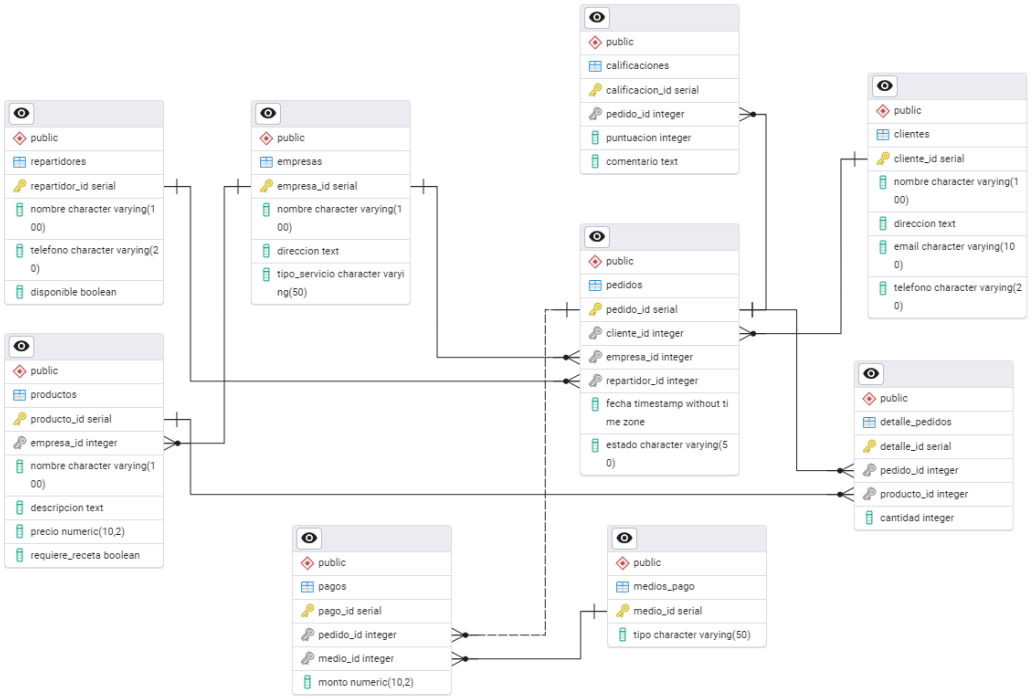
Capturas y resultados pruebas
Laboratorio 1
Taller de Base de Datos
1er Semestre 2025

Grupo 4
Alumnos:
Lucas Contador
Cristobal Milla
Alvaro Muñoz
Mauricio Rivera
Claudia Villa

Gráfico MR:



1. ¿Qué cliente ha gastado más dinero en pedidos entregados?

PSQL Tool Workspace

```sql
SELECT sp.id_cliente, c.nombre AS cliente, sp.num_pedidos AS num_pedidos_pagados, sp.suma_pagos
FROM clientes c
INNER JOIN (SELECT p.cliente_id AS id_cliente, COUNT(*) AS num_pedidos, SUM(pa.monto) AS suma_pagos
            FROM pedidos p
            INNER JOIN pagos pa ON p.pedido_id = pa.pedido_id
            WHERE p.estado = 'entregado'
            GROUP BY p.cliente_id) AS sp
ON c.cliente_id = sp.id_cliente
WHERE sp.suma_pagos = (SELECT MAX(suma_pagos) AS max_pagos
                       FROM (SELECT p.cliente_id, SUM(pa.monto) AS suma_pagos
                             FROM pedidos p
                             INNER JOIN pagos pa ON p.pedido_id = pa.pedido_id
                             WHERE p.estado = 'entregado'
                             GROUP BY p.cliente_id));
```

Data Output | Messages | Notifications

Showing rows: 1 to 1

	id_cliente integer	cliente character varying (100)	num_pedidos_pagados bigint	suma_pagos numeric
1	1	Ana Torres	1	9200.00

2. ¿Cuáles son los productos o servicios más pedidos en el último mes por categoría?

Query | Query History

```sql
SELECT categoria, producto_id, nombre, total_cantidad
FROM (
    SELECT
        p.categoria,
        p.producto_id,
        p.nombre,
        SUM(dp.cantidad) AS total_cantidad,
        RANK() OVER(PARTITION BY p.categoria ORDER BY SUM(dp.cantidad) DESC) AS rnk
    FROM pedidos pe
    INNER JOIN detalle_pedidos dp ON pe.pedido_id = dp.pedido_id
    INNER JOIN productos p ON dp.producto_id = p.producto_id
    WHERE pe.fecha >= CURRENT_DATE - INTERVAL '1 month'
    GROUP BY p.categoria, p.producto_id, p.nombre
)
WHERE rnk = 1
ORDER BY categoria;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2

	categoria character varying (100)	producto_id [PK] integer	nombre character varying (100)	total_cantidad bigint
1	Documentos	3	Envío carta notarial	1
2	Medicamentos	1	Paracetamol 500mg	2

Total rows: 2 | Query complete 00:00:00.501

3. Listar las empresas asociadas con más entregas fallidas.

Query | Query History

```sql
SELECT e.empresa_id, e.nombre AS empresa_nombre, COUNT(p.pedido_id) AS pedidos_cancelados_count
FROM empresas e
LEFT JOIN pedidos p ON e.empresa_id = p.empresa_id
WHERE p.estado = 'cancelado'
GROUP BY e.empresa_id, e.nombre
HAVING COUNT(p.pedido_id) > 0
ORDER BY pedidos_cancelados_count DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1

	empresa_id [PK] integer	empresa_nombre character varying (100)	pedidos_cancelados_count bigint
1	2	Express Documentos	1

Successfully run.

Total rows: 1 | Query complete 00:00:00.141

4. Calcular el tiempo promedio entre pedido y entrega por repartidor.

```
Query  Query History
1  UPDATE pedidos SET estado = 'entregado' WHERE pedido_id = 14;
2  UPDATE pedidos SET fecha = '2025-05-01 10:30:00' WHERE pedido_id = 14;
3
4
5  SELECT r.repartidor_id, r.nombre, r.telefono, r.disponible, AVG(EXTRACT(EPOCH FROM p.fecha_entrega - p.fecha) / 86400) as avg_tiempo_entrega_min
6  FROM repartidores as r
7  INNER JOIN pedidos as p ON r.repartidor_id = p.repartidor_id
8  WHERE p.estado = 'entregado'
9  GROUP BY r.repartidor_id;
```

Data Output | Messages | Notifications

Showing rows: 1 to 5 | Page No: 1

	repartidor_id [PK] integer	nombre character varying (100)	telefono character varying (20)	disponible boolean	avg_tiempo_entrega_min numeric
1	1	Carlos Gómez	911112223	true	8.5044560185185185
2	2	Sofía Rojas	922223334	true	[null]
3	3	Miguel Torres	933445566	true	[null]
4	4	Elena Mendoza	944556677	true	[null]
5	5	Raúl Vargas	955667788	true	[null]

5. Obtener los 3 repartidores con mejor rendimiento (basado en entregas y puntuación).

```
Query  Query History
1  SELECT r.nombre as repartidor, COUNT(p.pedido_id) as pedidos, AVG(c.puntuacion) as calificación_avg
2  FROM (repartidores as r
3  INNER JOIN pedidos as p ON r.repartidor_id = p.repartidor_id)
4  FULL JOIN calificaciones as c ON p.pedido_id = c.pedido_id
5  WHERE p.estado = 'entregado'
6  GROUP BY r.repartidor_id
7  ORDER BY COUNT(p.pedido_id) DESC, AVG(c.puntuacion) DESC
8  LIMIT 3;
```

Data Output | Messages | Notifications

Showing rows: 1 to 3

	repartidor character varying (100)	pedidos bigint	calificación_avg numeric
1	Miguel Torres	4	4.7500000000000000
2	Sofía Rojas	2	[null]
3	Elena Mendoza	2	4.5000000000000000

Total rows: 3 | Query complete 00:00:00.353

6. ¿Qué medio de pago se utiliza más en pedidos urgentes?

Query | Query History

```sql
SELECT mp.tipo AS medio_pago, COUNT(*) AS total_urgentes
FROM pagos pg
JOIN urgencias u ON pg.pedido_id = u.pedido_id
JOIN medios_pago mp ON pg.medio_id = mp.medio_id
WHERE u.nivel = 'urgente'
GROUP BY mp.tipo
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1

	medio_pago character varying (50)	total_urgentes bigint
1	Tarjeta de crédito	3

Total rows: 1 | Query complete 00:00:00.153

7. Registrar un pedido completo.

Query | Query History

```sql
CREATE OR REPLACE FUNCTION registrar_pedido(
	p_cliente_id INT,
	p_empresa_id INT,
	p_repartidor_id INT,
	p_estado VARCHAR,
	p_productos INT[], --lista de id de productos
	p_cantidades INT[],
	p_medio_pago_id INT
)
RETURNS INT AS $$
DECLARE
	nuevo_id INT; --id del pedido
	i INT; -- indice, empieza en 1
BEGIN
    -- primeros insertamos el pedido
    INSERT INTO pedidos (cliente_id, empresa_id, repartidor_id, fecha, estado)
    VALUES (p_cliente_id, p_empresa_id, p_repartidor_id, NOW()::timestamp(0), p_estado)
    RETURNING pedido_id INTO nuevo_id;

```

```
    -- ahora insertamos el detalle del pedido
    FOR I IN 1..array_length(p_productos, 1) LOOP --recorremos desde 1 hasta el fin del arreglo de los productos
        INSERT INTO detalle_pedidos (pedido_id, producto_id, cantidad) -- insertamos producto por producto en detalle_pedido
        VALUES (nuevo_id, p_productos[i], p_cantidades[i]);
    END LOOP;

    -- luego, insertamos en pagos con su precio
    INSERT INTO pagos (pedido_id, medio_id, monto)
    SELECT nuevo_id, p_medio_pago_id,
           SUM(p.precio * DP.cantidad)
    FROM detalle_pedidos DP
    JOIN productos P ON P.producto_id = DP.producto_id
    WHERE DP.pedido_id = nuevo_id
    GROUP BY DP.pedido_id; --agrupamos x id del pedido

    RETURN nuevo_id;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 136 msec.
```

Total rows: | Query complete 00:00:00.136

Query | Query History

```
SELECT registrar_pedido(
                1,              -- cliente_id
                1,              -- empresa_id
                2,              -- repartidor_id
                'en camino',    -- estado del pedido
                ARRAY[1, 2],    -- productos
                ARRAY[2, 1],    -- cantidades
                2               -- medio de pago (efectivo)
);
```

Data Output | Messages | Notifications

	registrar_pedido integer
1	13

8. Cambiar el estado de un pedido con validación.

Query | Query History

```
CREATE OR REPLACE FUNCTION cambiar_estado_pedido(
    p_pedido_id INT,
    p_nuevo_estado VARCHAR)
RETURNS VOID AS $$
BEGIN
    -- verficamos que el pedido exista
    IF NOT EXISTS (
            SELECT 1 FROM pedidos
            WHERE pedido_id = p_pedido_id
    ) THEN
        RAISE EXCEPTION 'En la función cambiar_estado_pedido. Se ingresó un pedido de id % que no existe.', p_pedido_id;
    END IF;

    --verificamos q el estado sea valido para cambiarlo
    IF EXISTS (
        SELECT 1 FROM pedidos
        WHERE pedido_id = p_pedido_id
          AND estado IN ('entregado', 'cancelado') --validación, no puede tener estos estados
    ) THEN
        RAISE EXCEPTION 'En la función cambiar_estado_pedido. Se intentó cambiar un estado del pedido %, que ya fue entregado o cancelado.', p_pedido_id;
    end if;

    --validamos q el nuevo estado no este vacio
    IF p_nuevo_estado IS NULL OR TRIM(p_nuevo_estado) = '' THEN --si esta vacio o tiene espacios
        RAISE EXCEPTION 'En la función cambiar_estado_pedido. Se ingresó un estado vacío, por favor rellenarlo';
    END IF;

    UPDATE pedidos
    SET estado = p_nuevo_estado
    WHERE pedido_id = p_pedido_id;
end;
$$ LANGUAGE plpgsql;
```

Query Query History

```
SELECT cambiar_estado_pedido(13, 'entregado');-- se cambia del estado 'en camino' a 'entregado'
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	cambiar_estado_pedido void
1	

9. Descontar stock al confirmar pedido (si aplica).

Query Query History

```
CREATE OR REPLACE FUNCTION confirmar_pedido_descontar(p_pedido_id INT)
RETURNS VOID AS $$
BEGIN
    -- Verificar que el pedido exista
    IF NOT EXISTS (
        SELECT 1 FROM pedidos
        WHERE pedido_id = p_pedido_id
    ) THEN
        RAISE EXCEPTION 'No existe el pedido con ID %', p_pedido_id;
    END IF;

    -- Cambiar estado a 'entregado'
    UPDATE pedidos
    SET estado = 'entregado'
    WHERE pedido_id = p_pedido_id;

    -- Descontar stock solo en productos que tengan stock definido
    UPDATE productos
    SET stock = stock - dp.cantidad
    FROM detalle_pedidos dp
    WHERE productos.producto_id = dp.producto_id
      AND dp.pedido_id = p_pedido_id
      AND productos.stock IS NOT NULL;
END;
$$ LANGUAGE plpgsql;
```

Query Query History

```
-- Ejmplo de uso:
SELECT confirmar_pedido_descontar(1);
-- En este caso, se cambiara a entregado (el pedido 1) y se descontará el stock de los productos en el pedido con ID 1.

```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	confirmar_pedido_descontar void
1	

10. Insertar automáticamente la fecha de entrega al marcar como entregado.

Query | Query History

```sql
--función del trigger
CREATE OR REPLACE FUNCTION insertar_fecha_entrega()
RETURNS TRIGGER AS $$
BEGIN
    --primero hay q verificar q este entregado
    IF NEW.estado = 'entregado'
        THEN
        NEW.fecha_entrega = NOW()::timestamp(0);
    end if;
    RETURN NEW;
end;
$$ LANGUAGE plpgsql;

--trigger
CREATE TRIGGER trigger_insertar_fecha_entrega
BEFORE UPDATE ON pedidos
FOR EACH ROW
WHEN (OLD.estado IS DISTINCT FROM
    'entregado' AND NEW.estado = 'entregado')
EXECUTE FUNCTION insertar_fecha_entrega();
```

Data Output | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 252 msec.
```

Query | Query History

```sql
INSERT INTO pedidos (cliente_id, empresa_id, repartidor_id, fecha, estado) VALUES
(1, 2, 1, '2025-05-22 10:30:00', 'en camino');

UPDATE pedidos SET estado = 'entregado' WHERE pedido_id = 14;
```

	pedido_id [PK] integer	cliente_id integer	empresa_id integer	repartidor_id integer	fecha timestamp without time zone	fecha_entrega timestamp without time zone	estado character varying (50)
1	1	1	1	1	2025-05-20 09:00:00	[null]	entregado
2	2	2	2	2	2025-05-21 14:00:00	[null]	cancelado
3	3	3	1	3	2025-03-22 10:30:00	[null]	entregado
4	4	4	2	4	2025-03-22 12:15:00	[null]	entregado
5	5	5	1	5	2025-03-23 09:45:00	[null]	entregado
6	6	1	2	3	2025-03-24 11:00:00	[null]	entregado
7	7	2	1	3	2025-03-25 13:00:00	[null]	entregado
8	8	3	2	3	2025-03-26 15:00:00	[null]	entregado
9	9	4	1	5	2025-03-27 09:30:00	[null]	entregado
10	10	5	2	4	2025-03-28 14:00:00	[null]	entregado
11	11	1	1	2	2025-03-29 16:00:00	[null]	entregado
12	12	2	2	2	2025-03-30 17:30:00	[null]	entregado
13	13	1	1	2	2025-05-09 21:57:26	[null]	entregado
14	14	1	2	1	2025-05-22 10:30:00	2025-05-09 22:12:22	entregado

11. Registrar una notificación si se detecta un problema crítico en el pedido.

Query Query History

```sql
-- 11. Registrar una notificación si se detecta un problema crítico en el pedido.
CREATE TABLE IF NOT EXISTS notificaciones (
    notificacion_id SERIAL PRIMARY KEY,
    pedido_id INT REFERENCES pedidos(pedido_id),
    mensaje VARCHAR(50),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION registrar_notificacion_critica()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.estado = 'critico' THEN
        INSERT INTO notificaciones (pedido_id, mensaje)
        VALUES (NEW.pedido_id, 'Pedido critico.');
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER notificacion_pedido_critico
AFTER UPDATE OF estado ON pedidos
FOR EACH ROW
WHEN (NEW.estado = 'critico')
EXECUTE FUNCTION registrar_notificacion_critica();
```

Query Query History

```sql
UPDATE pedidos SET estado = 'critico' WHERE pedido_id = 14
```

Query Query History

```sql
SELECT * FROM public.notificaciones
ORDER BY notificacion_id ASC
```

Data Output Messages Notifications

	notificacion_id [PK] integer	pedido_id integer	mensaje character varying (50)	fecha timestamp without time zone
1	1	14	Pedido critico.	2025-05-09 22:15:42.653282

12. Insertar una calificación automática si no se recibe en 48 horas.

Query | Query History

```sql
CREATE OR REPLACE FUNCTION calificacion_auto() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO calificaciones ("pedido_id", "puntuacion", "comentario")
    SELECT p.pedido_id, 3, 'calificado automaticamente'
    FROM pedidos p
    LEFT JOIN calificaciones c ON p.pedido_id = c.pedido_id
    WHERE c.pedido_id IS NULL
    AND p.fecha + INTERVAL '48 hours' <= NOW();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER activar_calificacion_auto
    AFTER INSERT ON pedidos
    FOR EACH STATEMENT
    EXECUTE FUNCTION calificacion_auto();

INSERT INTO pedidos (cliente_id, empresa_id, repartidor_id, fecha, estado) VALUES
(3, 2, 3, '2025-05-01 12:30:00', 'en camino');
```

Query | Query History

```sql
SELECT * FROM public.calificaciones
ORDER BY calificacion_id ASC
```

Data Output | Messages | Notifications

	calificacion_id [PK] integer	pedido_id integer	puntuacion integer	comentario text
1	1	1	5	Muy buen servicio y rápido.
2	2	3	5	Muy amable y rápido.
3	3	4	4	Buen servicio.
4	4	5	3	Tardó un poco, pero correcto.
5	5	6	5	Excelente. Muy recomenda...
6	6	7	4	Bien.
7	7	8	5	Rápido y eficiente.
8	8	9	2	Se equivocó en la entrega.
9	9	10	5	Perfecto.
10	10	11	3	calificado automaticamente
11	11	17	3	calificado automaticamente
12	12	12	3	calificado automaticamente

13. Resumen de pedidos por cliente (monto total, número de pedidos).

Query | Query History

```
CREATE OR REPLACE VIEW resumen_pedidos_x_cliente AS
    SELECT p.cliente_id id, c.nombre AS nombre_cliente, COUNT(*) AS num_pedidos, SUM(pa.monto) AS monto_total
    FROM pedidos p
    INNER JOIN pagos pa ON p.pedido_id = pa.pedido_id
    INNER JOIN clientes c ON c.cliente_id = p.cliente_id
    WHERE p.estado = 'entregado'
    GROUP BY p.cliente_id, c.nombre;
```

Data Output | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 194 msec.
```

Query | Query History

```
SELECT * FROM public.resumen_pedidos_x_cliente
```

Data Output | Messages | Notifications

	id integer	nombre_cliente character varying (100)	num_pedidos bigint	monto_total numeric
1	1	Ana Torres	3	26400.00
2	2	Luis Pérez	1	2500.00
3	3	María López	2	10900.00
4	4	Pedro Ramírez	2	24000.00
5	5	Lucía Díaz	2	11700.00

14. Vista de desempeño por repartidor.

Query | Query History

```
CREATE OR REPLACE VIEW repartidores_desempenios AS
SELECT r.nombre as repartidor, COUNT(p.pedido_id) as pedidos_entregados, AVG(c.puntuacion) as calificación_avg
FROM (repartidores as r
    INNER JOIN pedidos as p ON r.repartidor_id = p.repartidor_id)
        FULL JOIN calificaciones as c ON p.pedido_id = c.pedido_id
WHERE p.estado = 'entregado'
GROUP BY r.repartidor_id;
```

Data Output | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 157 msec.
```

Query | Query History

```
1  SELECT * FROM public.repartidores_desempenios
2
```

Data Output | Messages | Notifications

	repartidor character varying (100)	pedidos_entregados bigint	calificación_avg numeric
1	Carlos Gómez	2	5.0000000000000000
2	Sofía Rojas	3	3.0000000000000000
3	Miguel Torres	4	4.7500000000000000
4	Elena Mendoza	2	4.5000000000000000
5	Raúl Vargas	2	2.5000000000000000

15. Vista de empresas asociadas con mayor volumen de paquetes entregados.

Query | Query History

```
1  CREATE OR REPLACE VIEW empresas_mas_pedidos_entregados AS
2  SELECT e.empresa_id, e.nombre AS empresa_nombre, COUNT(p.pedido_id) AS total_pedidos_entregados
3  FROM empresas e
4  LEFT JOIN pedidos p ON e.empresa_id = p.empresa_id
5  WHERE p.estado = 'entregado'
6  GROUP BY e.empresa_id, e.nombre
7  ORDER BY total_pedidos_entregados DESC;
```

Data Output | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 133 msec.
```

Query | Query History

```
1  SELECT * FROM public.empresas_mas_pedidos_entregados
2
```

Data Output | Messages | Notifications

	empresa_id integer	empresa_nombre character varying (100)	total_pedidos_entregados bigint
1	1	Farmacia Salud	7
2	2	Express Documentos	5