena Rodríguez', 'CIF65432I', 'Empresa I S.A.', 'Calle Antigua 9', '600123654', 'ES9012345678901234567890'),
('Empresa J', 'Fernando Torres', 'CIF34567J', 'Empresa J S.L.', 'Calle Vieja 10', '600456123', 'ES0123456789012345678901'),
('Empresa K', 'Carlos Martínez', 'CIF99999K', 'Empresa K S.L.', 'Calle Ficticia 11', '600123789', 'ES9999999999999999999'),
('Empresa L', 'Lucía Gómez', 'CIF11111L', 'Empresa L S.A.', 'Calle Ficticia 12', '600234567', 'ES0123456789012345678901');


INSERT INTO Producto (Nombre, Descripcion, Precio_Alquiler, Categoria, Estado)
VALUES
('Cámara básica', 'Cámara fotográfica de baja calidad', 15.00, 'Categoría A', 'Disponible'),
('Cámara de bolsillo', 'Cámara pro de tamaño bolsillo', 20.00, 'Categoría B', 'Disponible'),
('Mueble victoriano', 'Mueble victoriano de pega', 10.00, 'Categoría C', 'Disponible'),
('Traje de gala', 'Traje de gala negro', 25.00, 'Categoría A', 'No Disponible'),
('Brújula pirata', 'Brújula pirata de pega', 30.00, 'Categoría B', 'Disponible'),
('Bastón grande', 'Bastón innecesariamente grande', 12.00, 'Categoría C', 'No Disponible'),
('Sombrero de copa', 'Sombrero de copa', 18.00, 'Categoría A', 'Disponible'),
('Copa aristocrática', 'Copa aristocrática con detalles dorados', 22.00, 'Categoría B', 'Disponible'),
('Roca de cartón', 'Roca de cartón - piedra', 16.00, 'Categoría C', 'Disponible'),
('Lianas de pega', 'Lianas de pega', 24.00, 'Categoría A', 'Disponible');
```

```
INSERT INTO DetallePedido (ID_Pedido, ID_Producto, Cantidad_Alquilada)
VALUES
(1, 1, 2),
(1, 2, 1),
(2, 3, 5),
(2, 4, 1),
(3, 5, 3),
(4, 6, 2),
(5, 7, 1),
(6, 8, 4),
(7, 9, 2),
(8, 10,1),
(9, 1, 2),
(10,4, 1),
(11,2, 2),
(12,5, 3);

INSERT INTO Orden (ID_Pedido, Tipo_Orden, Fecha_Programada, Estado, Nombre_Trans, Tlf_Trans)
VALUES
(1, 'Entrega', '2025-01-05', 'Pendiente', 'Manolo el del Bolo', '600111222'),
(2, 'Devolución', '2025-01-06', 'Pendiente', 'Pepita Rita', '600333444'),
(3, 'Entrega', '2025-01-07', 'Completada', 'Francisco Fiasco', '600555666'),
(4, 'Devolución', '2025-01-08', 'Completada', 'Benito Manito', '600777888'),
(5, 'Entrega', '2025-01-09', 'Pendiente', 'Ernesta Fiesta', '600999000'),
(6, 'Devolución', '2025-01-10', 'Pendiente', 'Lola Trola', '600123456'),
(7, 'Entrega', '2025-01-11', 'Pendiente', 'Paco Urraco', '600654321'),
(8, 'Entrega', '2025-01-12', 'Pendiente', 'Pablo Diablo', '600987654'),
(9, 'Devolución', '2025-01-13', 'Completada', 'Rubenico Pánico', '600321987'),
(10, 'Entrega', '2025-01-14', 'Pendiente', 'Ramón Jamón', '600654987'),
(11, 'Entrega', '2025-01-15', 'Pendiente', 'Carlos Fernández', '600321654'),
(12, 'Devolución', '2025-01-20', 'Pendiente', 'Luis García', '600987321');
```

```
INSERT INTO Pedido (ID_Cliente, Fecha_Registro, Fecha_Est_Entrega, Fecha_Entrega, Fecha_Est_Devolucion, Fecha_Devolucion, Direccion_Envio, Estado, Total_Pre
VALUES
(1, '2025-01-01', '2025-01-05', '2025-01-05', '2025-01-10', NULL, 'Calle Mayor 1', 'Devuelto', 90.00),
(2, '2025-01-02', '2025-01-06', '2025-01-06', '2025-01-11', NULL, 'Calle Menor 2', 'Devuelto', 160.00),
(3, '2025-01-03', '2025-01-07', '2025-01-07', '2025-01-12', NULL, 'Calle Central 3', 'Entregado', 120.00),
(4, '2025-01-04', '2025-01-08', '2025-01-08', '2025-01-13', '2025-01-13', 'Calle Norte 4', 'Devuelto', 24.00),
(5, '2025-01-05', '2025-01-09', '2025-01-09', '2025-01-14', NULL, 'Calle Este 5', 'Devuelto', 72.00),
(6, '2025-01-06', '2025-01-10', '2025-01-10', '2025-01-15', '2025-01-15', 'Calle Oeste 6', 'Entregado', 144.00),
(7, '2025-01-07', '2025-01-11', '2025-01-11', '2025-01-16', NULL, 'Calle Sur 7', 'Devuelto', 80.00),
(8, '2025-01-08', '2025-01-12', '2025-01-12', '2025-01-17', NULL, 'Calle Nueva 8', 'Devuelto', 78.00),
(9, '2025-01-09', '2025-01-13', '2025-01-13', '2025-01-18', '2025-01-18', 'Calle Antigua 9', 'Entregado', 90.00),
(10, '2025-01-10', '2025-01-14', '2025-01-14', '2025-01-19', NULL, 'Calle Vieja 10', 'Devuelto', 24.00),
(11, '2025-01-11', '2025-01-15', '2025-01-15', '2025-01-20', '2025-01-20', 'Calle Este 11', 'Devuelto', 60.00),
(12, '2025-01-12', '2025-01-16', '2025-01-16', '2025-01-21', '2025-01-21', 'Calle Oeste 12', 'Devuelto', 110.00);
```

Realizamos la comprobación de la db a continuación:

```
select * from cliente
```

	id_cliente [PK] integer	nombre_empresa character varying (255)	contacto character varying (255)	cif character varying (50)	razon_social character varying (255)	domicilio_fiscal character varying (255)	tlf character varying (20)	cuenta_bancaria character varying (50)
1	1	Empresa A	Juan Pérez	CIF12345A	Empresa A S.A.	Calle Mayor 1	600123456	ES1234567890123456789012
2	2	Empresa B	María López	CIF67890B	Empresa B S.L.	Calle Menor 2	600654321	ES9876543210987654321098
3	3	Empresa C	Carlos Ruiz	CIF54321C	Empresa C S.A.	Calle Central 3	600789123	ES3456789012345678901234
4	4	Empresa D	Ana Martín	CIF98765D	Empresa D S.L.	Calle Norte 4	600987654	ES4567890123456789012345
5	5	Empresa E	Pedro Gómez	CIF45678E	Empresa E S.A.	Calle Este 5	600456789	ES5678901234567890123456
6	6	Empresa F	Laura Sánchez	CIF87654F	Empresa F S.L.	Calle Oeste 6	600321987	ES6789012345678901234567
7	7	Empresa G	Luis García	CIF23456G	Empresa G S.A.	Calle Sur 7	600654123	ES7890123456789012345678
8	8	Empresa H	Marta Díaz	CIF76543H	Empresa H S.L.	Calle Nueva 8	600987321	ES8901234567890123456789
9	9	Empresa I	Elena Rodríguez	CIF65432I	Empresa I S.A.	Calle Antigua 9	600123654	ES9012345678901234567890
10	10	Empresa J	Fernando Torres	CIF34567J	Empresa J S.L.	Calle Vieja 10	600456123	ES0123456789012345678901
11	11	Empresa K	Carlos Martínez	CIF99999K	Empresa K S.L.	Calle Ficticia 11	600123789	ES9999999999999999999
12	12	Empresa L	Lucía Gómez	CIF11111L	Empresa L S.A.	Calle Ficticia 12	600234567	ES0123456789012345678901

```
select * from detallepedido
```

	id_detalle_pedido [PK] integer	id_pedido integer	id_producto integer	cantidad_alquilada integer
1	1	1	1	2
2	2	1	2	1
3	3	2	3	5
4	4	2	4	1
5	5	3	5	3
6	6	4	6	2
7	7	5	7	1
8	8	6	8	4
9	9	7	9	2
10	10	8	10	1
11	21	9	1	2
12	22	10	4	1
13	23	11	2	2
14	24	12	5	3

```
select * from orden
```

	id_orden [PK] integer	id_pedido integer	tipo_orden character varying (50)	fecha_programada date	estado character varying (50)	nombre_trans character varying (255)	tlf_trans character varying (20)
1	1	1	Entrega	2025-01-05	Pendiente	Manolo el del Bolo	600111222
2	2	2	Devolución	2025-01-06	Pendiente	Pepita Rita	600333444
3	3	3	Entrega	2025-01-07	Completada	Francisco Fiasco	600555666
4	4	4	Devolución	2025-01-08	Completada	Benito Manito	600777888
5	5	5	Entrega	2025-01-09	Pendiente	Ernesta Fiesta	600999000
6	6	6	Devolución	2025-01-10	Pendiente	Lola Trola	600123456
7	7	7	Entrega	2025-01-11	Pendiente	Paco Urraco	600654321
8	8	8	Entrega	2025-01-12	Pendiente	Pablo Diablo	600987654
9	9	9	Devolución	2025-01-13	Completada	Rubenico Pánico	600321987
10	10	10	Entrega	2025-01-14	Pendiente	Ramón Jamón	600654987
11	11	11	Entrega	2025-01-15	Pendiente	Carlos Fernández	600321654
12	12	12	Devolución	2025-01-20	Pendiente	Luis García	600987321

```
select * from pedido
```

	id_pedido [PK] integer	id_cliente integer	fecha_registro date	fecha_est_entrega date	fecha_entrega date	fecha_est_devolucion date	fecha_devolucion date	direccion_envio character varying (255)	estado character varying (50)	total_precio numeric (10,2)
1	1	1	2025-01-01	2025-01-05	2025-01-05	2025-01-10	[null]	Calle Mayor 1	Devuelto	90.00
2	2	2	2025-01-02	2025-01-06	2025-01-06	2025-01-11	[null]	Calle Menor 2	Devuelto	160.00
3	3	3	2025-01-03	2025-01-07	2025-01-07	2025-01-12	[null]	Calle Central 3	Entregado	120.00
4	4	4	2025-01-04	2025-01-08	2025-01-08	2025-01-13	2025-01-13	Calle Norte 4	Devuelto	24.00
5	5	5	2025-01-05	2025-01-09	2025-01-09	2025-01-14	[null]	Calle Este 5	Devuelto	72.00
6	6	6	2025-01-06	2025-01-10	2025-01-10	2025-01-15	2025-01-15	Calle Oeste 6	Entregado	144.00
7	7	7	2025-01-07	2025-01-11	2025-01-11	2025-01-16	[null]	Calle Sur 7	Devuelto	80.00
8	8	8	2025-01-08	2025-01-12	2025-01-12	2025-01-17	[null]	Calle Nueva 8	Devuelto	78.00
9	9	9	2025-01-09	2025-01-13	2025-01-13	2025-01-18	2025-01-18	Calle Antigua 9	Entregado	90.00
10	10	10	2025-01-10	2025-01-14	2025-01-14	2025-01-19	[null]	Calle Vieja 10	Devuelto	24.00
11	11	11	2025-01-11	2025-01-15	2025-01-15	2025-01-20	2025-01-20	Calle Este 11	Devuelto	60.00
12	12	12	2025-01-12	2025-01-16	2025-01-16	2025-01-21	2025-01-21	Calle Oeste 12	Devuelto	110.00

```
select * from producto
```

	id_producto [PK] integer	nombre character varying (255)	descripcion character varying (255)	precio_alquiler numeric (10,2)	categoria character varying (50)	estado character varying (50)
1	1	Cámara básica	Cámara fotográfica de baja calidad	15.00	Categoría A	Disponible
2	2	Cámara de bolsillo	Cámara pro de tamaño bolsillo	20.00	Categoría B	Disponible
3	3	Mueble victoriano	Mueble victoriano de pega	10.00	Categoría C	Disponible
4	4	Traje de gala	Traje de gala negro	25.00	Categoría A	No Disponible
5	5	Brújula pirata	Brújula pirata de pega	30.00	Categoría B	Disponible
6	6	Bastón grande	Bastón innecesariamente grande	12.00	Categoría C	No Disponible
7	7	Sombrero de copa	Sombrero de copa	18.00	Categoría A	Disponible
8	8	Copa aristocrática	Copa aristocrática con detalles dorados	22.00	Categoría B	Disponible
9	9	Roca de cartón	Roca de cartón - piedra	16.00	Categoría C	Disponible
10	10	Lianas de pega	Lianas de pega	24.00	Categoría A	Disponible

5. Realiza consultas en múltiples tablas. Debes plantear las consultas que vayan a ser útiles para la futura aplicación web. Mínimo 20 consultas SQL.

```
-- 1. Consultar los pedidos junto con los detalles de los productos alquilados -- Esto nos permite saber en general datos relativos a las ventas
-- realizadas con las cantidades obtenidas de los productos.
SELECT p.ID_Pedido, p.Fecha_Registro, dp.ID_Producto, pr.Nombre, dp.Cantidad_Alquilada
FROM Pedido p
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
INNER JOIN Producto pr ON dp.ID_Producto = pr.ID_Producto;
```

	id_pedido integer	fecha_registro date	id_producto integer	nombre character varying (255)	cantidad_alquilada integer
1	1	2025-01-01	1	Cámara básica	2
2	1	2025-01-01	2	Cámara de bolsillo	1
3	2	2025-01-02	3	Mueble victoriano	5
4	2	2025-01-02	4	Traje de gala	1
5	3	2025-01-03	5	Brújula pirata	3
6	4	2025-01-04	6	Bastón grande	2
7	5	2025-01-05	7	Sombrero de copa	1
8	6	2025-01-06	8	Copa aristocrática	4
9	7	2025-01-07	9	Roca de cartón	2
10	8	2025-01-08	10	Lianas de pega	1
11	9	2025-01-09	1	Cámara básica	2
12	10	2025-01-10	4	Traje de gala	1
13	11	2025-01-11	2	Cámara de bolsillo	2
14	12	2025-01-12	5	Brújula pirata	3

```sql
-- 2. Consultar el total de productos alquilados y su precio total por cliente -- Nos posibilita saber cuánto han gastado en nosotros las empresas
-- que figurancomo nuestros clientes.
SELECT c.Nombre_Empresa, SUM(dp.Cantidad_Alquilada * pr.Precio_Alquiler) AS Total_Precio
FROM Cliente c
INNER JOIN Pedido p ON c.ID_Cliente = p.ID_Cliente
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
INNER JOIN Producto pr ON dp.ID_Producto = pr.ID_Producto
GROUP BY c.ID_Cliente;
```

```sql
-- 3. Consultar los clientes que tienen pedidos con productos en estado 'No Disponible' -- De esta manera vemos cuál es el estado de nuestro stock
```

	nombre_empresa character varying (255)	total_precio numeric
1	Empresa D	24.00
2	Empresa J	25.00
3	Empresa I	30.00
4	Empresa G	32.00
5	Empresa F	88.00
6	Empresa L	90.00
7	Empresa C	90.00
8	Empresa A	50.00
9	Empresa E	18.00
10	Empresa B	75.00
11	Empresa K	40.00
12	Empresa H	24.00

```sql
-- 3. Consultar los clientes que tienen pedidos con productos en estado 'No Disponible' -- De esta manera vemos cuál es el estado de nuestro stock
-- en relación a los productos deseados.
SELECT DISTINCT c.Nombre_Empresa, p.ID_Pedido
FROM Cliente c
INNER JOIN Pedido p ON c.ID_Cliente = p.ID_Cliente
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
INNER JOIN Producto pr ON dp.ID_Producto = pr.ID_Producto
WHERE pr.Estado = 'No Disponible';
```

	nombre_empresa character varying (255)	id_pedido integer
1	Empresa J	10
2	Empresa D	4
3	Empresa B	2

```sql
-- 4. Consultar todos los productos alquilados en pedidos que  están entregados -- Para ver el índice de pedidos que hemos entregados
SELECT pr.Nombre, dp.Cantidad_Alquilada, p.Estado
FROM Producto pr
INNER JOIN DetallePedido dp ON pr.ID_Producto = dp.ID_Producto
INNER JOIN Pedido p ON dp.ID_Pedido = p.ID_Pedido
WHERE p.Estado = 'Entregado';
```

	nombre character varying (255)	cantidad_alquilada integer	estado character varying (50)
1	Copa aristocrática	4	Entregado
2	Cámara básica	2	Entregado

```sql
-- 5. Consultar todos los pedidos con la información de la orden de entrega -- Vemos el tipo de orden de entrega, y por otro lado,la fecha en la que están programadas.
SELECT p.ID_Pedido, o.Tipo_Orden, o.Fecha_Programada
FROM Pedido p
INNER JOIN Orden o ON p.ID_Pedido = o.ID_Pedido
WHERE o.Tipo_Orden = 'Entrega';
```

	id_pedido integer	tipo_orden character varying (50)	fecha_programada date
1	1	Entrega	2025-01-05
2	3	Entrega	2025-01-07
3	5	Entrega	2025-01-09
4	7	Entrega	2025-01-11
5	8	Entrega	2025-01-12
6	10	Entrega	2025-01-14
7	11	Entrega	2025-01-15

```sql
-- 6. Consultar las órdenes que están asociadas a pedidos con una fecha de entrega mayor al 10 de enero. Así podemos ver el incremento o detrimento de
-- los pedidos a partir de esta fecha
SELECT o.ID_Orden, p.ID_Pedido, o.Fecha_Programada
FROM Orden o
INNER JOIN Pedido p ON o.ID_Pedido = p.ID_Pedido
WHERE p.Fecha_Entrega > '2025-01-10';
```

	id_orden integer	id_pedido integer	fecha_programada date
1	7	7	2025-01-11
2	8	8	2025-01-12
3	9	9	2025-01-13
4	10	10	2025-01-14
5	11	11	2025-01-15
6	12	12	2025-01-20

```sql
-- 7. Consultar los productos alquilados junto con su categoria y estado. Una forma de ver los productos junto con sus categorias y cantidades
--, además de si se encuentran disponibles o no, en este momento.
SELECT pr.Nombre, pr.Categoria, pr.Estado, dp.Cantidad_Alquilada
FROM Producto pr
INNER JOIN DetallePedido dp ON pr.ID_Producto = dp.ID_Producto;
```

	nombre character varying (255)	categoria character varying (50)	estado character varying (50)	cantidad_alquilada integer
1	Cámara básica	Categoría A	Disponible	2
2	Cámara de bolsillo	Categoría B	Disponible	1
3	Mueble victoriano	Categoría C	Disponible	5
4	Traje de gala	Categoría A	No Disponible	1
5	Brújula pirata	Categoría B	Disponible	3
6	Bastón grande	Categoría C	No Disponible	2
7	Sombrero de copa	Categoría A	Disponible	1
8	Copa aristocrática	Categoría B	Disponible	4
9	Roca de cartón	Categoría C	Disponible	2
10	Lianas de pega	Categoría A	Disponible	1
11	Cámara básica	Categoría A	Disponible	2
12	Traje de gala	Categoría A	No Disponible	1
13	Cámara de bolsillo	Categoría B	Disponible	2
14	Brújula pirata	Categoría B	Disponible	3

```sql
-- 8. Consultar el total de precios por pedido, incluyendo el nombre del cliente. Nos permite saber qué clientes gastaron qué dinero, junto con el id
-- de su pedido
SELECT p.ID_Pedido, c.Nombre_Empresa, SUM(dp.Cantidad_Alquilada * pr.Precio_Alquiler) AS Total_Precio
FROM Pedido p
INNER JOIN Cliente c ON p.ID_Cliente = c.ID_Cliente
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
INNER JOIN Producto pr ON dp.ID_Producto = pr.ID_Producto
GROUP BY p.ID_Pedido, c.Nombre_Empresa;
```

	id_pedido integer	nombre_empresa character varying (255)	total_precio numeric
1	3	Empresa C	90.00
2	5	Empresa E	18.00
3	6	Empresa F	88.00
4	8	Empresa H	24.00
5	1	Empresa A	50.00
6	10	Empresa J	25.00
7	11	Empresa K	40.00
8	2	Empresa B	75.00
9	9	Empresa I	30.00
10	7	Empresa G	32.00
11	4	Empresa D	24.00
12	12	Empresa L	90.00

```sql
-- 9. Consultar los pedidos que incluyen productos de una categoría específica. Observamos todos los productos y su categoría, junto con
-- el id que los pidió
SELECT p.ID_Pedido, pr.Nombre, pr.Categoria
FROM Pedido p
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
INNER JOIN Producto pr ON dp.ID_Producto = pr.ID_Producto
WHERE pr.Categoria = 'Categoría A';
```

	id_pedido integer	nombre character varying (255)	categoria character varying (50)
1	1	Cámara básica	Categoría A
2	2	Traje de gala	Categoría A
3	5	Sombrero de copa	Categoría A
4	8	Lianas de pega	Categoría A
5	9	Cámara básica	Categoría A
6	10	Traje de gala	Categoría A

```sql
-- 10. Consultar los productos que han sido alquilados más de 3 veces. En este caso 4 productos destacan de nuestro stock, sobre otros
SELECT pr.Nombre, SUM(dp.Cantidad_Alquilada) AS Total_Alquilado
FROM Producto pr
INNER JOIN DetallePedido dp ON pr.ID_Producto = dp.ID_Producto
GROUP BY pr.ID_Producto, pr.Nombre
HAVING SUM(dp.Cantidad_Alquilada) > 3;
```

	nombre character varying (255)	total_alquilado bigint
1	Mueble victoriano	5
2	Cámara básica	4
3	Brújula pirata	6
4	Copa aristocrática	4

```sql
-- 11. Consultar las órdenes y los detalles del cliente que la solicitó. De esta manera vemos los pedidos que realizaron nuestros clientes,
-- en este caso realizó cada uno, un pedido.
SELECT o.ID_Orden, p.ID_Pedido, c.Nombre_Empresa
FROM Orden o
INNER JOIN Pedido p ON o.ID_Pedido = p.ID_Pedido
INNER JOIN Cliente c ON p.ID_Cliente = c.ID_Cliente;
```

	id_orden integer	id_pedido integer	nombre_empresa character varying (255)
1	1	1	Empresa A
2	2	2	Empresa B
3	3	3	Empresa C
4	4	4	Empresa D
5	5	5	Empresa E
6	6	6	Empresa F
7	7	7	Empresa G
8	8	8	Empresa H
9	9	9	Empresa I
10	10	10	Empresa J
11	11	11	Empresa K
12	12	12	Empresa L

```sql
-- 12. Consultar los productos alquilados con su precio de alquiler y cantidad, solo los que están disponibles. Vemos el precio de cada producto alquilado
-- que además, está disponible, cámara básica, cámara de bolsillo, y brújula pirata, fueron alquilados 2 veces.
SELECT pr.Nombre, pr.Precio_Alquiler, dp.Cantidad_Alquilada
FROM Producto pr
INNER JOIN DetallePedido dp ON pr.ID_Producto = dp.ID_Producto
WHERE pr.Estado = 'Disponible';
```

	nombre character varying (255)	precio_alquiler numeric (10,2)	cantidad_alquilada integer
1	Cámara básica	15.00	2
2	Cámara de bolsillo	20.00	1
3	Mueble victoriano	10.00	5
4	Brújula pirata	30.00	3
5	Sombrero de copa	18.00	1
6	Copa aristocrática	22.00	4
7	Roca de cartón	16.00	2
8	Lianas de pega	24.00	1
9	Cámara básica	15.00	2
10	Cámara de bolsillo	20.00	2
11	Brújula pirata	30.00	3

```sql
-- 13. Consultar los clientes y la cantidad total de productos que han alquilado. Observamos cúantas veces han alquilado algún producto nuestros clientes.
SELECT c.Nombre_Empresa, SUM(dp.Cantidad_Alquilada) AS Total_Alquilado
FROM Cliente c
INNER JOIN Pedido p ON c.ID_Cliente = p.ID_Cliente
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
GROUP BY c.ID_Cliente;
```

	nombre_empresa character varying (255)	total_alquilado bigint
1	Empresa D	2
2	Empresa J	1
3	Empresa I	2
4	Empresa G	2
5	Empresa F	4
6	Empresa L	3
7	Empresa C	3
8	Empresa A	3
9	Empresa E	1
10	Empresa B	6
11	Empresa K	2
12	Empresa H	1

```sql
-- 14. Consultar los pedidos con sus productos, fechas de entrega y devolución. Observamos cómo nuestros productos han sido solicitados en cada producto.
-- Aparecen también aquellos que aún no han sido devueltos, y por tanto aparece null, así como unificados, los productos que fueron pedidos
-- en un mismo id pedido
SELECT
    p.ID_Pedido,
    STRING_AGG(pr.Nombre, ', ') AS Productos,
    p.Fecha_Entrega,
    p.Fecha_Devolucion
FROM Pedido p
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
INNER JOIN Producto pr ON dp.ID_Producto = pr.ID_Producto
GROUP BY p.ID_Pedido, p.Fecha_Entrega, p.Fecha_Devolucion
ORDER BY p.ID_Pedido;
```

	id_pedido [PK] integer	productos text	fecha_entrega date	fecha_devolucion date
1	1	Cámara básica, Cámara de bolsillo	2025-01-05	2025-01-30
2	2	Mueble victoriano, Traje de gala	2025-01-06	2025-01-30
3	3	Brújula pirata	2025-01-07	2025-01-30
4	4	Bastón grande	2025-01-08	2025-01-13
5	5	Sombrero de copa	2025-01-09	2025-01-30
6	6	Copa aristocrática	2025-01-10	2025-01-15
7	7	Roca de cartón	2025-01-11	[null]
8	8	Lianas de pega	2025-01-12	[null]
9	9	Cámara básica	2025-01-13	2025-01-18
10	10	Traje de gala	2025-01-14	[null]
11	11	Cámara de bolsillo	2025-01-15	2025-01-20
12	12	Brújula pirata	2025-01-16	2025-01-21

```sql
-- 15. Consultar el estado de las órdenes junto con el nombre del transportista. Vemos el estado de las órdenes junto con el estaod del mismo pedido,
-- el nombre del tranportista, su contacto, y en general detalles de interés sobre la clase de pedido y orden ante la que estamos.
SELECT o.Estado, o.Nombre_Trans, o.Tlf_Trans, p.Estado AS Estado_Pedido, c.Nombre_Empresa
FROM Orden o
INNER JOIN Pedido p ON o.ID_Pedido = p.ID_Pedido
INNER JOIN Cliente c ON p.ID_Cliente = c.ID_Cliente;
```

	estado character varying (50)	nombre_trans character varying (255)	tlf_trans character varying (20)	estado_pedido character varying (50)	nombre_empresa character varying (255)
1	Pendiente	Manolo el del Bolo	600111222	Devuelto	Empresa A
2	Pendiente	Pepita Rita	600333444	Devuelto	Empresa B
3	Completada	Francisco Fiasco	600555666	Devuelto	Empresa C
4	Completada	Benito Manito	600777888	Devuelto	Empresa D
5	Pendiente	Ernesta Fiesta	600999000	Devuelto	Empresa E
6	Pendiente	Lola Trola	600123456	Entregado	Empresa F
7	Pendiente	Paco Urraco	600654321	Devuelto	Empresa G
8	Pendiente	Pablo Diablo	600987654	Devuelto	Empresa H
9	Completada	Rubenico Pánico	600321987	Entregado	Empresa I
10	Pendiente	Ramón Jamón	600654987	Devuelto	Empresa J
11	Pendiente	Carlos Fernández	600321654	Devuelto	Empresa K
12	Pendiente	Luis García	600987321	Devuelto	Empresa L

```sql
-- 16. Consultar los pedidos de un cliente específico y sus detalles. Empresa A, al haber realizado dos pedidos diferentes, tenemos 2 productos por su cuenta asociados al mismo i
SELECT p.ID_Pedido, p.Fecha_Entrega, dp.Cantidad_Alquilada, pr.Nombre,c.Nombre_Empresa
FROM Cliente c
INNER JOIN Pedido p ON c.ID_Cliente = p.ID_Cliente
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
INNER JOIN Producto pr ON dp.ID_Producto = pr.ID_Producto
WHERE c.Nombre_Empresa = 'Empresa A';
```

	id_pedido integer	fecha_entrega date	cantidad_alquilada integer	nombre character varying (255)	nombre_empresa character varying (255)
1	1	2025-01-05	2	Cámara básica	Empresa A
2	1	2025-01-05	1	Cámara de bolsillo	Empresa A

```
-- 17. Consultar las órdenes de entrega y devolución, con su respectivo cliente. Observamos a nuestros clientes con sus ordenes y el tipo de la misma, en relación a los pedidos
-- realizados
SELECT o.ID_Orden, p.ID_Pedido, c.Nombre_Empresa, o.Tipo_Orden
FROM Orden o
INNER JOIN Pedido p ON o.ID_Pedido = p.ID_Pedido
INNER JOIN Cliente c ON p.ID_Cliente = c.ID_Cliente;
```

	id_orden integer	id_pedido integer	nombre_empresa character varying (255)	tipo_orden character varying (50)
1	1	1	Empresa A	Entrega
2	2	2	Empresa B	Devolución
3	3	3	Empresa C	Entrega
4	4	4	Empresa D	Devolución
5	5	5	Empresa E	Entrega
6	6	6	Empresa F	Devolución
7	7	7	Empresa G	Entrega
8	8	8	Empresa H	Entrega
9	9	9	Empresa I	Devolución
10	10	10	Empresa J	Entrega
11	11	11	Empresa K	Entrega
12	12	12	Empresa L	Devolución

```
-- 18. Consultar los productos y su cantidad total alquilada por todos los clientes. Podemos ver cuáles son los productos más populares en los hábitos de consumo de nuestra
-- clientela
SELECT pr.Nombre, SUM(dp.Cantidad_Alquilada) AS Total_Alquilado
FROM Producto pr
INNER JOIN DetallePedido dp ON pr.ID_Producto = dp.ID_Producto
GROUP BY pr.ID_Producto;
```

	nombre character varying (255)	total_alquilado bigint
1	Traje de gala	2
2	Lianas de pega	1
3	Bastón grande	2
4	Cámara de bolsillo	3
5	Roca de cartón	2
6	Sombrero de copa	1
7	Mueble victoriano	5
8	Cámara básica	4
9	Brújula pirata	6
10	Copa aristocrática	4

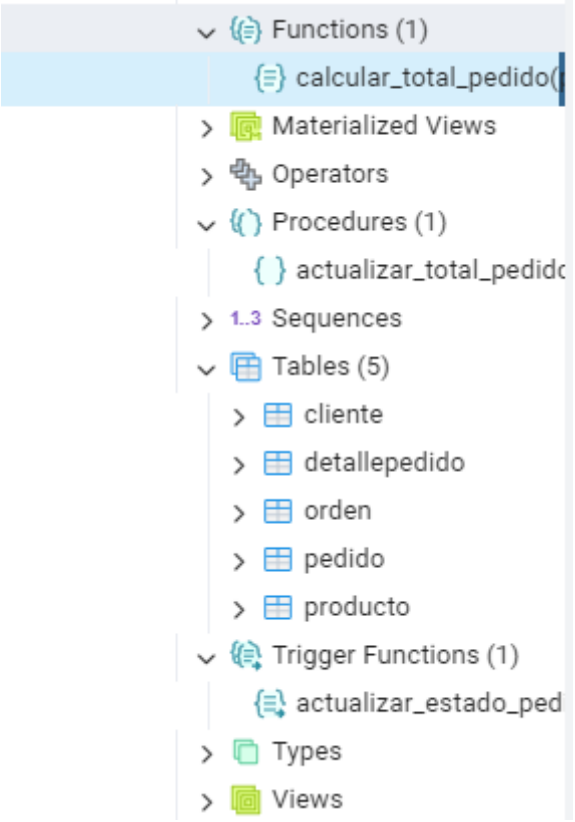
```sql
-- 19. Consultar las órdenes con la fecha de entrega y la cantidad de productos alquilados. Podemos ver el total de productos alquilados en cada orden junto con su fecha.
SELECT o.ID_Orden, o.Fecha_Programada, SUM(dp.Cantidad_Alquilada) AS Total_Productos
FROM Orden o
INNER JOIN Pedido p ON o.ID_Pedido = p.ID_Pedido
INNER JOIN DetallePedido dp ON p.ID_Pedido = dp.ID_Pedido
GROUP BY o.ID_Orden, o.Fecha_Programada;
```

	id_orden [PK] integer	fecha_programada date	total_productos bigint
1	4	2025-01-08	2
2	10	2025-01-14	1
3	6	2025-01-10	4
4	2	2025-01-06	6
5	11	2025-01-15	2
6	9	2025-01-13	2
7	7	2025-01-11	2
8	12	2025-01-20	3
9	3	2025-01-07	3
10	1	2025-01-05	3
11	5	2025-01-09	1
12	8	2025-01-12	1

```sql
-- 20. Consultar las órdenes asociadas a pedidos que no han sido devueltos. Podemos ver las ordenes donde la fecha de devolución, aún no ha sido registrada.
SELECT o.ID_Orden, p.ID_Pedido, p.Fecha_Devolucion
FROM Orden o
INNER JOIN Pedido p ON o.ID_Pedido = p.ID_Pedido
WHERE p.Fecha_Devolucion IS NULL;
```

	id_orden integer	id_pedido integer	fecha_devolucion date
1	7	7	[null]
2	8	8	[null]
3	10	10	[null]

6. Desarrolla un procedimiento, una función y un desencadenador que sean de utilidad para satisfacer algún requerimiento del usuario. Justifica en qué manera el trabajo realizado en este ejercicio es útil para el futuro proyecto.



Mi función es la siguiente;

```
CREATE OR REPLACE FUNCTION calcular_total_pedido(p_id_pedido INT)
RETURNS NUMERIC(10,2)
LANGUAGE plpgsql
AS $$
DECLARE
    v_total NUMERIC(10,2);
BEGIN
    -- Calcular el total sumando (cantidad * precio) de cada producto en el pedido
    SELECT COALESCE(SUM(dp.Cantidad_Alquilada * p.Precio_Alquiler), 0)
    INTO v_total
    FROM DetallePedido dp
    JOIN Producto p ON dp.ID_Producto = p.ID_Producto
    WHERE dp.ID_Pedido = p_id_pedido;

    RETURN v_total;
END;
$$;
```

Esta función, a partir del id_pedido que se introduciría entre paréntesis al utilizarla, se basa en este dato para mostrar el valor total del pedido, basándose en la cantidad alquilada del mismo en este pedido, y del valor que tiene de alquiler el producto en sí. Y el producto de esta multiplicación, resulta en el valor total del proceso de pedido realizado.

```
CREATE OR REPLACE FUNCTION calcular_total_pedido(p_id_pedido INT)
RETURNS NUMERIC(10,2)
LANGUAGE plpgsql
AS $$
DECLARE
    v_total NUMERIC(10,2);
BEGIN
    -- Calcular el total sumando (cantidad * precio) de cada producto en el pedido
    SELECT COALESCE(SUM(dp.Cantidad_Alquilada * p.Precio_Alquiler), 0)
    INTO v_total
    FROM DetallePedido dp
    JOIN Producto p ON dp.ID_Producto = p.ID_Producto
    WHERE dp.ID_Pedido = p_id_pedido;

    RETURN v_total;
END;
$$;
```

Para realizar un ejemplo de aplicación, a través de un select de los pedios, dispongo de los precios totales de los pedidos;

4	1	1	2025-01-01	2025-01-05	2025-01-05	2025-01-10	2025-01-30	Calle Mayor 1	Devuelto	50.00
5	2	2	2025-01-02	2025-01-06	2025-01-06	2025-01-11	2025-01-30	Calle Menor 2	Devuelto	75.00
6	3	3	2025-01-03	2025-01-07	2025-01-07	2025-01-12	2025-01-30	Calle Central 3	Devuelto	90.00
7	4	4	2025-01-04	2025-01-08	2025-01-08	2025-01-13	2025-01-13	Calle Norte 4	Devuelto	24.00
8	5	5	2025-01-05	2025-01-09	2025-01-09	2025-01-14	2025-01-30	Calle Este 5	Devuelto	18.00
9	6	6	2025-01-06	2025-01-10	2025-01-10	2025-01-15	2025-01-15	Calle Oeste 6	Entregado	88.00
10	7	7	2025-01-07	2025-01-11	2025-01-11	2025-01-16	2025-01-30	Calle Sur 7	Devuelto	32.00
11	8	8	2025-01-08	2025-01-12	2025-01-12	2025-01-17	2025-01-30	Calle Nueva 8	Devuelto	24.00
12	9	9	2025-01-09	2025-01-13	2025-01-13	2025-01-18	2025-01-18	Calle Antigua 9	Entregado	30.00

No obstante, he insertado 3 pedidos con un precio incorrecto de base, para poder mostrar la utilidad de esta función:

1	10	10	2025-01-10	2025-01-14	2025-01-14	2025-01-19	[null]	Calle Vieja 10	Devuelto	24.00
2	11	11	2025-01-11	2025-01-15	2025-01-15	2025-01-20	2025-01-20	Calle Este 11	Devuelto	60.00
3	12	12	2025-01-12	2025-01-16	2025-01-16	2025-01-21	2025-01-21	Calle Oeste 12	Devuelto	110.00

- Usaré el pedido con el id 11, que tiene supuestamente 60€ de valor total, no obstante, en la tabla detalle – pedido, se ha realizado la siguiente operación:

```
(10,4, 1),
(11,2, 2),
(12,5, 3);
```

Con el id_pedido 11, se alquiló el producto con el id_producto 2, en cantidad de 2, en vista de que el producto con el id 2, cuesta por sí solo, 20€, debería ser 40, el valor total, y no 60.

```
INSERT INTO Producto (Nombre, Descripcion, Precio_Alquiler, Categoria, Estado)
VALUES
('Cámara básica', 'Cámara fotográfica de baja calidad', 15.00, 'Categoría A', 'Disponible'),
('Cámara de bolsillo', 'Cámara pro de tamaño bolsillo', 20.00, 'Categoría B', 'Disponible'),
```

Así que, vamos a utilizar la función para comprobar cuál es el coste real de este pedido:

```
select calcular_total_pedido(11)
```

	calcular_total_pedido numeric
1	40.00

- De esta manera, podemos observar si hay algún error con el valor del pedido realizado, en procedimiento aprovecha esta función para arreglar este dato, a continuación, continuo con él.

```
CREATE OR REPLACE PROCEDURE actualizar_total_pedido(p_id_pedido INT)
LANGUAGE plpgsql
AS $$
DECLARE
    v_total NUMERIC(10,2);
BEGIN
    -- Calcular el total del pedido usando la función calcular_total_pedido
    SELECT calcular_total_pedido(p_id_pedido) INTO v_total;

    -- Actualizar el total en la tabla Pedido
    UPDATE Pedido
    SET Total_Precio = v_total
    WHERE ID_Pedido = p_id_pedido;
END;
$$;
```

- Este procedimiento, emplea la función para calcular al total de un pedido, para, en base a este resultado verdadero, cambiar la tabla pedido, concretamente el campo de "Total_Precio", y sustituirlo por el valor total que arroja la función, donde coincida con el id_pedido, de esta forma, podemos solventar problemas donde no cuadren los precios de los productos, y los presuntos pedidos con su cifra que la acompaña.

- Echemos un vistazo al select * de pedidos para observar cómo cambiarían los datos en los campos mencionados, tal y como se comenta:

	id_pedido [PK] integer	id_cliente integer	fecha_registro date	fecha_est_entrega date	fecha_entrega date	fecha_est_devolucion date	fecha_devolucion date	direccion_envio character varying (255)	estado character varying (50)	total_precio numeric (10,2)
1	10	10	2025-01-10	2025-01-14	2025-01-14	2025-01-19	[null]	Calle Vieja 10	Devuelto	24.00
2	11	11	2025-01-11	2025-01-15	2025-01-15	2025-01-20	2025-01-20	Calle Este 11	Devuelto	60.00
3	12	12	2025-01-12	2025-01-16	2025-01-16	2025-01-21	2025-01-21	Calle Oeste 12	Devuelto	110.00
4	1	1	2025-01-01	2025-01-05	2025-01-05	2025-01-10	2025-01-30	Calle Mayor 1	Devuelto	50.00
5	2	2	2025-01-02	2025-01-06	2025-01-06	2025-01-11	2025-01-30	Calle Menor 2	Devuelto	75.00
6	3	3	2025-01-03	2025-01-07	2025-01-07	2025-01-12	2025-01-30	Calle Central 3	Devuelto	90.00
7	4	4	2025-01-04	2025-01-08	2025-01-08	2025-01-13	2025-01-13	Calle Norte 4	Devuelto	24.00
8	5	5	2025-01-05	2025-01-09	2025-01-09	2025-01-14	2025-01-30	Calle Este 5	Devuelto	18.00
9	6	6	2025-01-06	2025-01-10	2025-01-10	2025-01-15	2025-01-15	Calle Oeste 6	Entregado	88.00
10	7	7	2025-01-07	2025-01-11	2025-01-11	2025-01-16	2025-01-30	Calle Sur 7	Devuelto	32.00
11	8	8	2025-01-08	2025-01-12	2025-01-12	2025-01-17	2025-01-30	Calle Nueva 8	Devuelto	24.00
12	9	9	2025-01-09	2025-01-13	2025-01-13	2025-01-18	2025-01-18	Calle Antigua 9	Entregado	30.00

- El precio continúa como 60, así que usamos el procedimiento:

```
call actualizar_total_pedido(11)
```

```
CALL

Query returned successfully in 89 msec.
```

	id_pedido [PK] integer	id_cliente integer	fecha_registro date	fecha_est_entrega date	fecha_entrega date	fecha_est_devolucion date	fecha_devolucion date	direccion_envio character varying (255)	estado character varying (50)	total_precio numeric (10,2)
1	10	10	2025-01-10	2025-01-14	2025-01-14	2025-01-19	[null]	Calle Vieja 10	Devuelto	24.00
2	12	12	2025-01-12	2025-01-16	2025-01-16	2025-01-21	2025-01-21	Calle Oeste 12	Devuelto	110.00
3	1	1	2025-01-01	2025-01-05	2025-01-05	2025-01-10	2025-01-30	Calle Mayor 1	Devuelto	50.00
4	2	2	2025-01-02	2025-01-06	2025-01-06	2025-01-11	2025-01-30	Calle Menor 2	Devuelto	75.00
5	3	3	2025-01-03	2025-01-07	2025-01-07	2025-01-12	2025-01-30	Calle Central 3	Devuelto	90.00
6	4	4	2025-01-04	2025-01-08	2025-01-08	2025-01-13	2025-01-13	Calle Norte 4	Devuelto	24.00
7	5	5	2025-01-05	2025-01-09	2025-01-09	2025-01-14	2025-01-30	Calle Este 5	Devuelto	18.00
8	6	6	2025-01-06	2025-01-10	2025-01-10	2025-01-15	2025-01-15	Calle Oeste 6	Entregado	88.00
9	7	7	2025-01-07	2025-01-11	2025-01-11	2025-01-16	2025-01-30	Calle Sur 7	Devuelto	32.00
10	8	8	2025-01-08	2025-01-12	2025-01-12	2025-01-17	2025-01-30	Calle Nueva 8	Devuelto	24.00
11	9	9	2025-01-09	2025-01-13	2025-01-13	2025-01-18	2025-01-18	Calle Antigua 9	Entregado	30.00
12	11	11	2025-01-11	2025-01-15	2025-01-15	2025-01-20	2025-01-20	Calle Este 11	Devuelto	40.00

- Ahora figura como 40, como debería ser.

- Por último, pasemos al desencadenador;

```
CREATE OR REPLACE FUNCTION actualizar_estado_pedido()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    -- Si la fecha de entrega es NULL y ya pasó la fecha estimada, marcar como "Entregado"
    IF NEW.Fecha_Entrega IS NULL AND NEW.Fecha_Est_Entrega <= CURRENT_DATE THEN
        NEW.Fecha_Entrega := CURRENT_DATE;
        NEW.Estado := 'Entregado';
    END IF;

    -- Si la fecha de devolución es NULL y ya pasó la fecha estimada, marcar como "Devuelto"
    IF NEW.Fecha_Devolucion IS NULL AND NEW.Fecha_Est_Devolucion <= CURRENT_DATE THEN
        NEW.Fecha_Devolucion := CURRENT_DATE;
        NEW.Estado := 'Devuelto';
    END IF;

    RETURN NEW;
END;
$$;

CREATE TRIGGER trg_actualizar_estado_pedido
BEFORE INSERT OR UPDATE ON Pedido
FOR EACH ROW
EXECUTE FUNCTION actualizar_estado_pedido();
```

- Este desencadenador, procesa un nuevo insert into, en el que la fecha estimada sea menor que la fecha actual, por lo cual, dado que no se habría modificado la fecha estimada suponiendo que haya un atraso por cualquier motivo, cambia la fecha de entrega a la fecha actual, y pasa a figurar como entregado.

- De esta manera no hay que revisarlo manualmente, sino que directamente el programa se encarga de, en el caso de que se dé una inserción que no tenga sentido, porque el campo de entrega siga “NULL”, pese a que la fecha estimada de entrega sea más tardía, automáticamente figure como “entregado”, antes si quiera de que se realice la modificación o inserción en la tabla pedido.

- Lo mismo aplica para aquellos pedidos donde la fecha estimada de devolución, no cuadre con aquella en la que supuestamente tendría que haberse devuelto, porque siga siendo “NULL”.

- Pongamos a prueba el funcionamiento del desencadenador:

```
INSERT INTO Pedido (ID_Cliente, Fecha_Registro, Fecha_Est_Entrega, Fecha_Entrega, Fecha_Est_Devolucion, Fecha_Devolucion, Direccion_Envio, Estado, Total_Precio)
VALUES (12, '2025-01-01', '2025-01-05', NULL, '2025-01-10', NULL, 'Calle Prueba 1', 'Pendiente', 120.00);
```

Tratamos de hacer este insert into, donde la fecha estimada de entrega es menor a la fecha actual, así como la fecha de devolución, cuando tratamos de insertarlo, este es el resultado:

```
INSERT 0 1

Query returned successfully in 157 msec.
```

25	26	12	2025-01-01	2025-01-05	2025-01-30	2025-01-10	2025-01-30	Calle Prueba 1	Devuelto	120.00

17	17	5	2025-01-05	2025-01-09	2025-01-09	2025-01-14	2025-01-29	Calle Este 5	Devuelto	72.00
18	18	6	2025-01-06	2025-01-10	2025-01-10	2025-01-15	2025-01-15	Calle Oeste 6	Entregado	144.00
19	19	7	2025-01-07	2025-01-11	2025-01-11	2025-01-16	2025-01-29	Calle Sur 7	Devuelto	80.00
20	20	8	2025-01-08	2025-01-12	2025-01-12	2025-01-17	2025-01-29	Calle Nueva 8	Devuelto	78.00
21	21	9	2025-01-09	2025-01-13	2025-01-13	2025-01-18	2025-01-18	Calle Antigua 9	Entregado	90.00
22	22	10	2025-01-10	2025-01-14	2025-01-14	2025-01-19	2025-01-29	Calle Vieja 10	Devuelto	24.00
23	23	11	2025-01-11	2025-01-15	2025-01-15	2025-01-20	2025-01-20	Calle Este 11	Devuelto	60.00
24	24	12	2025-01-12	2025-01-16	2025-01-16	2025-01-21	2025-01-21	Calle Oeste 12	Devuelto	110.00
25	26	12	2025-01-01	2025-01-05	2025-01-30	2025-01-10	2025-01-30	Calle Prueba 1	Devuelto	120.00

- Y la prueba de que ha funcionado correctamente, es que las fechas de entrega y devolución, han cambiado a la fecha actual, de esta forma, evitamos que haya valores nulls que puedan resultar confusos en un pedido ya pasado, pudiendo generar confusión.

Enlace a GitHub

https://github.com/Cortes-cmd/BBDD.git

Bibliografía

ChatGPT. (s/f). Chatgpt.com. Recuperado el 30 de enero de 2025, de https://chatgpt.com/c/679b7e09-8cf0-800d-b0cb-04b374319ed1

Database procedure. (s/f). Recuperado el 30 de enero de 2025, de https://maxdb.sap.com/doc/7_7/44/bd1ea5a5d51388e10000000a155369/content.htm

Developing and using stored procedures. (s/f). Oracle.com. Recuperado el 30 de enero de 2025, de https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm

Khan, S. (2024, septiembre 23). *Master your database skills: The definitive guide to stored procedures with real-world applications*. Medium. https://medium.com/@ksaquib/master-your-database-skills-the-definitive-guide-to-stored-procedures-with-real-world-applications-d74fd7dee395

SQL stored procedures. (s/f). W3schools.com. Recuperado el 30 de enero de 2025, de https://www.w3schools.com/sql/sql_stored_procedures.asp

What are the Microsoft SQL database functions? - SQL Server. (s/f). Microsoft.com. Recuperado el 30 de enero de 2025, de https://learn.microsoft.com/en-us/sql/t-sql/functions/functions?view=sql-server-ver16

(S/f). Sparxsystems.com. Recuperado el 30 de enero de 2025, de https://sparxsystems.com/enterprise_architect_user_guide/17.0/modeling_domains/database_functions.html