

TALLER DE REPASO DE BASES DE DATOS

Brayan Alejandro Gutiérrez López

Carol Dayana Imbachi Inchima

Facultad de Ingeniería, Institución Universitaria Colegio Mayor del Cauca

Tópicos Avanzados en Bases de Datos

Ginna Fernanda Puliche Corrales



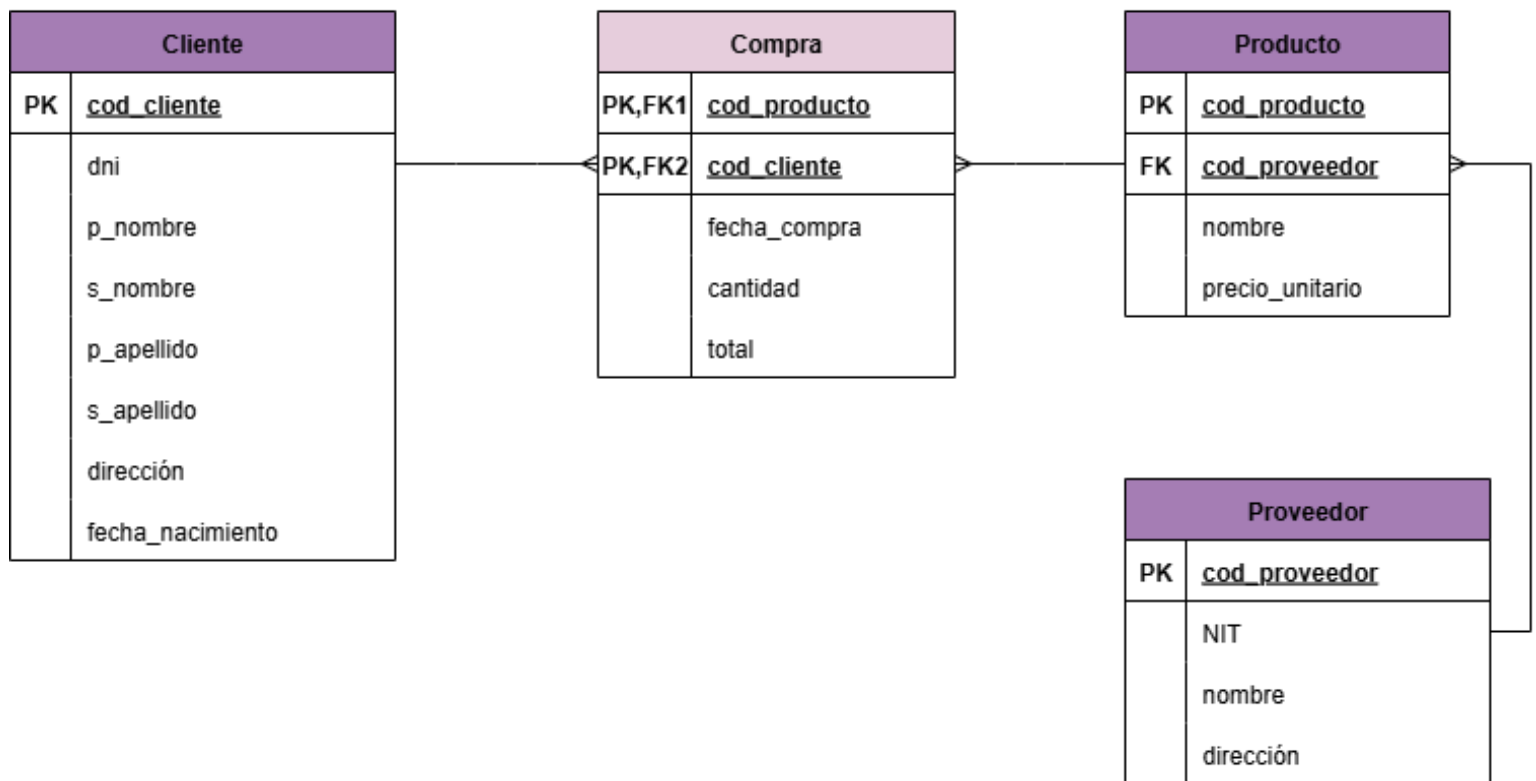
18 de agosto de 2025

Realizar los siguientes modelos E-R:

1. Una empresa vende productos a varios clientes. Se requiere conocer la información personal de los clientes (nombres, apellidos, identificación, dirección, fecha de nacimiento). Cada producto tiene nombre, código, precio unitario. Un cliente puede comprar varios productos a la empresa y un mismo producto puede ser comprado por varios clientes.

Los productos son suministrados por diferentes proveedores. Se debe tener en cuenta que un producto solo puede ser suministrado por un proveedor y que un proveedor puede suministrar diferentes productos. De cada proveedor se desea conocer el NIT, nombre y dirección.

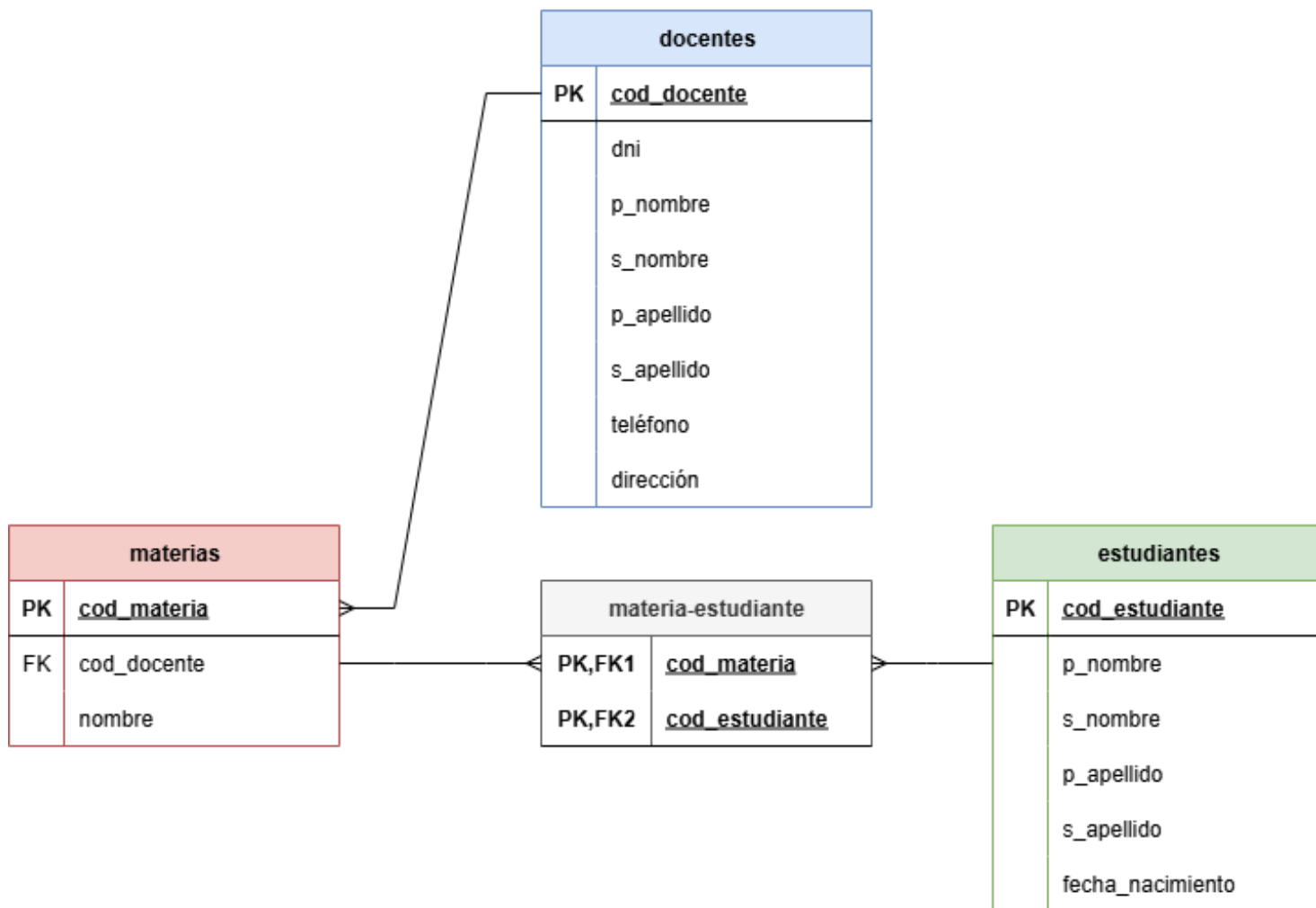
1. EMPRESA



2. Se desea diseñar una base de datos de un instituto. En la base de datos se requiere guardar la información de los docentes del Instituto como identificación , nombres, dirección, teléfono. Para tener en cuenta:

- a. Los docentes dictan materias y cada materia tiene un código y nombre
- b. Cada estudiante se encuentra matriculado en una o varias materias
- c. Se necesita guardar de cada estudiante código universitario, nombre, apellidos y fecha de nacimiento.
- d. Los docentes pueden dictar varias materias, pero una materia puede ser dictada por un docente

2. INSTITUTO

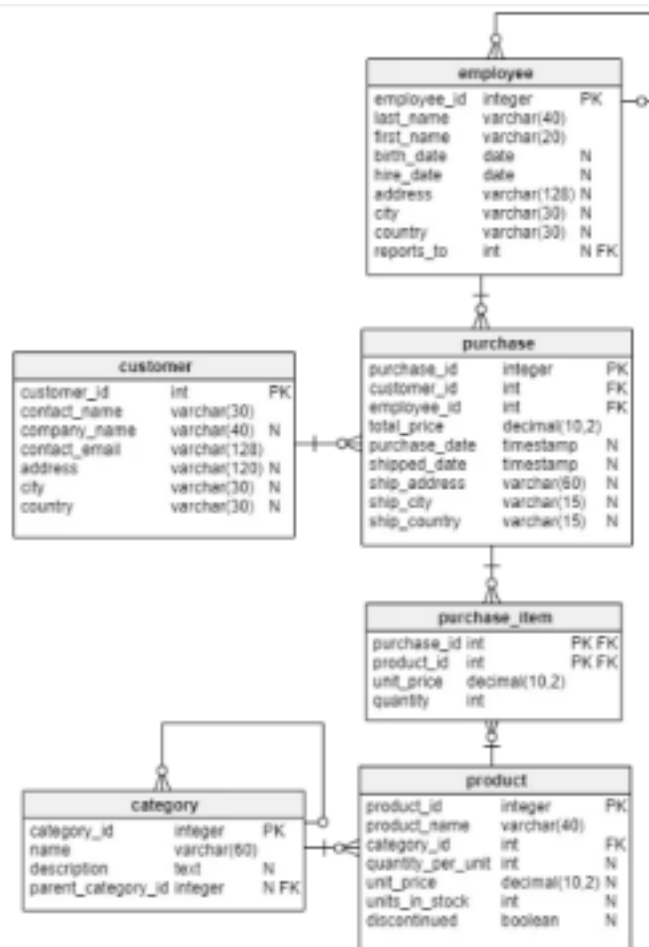


A continuación, debe elaborar las consultas sobre el diagrama de la Tienda, para este ejercicio

deberán realizar lo siguiente:

- Crear la base de datos
- Crear tablas
- Realizar inserción de datos
- Crear las consultas de acuerdo a lo solicitado.

Para tener en cuenta hay una breve descripción de las tablas que los guiaran para saber con qué características deberán tener los registros.

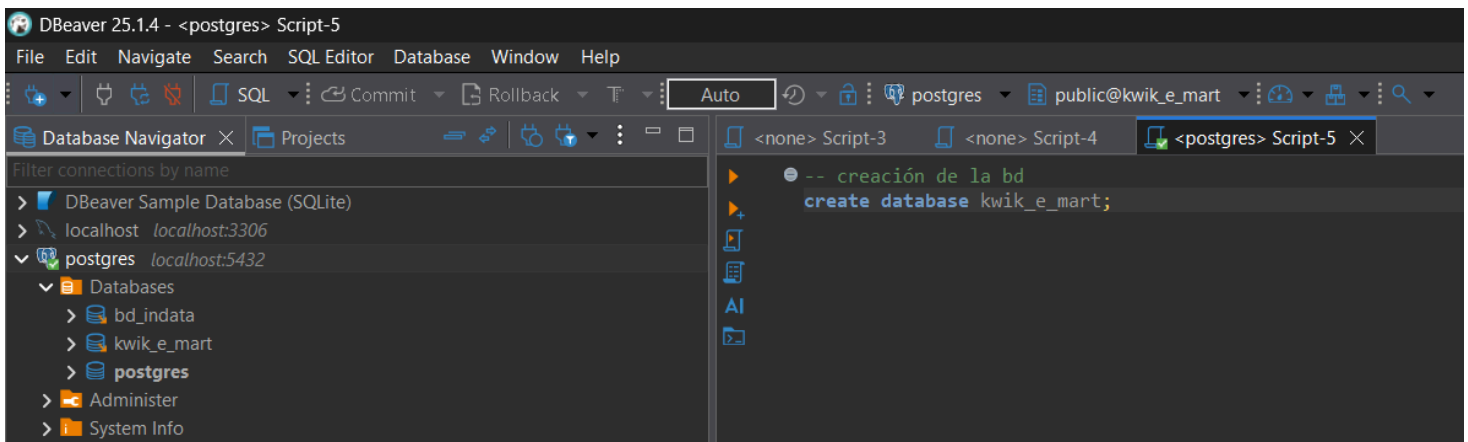


Descripción de las tablas:

- **employee:** Contiene detalles del empleado como ID, nombre, fecha de nacimiento, dirección, ciudad, país y supervisor inmediato.
- **customer:** Almacena información sobre los clientes, como su ID, nombre, empresa, correo electrónico, dirección, ciudad y país.
- **purchase:** Registra detalles del pedido, incluyendo ID del pedido, ID del cliente (quién realizó el pedido), empleado (quién atendió el pedido), precio total, y detalles de compra y envío.

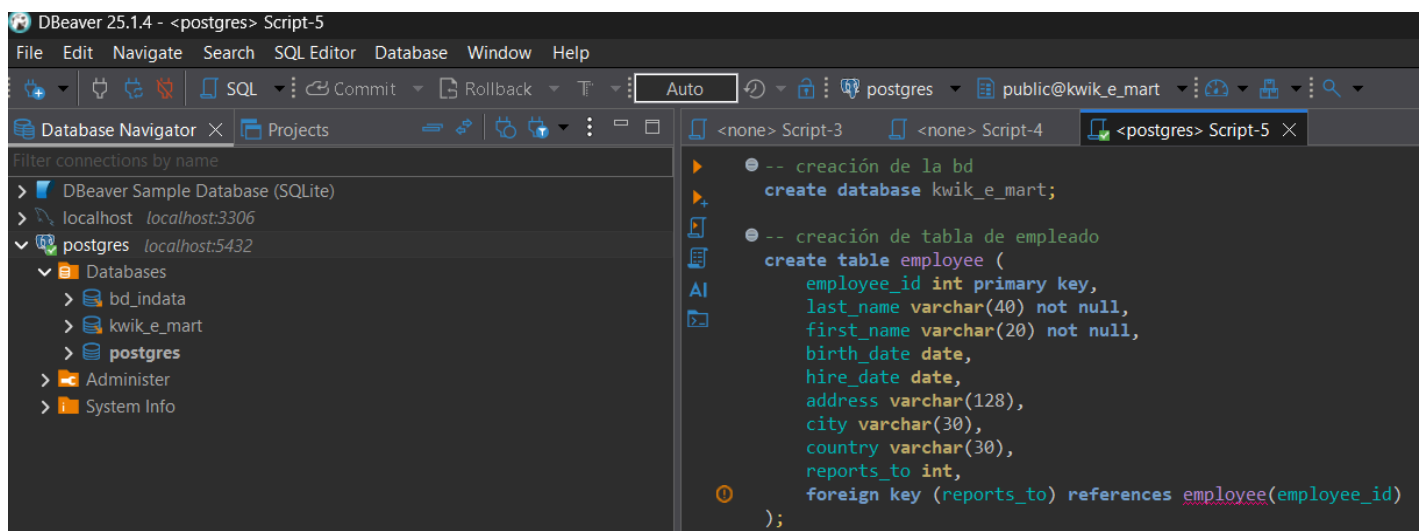
- **purchase_item:** Conecta las compras con los productos mediante ID, producto, precio unitario y cantidad.
- **category:** Proporciona información sobre las categorías de productos mediante el ID de categoría, el nombre, la descripción y el ID de categoría principal.
- **Product:** Enumera los productos de la tienda e incluye el ID del producto, el nombre del producto, el ID de la categoría, la cantidad por unidad, el precio unitario, las unidades en stock y el estado del producto.

I. Creación de la base de datos:

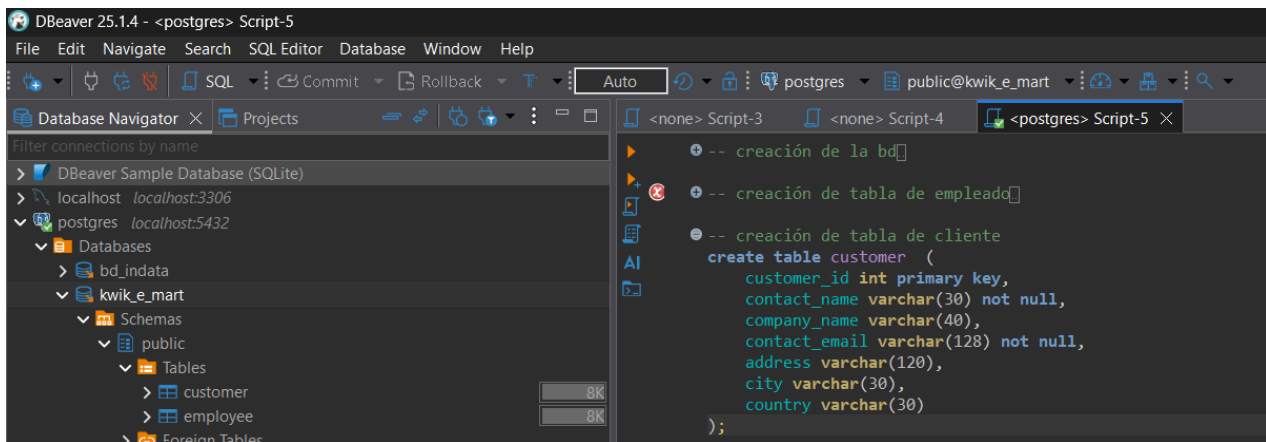


II. Creación de las tablas:

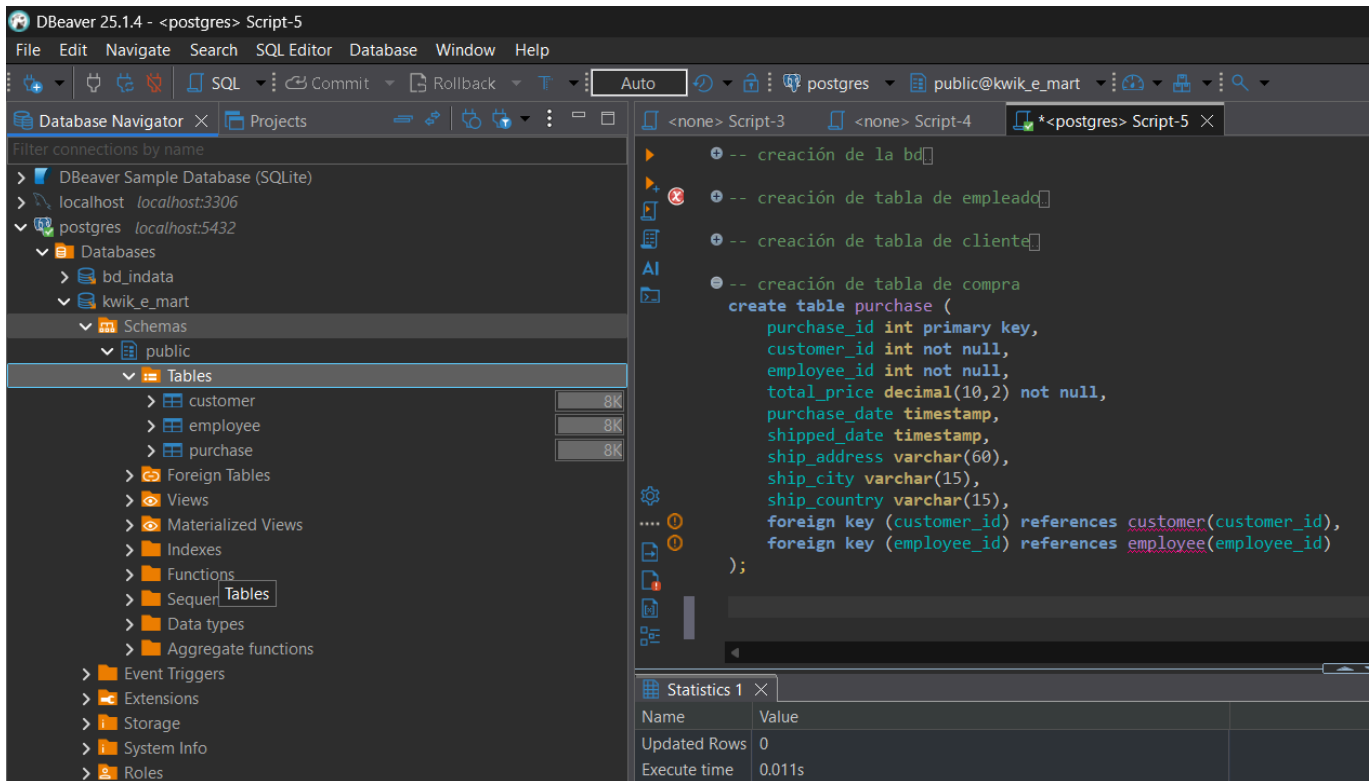
A. Tabla de empleado



B. Tabla de cliente



C. Tabla de compra



The screenshot displays the DBeaver 25.1.4 application window. The top menu bar includes File, Edit, Navigate, Search, SQL Editor, Database, Window, and Help. Below the menu is a toolbar with various icons for database operations. The main interface is divided into three panes:

- Database Navigator (Left Pane):** Shows a tree view of the database structure. The 'public' schema is selected, showing tables like 'category', 'customer', 'employee', and 'purchase'. The 'category' table is highlighted.
- SQL Editor (Right Pane):** Contains a SQL script for creating a table named 'category'. The script includes comments in Spanish and a foreign key constraint.
- Statistics (Bottom Pane):** Displays execution statistics for the SQL script, including 'Updated Rows', 'Execute time', and 'Start time'.

The SQL script in the editor is as follows:

```
-- creación de la bd
-- creación de tabla de empleado
-- creación de tabla de cliente
-- creación de tabla de compra
-- creacion de tabla de categoría
create table category (
    category_id int primary key,
    name varchar(60) not null,
    description text,
    parent_category_id int,
    foreign key (parent_category_id) references category(category_id)
);
```

The Statistics pane shows the following data:

Name	Value
Updated Rows	0
Execute time	0.013s
Start time	Sun Aug 17 19:10:52 COT 2025

The screenshot displays the DBeaver SQL Editor interface. The left sidebar shows the database structure: postgres > local host:5432 > Databases > bd_indata > kwik_e_mart > Schemas > public > Tables. The main editor shows the SQL script for creating the 'product' table. The bottom status bar shows the execution results: Updated Rows: 0, Execute time: 0.007s, Start time: Sun Aug 17 19:12:10 COT 2025, Finish time: Sun Aug 17 19:12:10 COT 2025.

```
-- creación de tabla de empleado
-- creación de tabla de cliente
-- creación de tabla de compra
-- creacion de tabla de categoría
-- creación de tabla de producto
create table product (
    product_id int primary key,
    product_name varchar(40) not null,
    category_id int not null,
    quantity_per_unit int,
    unit_price decimal(10,2),
    units_in_stock int,
    discontinued boolean,
    foreign key (category_id) references category(category_id)
);
```

Name	Value
Updated Rows	0
Execute time	0.007s
Start time	Sun Aug 17 19:12:10 COT 2025
Finish time	Sun Aug 17 19:12:10 COT 2025

F. Tabla de conexión entre compra y producto

The screenshot shows the DBeaver 25.1.4 interface with a PostgreSQL database named 'kwik_e_mart'. The Database Navigator on the left shows the 'public' schema with a table 'purchase_item' (8K). The SQL Editor in the center shows the following SQL script:

```
-- creación de la bd
-- creación de tabla de empleado
-- creación de tabla de cliente
-- creación de tabla de compra
-- creación de tabla de categoría
-- creación de tabla de producto
-- creación de tabla intermedia entre compra y producto
create table purchase_item (
  purchase_id int,
  product_id int,
  primary key (purchase_id, product_id),
  unit_price decimal(10,2),
  quantity int not null,
  foreign key (purchase_id) references purchase(purchase_id),
  foreign key (product_id) references product(product_id)
);
```

The Statistics window at the bottom right shows the following data:

Name	Value
Updated Rows	0
Execute time	0.008s
Start time	Sun Aug 17 19:13:39 COT 2025
Finish time	Sun Aug 17 19:13:39 COT 2025
Query	-- creación de tabla intermedia entre compra y producto

III. Inserción de datos:

A. Tabla de categorías

```
-- datos para la tabla de categoria
INSERT INTO category (category_id, name, description, parent_category_id) VALUES
(1, 'Cleaning', 'Household cleaning products', NULL),
(2, 'Beverages', 'Juices, sodas and water', NULL),
(3, 'Dairy', 'Milk, cheese, yogurt', NULL),
(4, 'Meat', 'Fresh and frozen meat products', NULL),
(5, 'Fruits and Vegetables', 'Fresh natural products', NULL),
(6, 'Snacks', 'Cookies, chips and similar', NULL),
(7, 'Technology', 'Electronic devices and accessories', NULL),
(8, 'Computer Accessories', 'PC peripherals and accessories', 7);
```

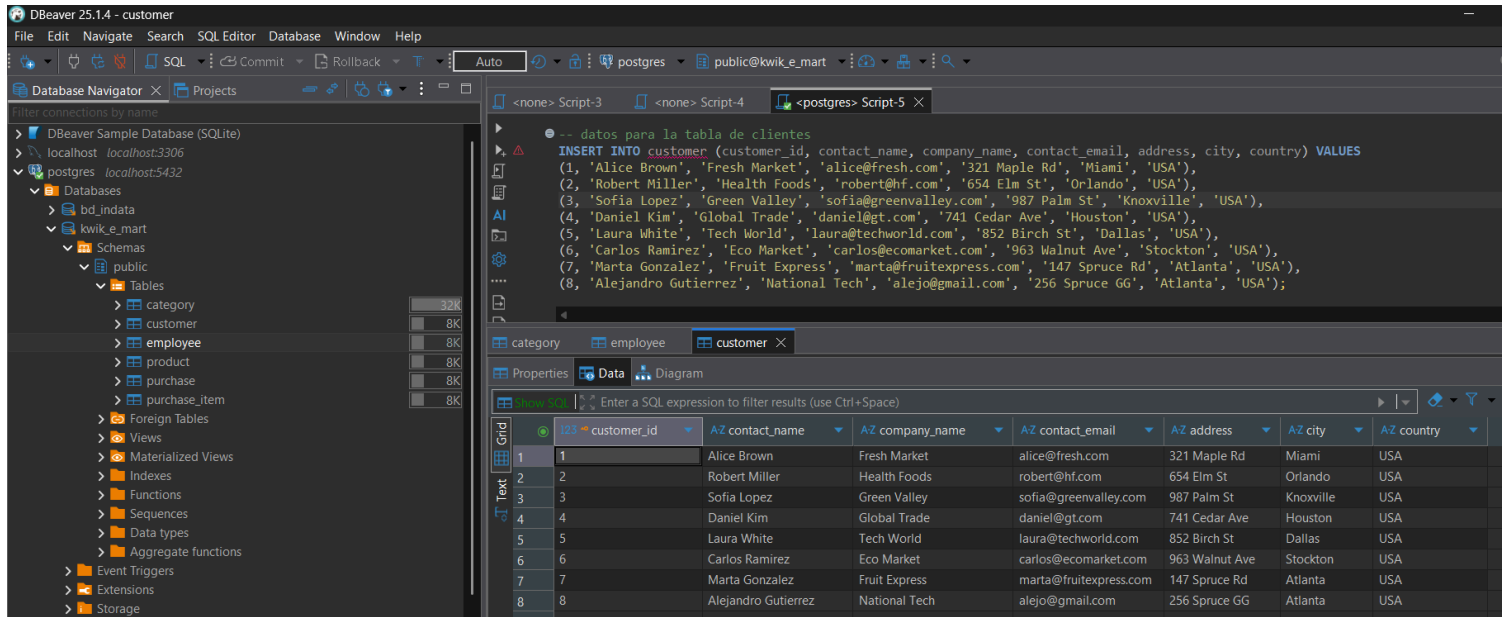
category_id	name	description	parent_category_id
1	Cleaning	Household cleaning products	[NULL]
2	Beverages	Juices, sodas and water	[NULL]
3	Dairy	Milk, cheese, yogurt	[NULL]
4	Meat	Fresh and frozen meat products	[NULL]
5	Fruits and Vegetables	Fresh natural products	[NULL]
6	Snacks	Cookies, chips and similar	[NULL]
7	Technology	Electronic devices and accessories	[NULL]
8	Computer Accessories	PC peripherals and accessories	7

B. Tabla de empleados

```
INSERT INTO employee (employee_id, last_name, first_name, birth_date, hire_date, address, city, country, reports_to) VALUES
(1, 'Smith', 'John', '1985-06-12', '2010-03-01', '123 Main St', 'New York', 'USA', NULL),
(2, 'Johnson', 'Emily', '1990-09-20', '2015-07-15', '456 Oak St', 'Los Angeles', 'USA', 1),
(3, 'Garcia', 'Luis', '1982-02-10', '2008-11-03', '789 Pine Ave', 'Chicago', 'USA', 1);
```

employee_id	last_name	first_name	birth_date	hire_date	address	city	country
1	Smith	John	1985-06-12	2010-03-01	123 Main St	New York	USA
2	Johnson	Emily	1990-09-20	2015-07-15	456 Oak St	Los Angeles	USA
3	Garcia	Luis	1982-02-10	2008-11-03	789 Pine Ave	Chicago	USA

C. Tabla de clientes

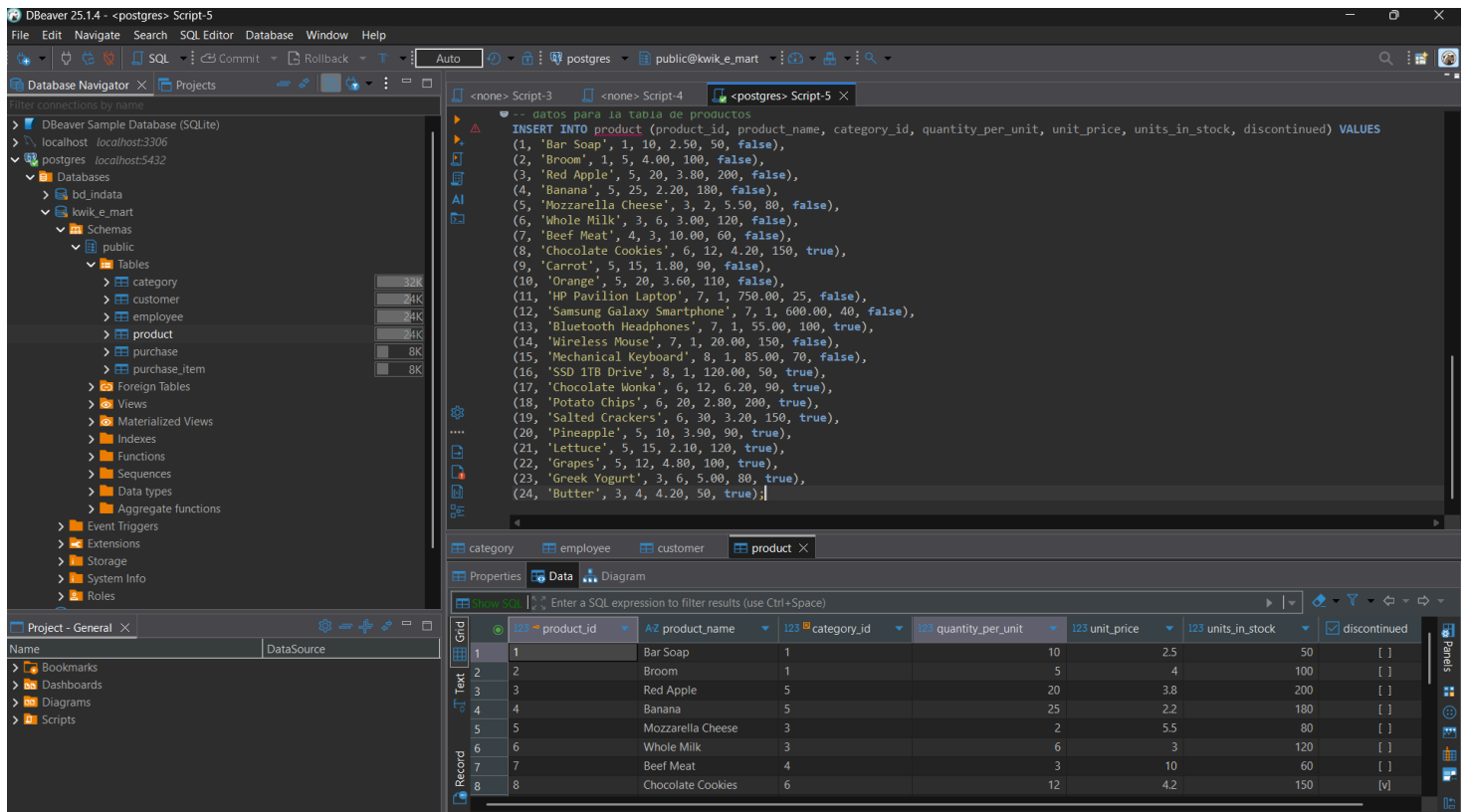


The screenshot shows the DBeaver 25.1.4 interface. On the left, the 'Database Navigator' pane shows the 'public' schema of a PostgreSQL database. The 'customer' table is selected. The main pane displays the table's data in a grid view. The SQL editor at the top shows the following INSERT statement:

```
INSERT INTO customer (customer_id, contact_name, company_name, contact_email, address, city, country) VALUES
(1, 'Alice Brown', 'Fresh Market', 'alice@fresh.com', '321 Maple Rd', 'Miami', 'USA'),
(2, 'Robert Miller', 'Health Foods', 'robert@hf.com', '654 Elm St', 'Orlando', 'USA'),
(3, 'Sofia Lopez', 'Green Valley', 'sofia@greenvalley.com', '987 Palm St', 'Knoxville', 'USA'),
(4, 'Daniel Kim', 'Global Trade', 'daniel@gt.com', '741 Cedar Ave', 'Houston', 'USA'),
(5, 'Laura White', 'Tech World', 'laura@techworld.com', '852 Birch St', 'Dallas', 'USA'),
(6, 'Carlos Ramirez', 'Eco Market', 'carlos@ecomarket.com', '963 Walnut Ave', 'Stockton', 'USA'),
(7, 'Marta Gonzalez', 'Fruit Express', 'marta@fruitexpress.com', '147 Spruce Rd', 'Atlanta', 'USA'),
(8, 'Alejandro Gutierrez', 'National Tech', 'alejo@gmail.com', '256 Spruce GG', 'Atlanta', 'USA');
```

customer_id	contact_name	company_name	contact_email	address	city	country
1	Alice Brown	Fresh Market	alice@fresh.com	321 Maple Rd	Miami	USA
2	Robert Miller	Health Foods	robert@hf.com	654 Elm St	Orlando	USA
3	Sofia Lopez	Green Valley	sofia@greenvalley.com	987 Palm St	Knoxville	USA
4	Daniel Kim	Global Trade	daniel@gt.com	741 Cedar Ave	Houston	USA
5	Laura White	Tech World	laura@techworld.com	852 Birch St	Dallas	USA
6	Carlos Ramirez	Eco Market	carlos@ecomarket.com	963 Walnut Ave	Stockton	USA
7	Marta Gonzalez	Fruit Express	marta@fruitexpress.com	147 Spruce Rd	Atlanta	USA
8	Alejandro Gutierrez	National Tech	alejo@gmail.com	256 Spruce GG	Atlanta	USA

D. Tabla de productos

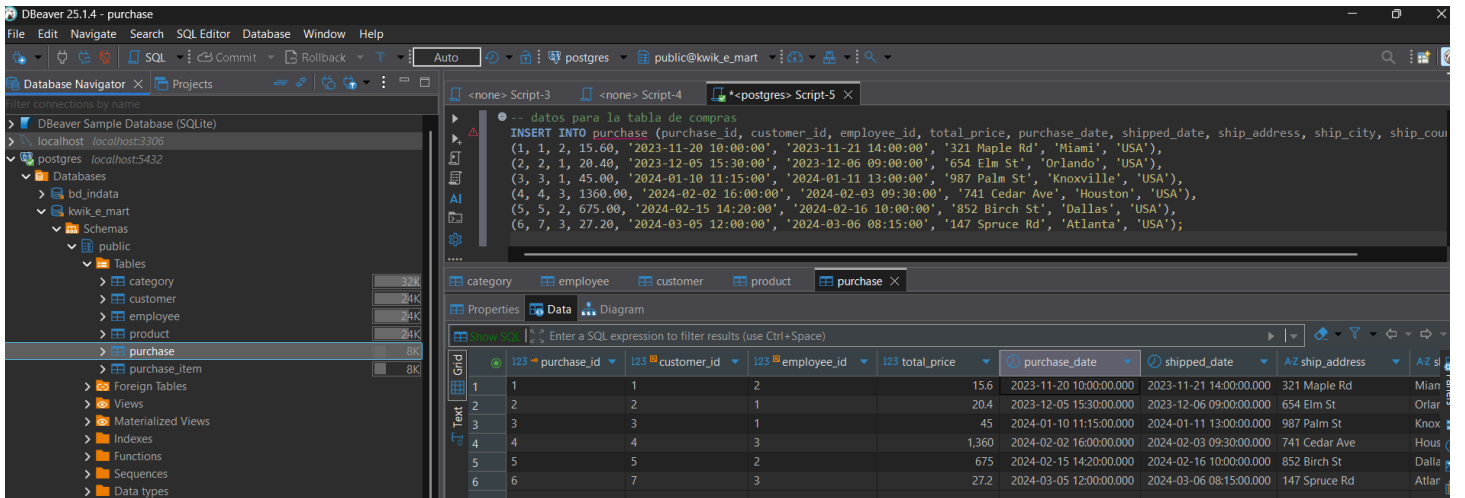


The screenshot shows the DBeaver 25.1.4 interface. On the left, the 'Database Navigator' pane shows the 'public' schema of a PostgreSQL database. The 'product' table is selected. The main pane displays the table's data in a grid view. The SQL editor at the top shows the following INSERT statement:

```
INSERT INTO product (product_id, product_name, category_id, quantity_per_unit, unit_price, units_in_stock, discontinued) VALUES
(1, 'Bar Soap', 1, 10, 2.50, 50, false),
(2, 'Broom', 1, 5, 4.00, 100, false),
(3, 'Red Apple', 5, 20, 3.80, 200, false),
(4, 'Banana', 5, 25, 2.20, 180, false),
(5, 'Mozzarella Cheese', 3, 2, 5.50, 80, false),
(6, 'Whole Milk', 3, 6, 3.00, 120, false),
(7, 'Beef Meat', 4, 3, 10.00, 60, false),
(8, 'Chocolate Cookies', 6, 12, 4.20, 150, true),
(9, 'Carrot', 5, 15, 1.80, 90, false),
(10, 'Orange', 5, 20, 3.60, 110, false),
(11, 'HP Pavilion Laptop', 7, 1, 750.00, 25, false),
(12, 'Samsung Galaxy Smartphone', 7, 1, 600.00, 40, false),
(13, 'Bluetooth Headphones', 7, 1, 55.00, 100, true),
(14, 'Wireless Mouse', 7, 1, 20.00, 150, false),
(15, 'Mechanical Keyboard', 8, 1, 85.00, 70, false),
(16, 'SSD 1TB Drive', 8, 1, 120.00, 50, true),
(17, 'Chocolate Wonka', 6, 12, 6.20, 90, true),
(18, 'Potato Chips', 6, 20, 2.80, 200, true),
(19, 'Salted Crackers', 6, 30, 3.20, 150, true),
(20, 'Pineapple', 5, 10, 3.90, 90, true),
(21, 'Lettuce', 5, 15, 2.10, 120, true),
(22, 'Grapes', 5, 12, 4.80, 100, true),
(23, 'Greek Yogurt', 3, 6, 5.00, 80, true),
(24, 'Butter', 3, 4, 4.20, 50, true);
```

product_id	product_name	category_id	quantity_per_unit	unit_price	units_in_stock	discontinued
1	Bar Soap	1	10	2.5	50	[]
2	Broom	1	5	4	100	[]
3	Red Apple	5	20	3.8	200	[]
4	Banana	5	25	2.2	180	[]
5	Mozzarella Cheese	3	2	5.5	80	[]
6	Whole Milk	3	6	3	120	[]
7	Beef Meat	4	3	10	60	[]
8	Chocolate Cookies	6	12	4.2	150	[v]

E. Tabla de compras



Database Navigator: PostgreSQL (localhost:5432) - public - Tables

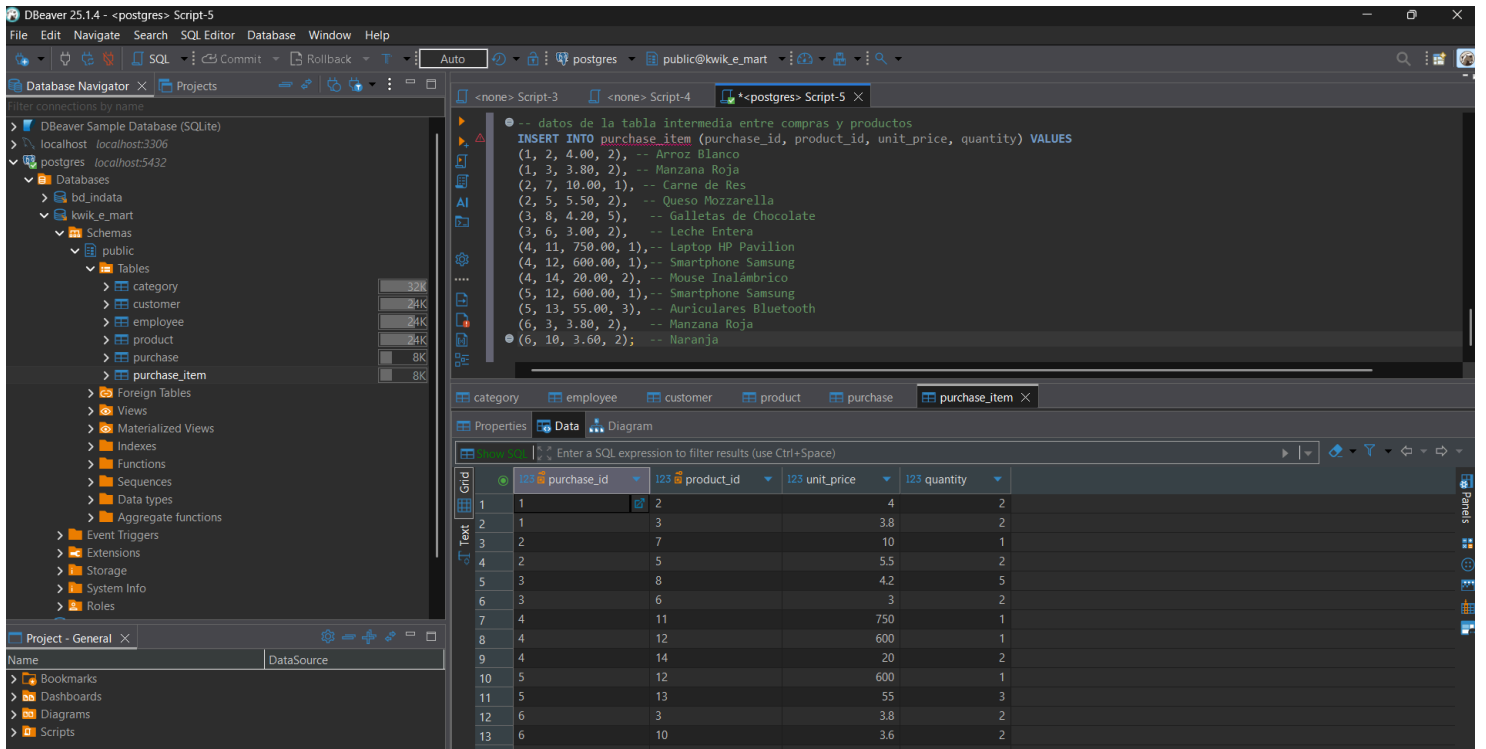
- category (32K)
- customer (24K)
- employee (24K)
- product (24K)
- purchase (8K)
- purchase_item (8K)

SQL Editor: Insert into purchase table

```
-- datos para la tabla de compras
INSERT INTO purchase (purchase_id, customer_id, employee_id, total_price, purchase_date, shipped_date, ship_address, ship_city, ship_country) VALUES
(1, 1, 2, 15.60, '2023-11-20 10:00:00', '2023-11-21 14:00:00', '321 Maple Rd', 'Miami', 'USA'),
(2, 2, 1, 20.40, '2023-12-05 15:30:00', '2023-12-06 09:00:00', '654 Elm St', 'Orlando', 'USA'),
(3, 3, 1, 45.00, '2024-01-10 11:15:00', '2024-01-11 13:00:00', '987 Palm St', 'Knoxville', 'USA'),
(4, 4, 3, 1360.00, '2024-02-02 16:00:00', '2024-02-03 09:30:00', '741 Cedar Ave', 'Houston', 'USA'),
(5, 5, 2, 675.00, '2024-02-15 14:20:00', '2024-02-16 10:00:00', '852 Birch St', 'Dallas', 'USA'),
(6, 7, 3, 27.20, '2024-03-05 12:00:00', '2024-03-06 08:15:00', '147 Spruce Rd', 'Atlanta', 'USA');
```

purchase_id	customer_id	employee_id	total_price	purchase_date	shipped_date	ship_address	ship_city	ship_country
1	1	2	15.6	2023-11-20 10:00:00	2023-11-21 14:00:00	321 Maple Rd	Miami	USA
2	2	1	20.4	2023-12-05 15:30:00	2023-12-06 09:00:00	654 Elm St	Orlando	USA
3	3	1	45	2024-01-10 11:15:00	2024-01-11 13:00:00	987 Palm St	Knoxville	USA
4	4	3	1360	2024-02-02 16:00:00	2024-02-03 09:30:00	741 Cedar Ave	Houston	USA
5	5	2	675	2024-02-15 14:20:00	2024-02-16 10:00:00	852 Birch St	Dallas	USA
6	7	3	27.2	2024-03-05 12:00:00	2024-03-06 08:15:00	147 Spruce Rd	Atlanta	USA

F. Tabla de intermedia entre compras y productos



Database Navigator: PostgreSQL (localhost:5432) - public - Tables

- category (32K)
- customer (24K)
- employee (24K)
- product (24K)
- purchase (8K)
- purchase_item (8K)

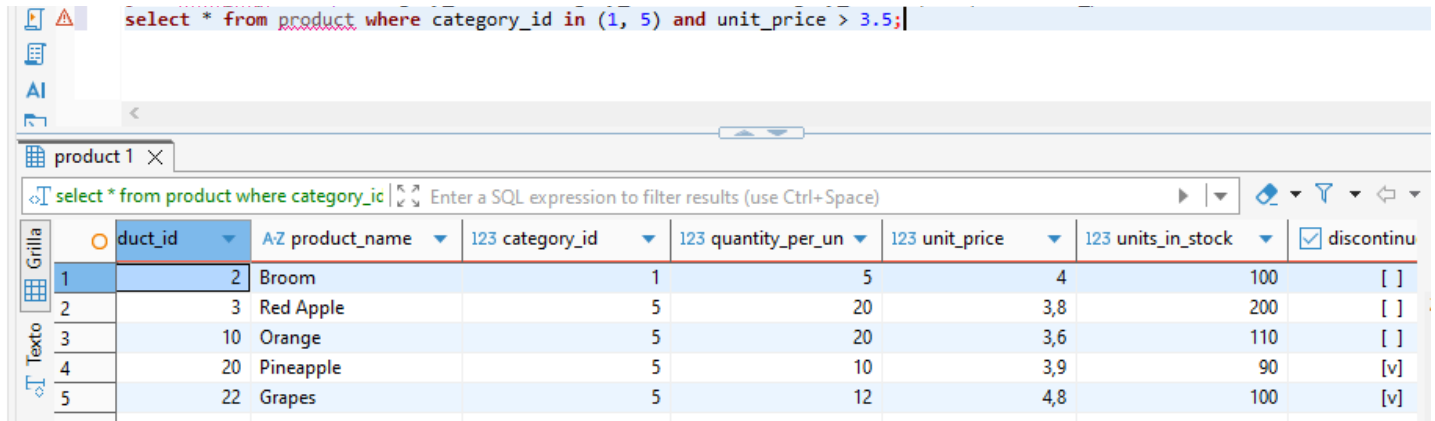
SQL Editor: Insert into purchase_item table

```
-- datos de la tabla intermedia entre compras y productos
INSERT INTO purchase_item (purchase_id, product_id, unit_price, quantity) VALUES
(1, 2, 4.00, 2), -- Arroz Blanco
(1, 3, 3.80, 2), -- Manzana Roja
(2, 7, 10.00, 1), -- Carne de Res
(2, 5, 5.50, 2), -- Queso Mozzarella
(3, 8, 4.20, 5), -- Galletas de Chocolate
(3, 6, 3.00, 2), -- Leche Entera
(4, 11, 750.00, 1), -- Laptop HP Pavilion
(4, 12, 600.00, 1), -- Smartphone Samsung
(4, 14, 20.00, 2), -- Mouse Inalámbrico
(5, 12, 600.00, 1), -- Smartphone Samsung
(5, 13, 55.00, 3), -- Auriculares Bluetooth
(6, 3, 3.80, 2), -- Manzana Roja
(6, 10, 3.60, 2), -- Naranja
```

purchase_id	product_id	unit_price	quantity
1	2	4	2
2	3	3.8	2
3	7	10	1
4	5	5.5	2
5	8	4.2	5
6	6	3	2
7	11	750	1
8	12	600	1
9	14	20	2
10	12	600	1
11	13	55	3
12	3	3.8	2
13	10	3.6	2

Realizar las siguientes consultas:

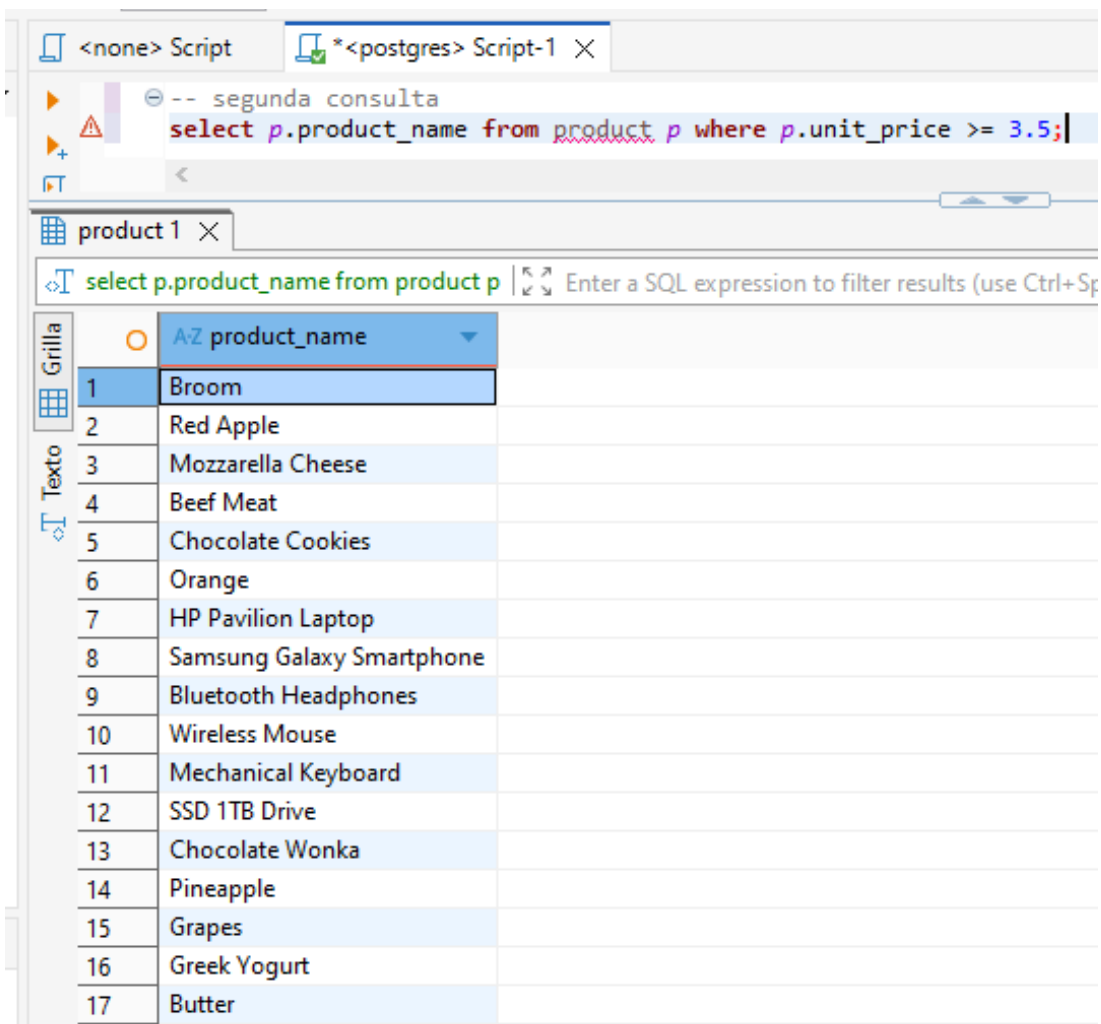
1. Mostrar los datos de todos los productos de las categorías con un ID de 1 (Alimentos) o 5 (Frutas y Verduras) y con un precio unitario superior a 3,5.



The screenshot shows a database client interface. At the top, a SQL query is entered: `select * from product where category_id in (1, 5) and unit_price > 3.5;`. Below the query editor, a table titled "product 1" displays the results. The table has columns: product_id, product_name, category_id, quantity_per_unit, unit_price, units_in_stock, and discontinued. The results show 5 rows of data.

product_id	product_name	category_id	quantity_per_unit	unit_price	units_in_stock	discontinued
2	Broom	1	5	4	100	[]
3	Red Apple	5	20	3,8	200	[]
10	Orange	5	20	3,6	110	[]
20	Pineapple	5	10	3,9	90	[v]
22	Grapes	5	12	4,8	100	[v]

2. Mostrar los nombres de los productos con un precio unitario mayor o igual a 3.5.



The screenshot shows a database client interface. At the top, a SQL query is entered: `-- segunda consulta
select p.product_name from product p where p.unit_price >= 3.5;`. Below the query editor, a table titled "product 1" displays the results. The table has columns: product_id, product_name. The results show 17 rows of data.

product_id	product_name
1	Broom
2	Red Apple
3	Mozzarella Cheese
4	Beef Meat
5	Chocolate Cookies
6	Orange
7	HP Pavilion Laptop
8	Samsung Galaxy Smartphone
9	Bluetooth Headphones
10	Wireless Mouse
11	Mechanical Keyboard
12	SSD 1TB Drive
13	Chocolate Wonka
14	Pineapple
15	Grapes
16	Greek Yogurt
17	Butter

3. Seleccione nombres de productos junto con sus categorías. Muestre dos columnas: product_name y category_name.

<none> Script *<postgres> Script-1 X

```
-- tercera consulta
select p.product_name, c.name as category_name from product p join category c on p.category_id = c.category_id;
```

product(+) 1 X

select p.product_name, c.name as cate | Enter a SQL expression to filter results (use Ctrl+Space)

	AZ product_name	AZ category_name
1	Bar Soap	Cleaning
2	Broom	Cleaning
3	Red Apple	Fruits and Vegetables
4	Banana	Fruits and Vegetables
5	Mozzarella Cheese	Dairy
6	Whole Milk	Dairy
7	Beef Meat	Meat
8	Chocolate Cookies	Snacks
9	Carrot	Fruits and Vegetables
10	Orange	Fruits and Vegetables
11	HP Pavilion Laptop	Technology
12	Samsung Galaxy Smartphone	Technology
13	Bluetooth Headphones	Technology
14	Wireless Mouse	Technology
15	Mechanical Keyboard	Computer Accessories
16	SSD 1TB Drive	Computer Accessories
17	Chocolate Wonka	Snacks
18	Potato Chips	Snacks
19	Salted Crackers	Snacks
20	Pineapple	Fruits and Vegetables
21	Lettuce	Fruits and Vegetables
22	Grapes	Fruits and Vegetables
23	Greek Yogurt	Dairy
24	Butter	Dairy

4. Para cada compra, muestre el ID de compra, el nombre del producto, el precio unitario en el momento de la compra y la cantidad de artículos de cada producto.

<none> Script *<postgres> Script-1 X

```
-- cuarta consulta
select pi.purchase_id, p.product_name, pi.unit_price, pi.quantity from purchase_item pi join product p on p.product_id = pi.product_id order by pi.purchase_id asc;
```

purchase_item(+) 1 X

select pi.purchase_id, p.product_name, | Enter a SQL expression to filter results (use Ctrl+Space)

	123 purchase_id	AZ product_name	123 unit_price	123 quantity
1	1	Broom	4	2
2	1	Red Apple	3,8	2
3	2	Beef Meat	10	1
4	2	Mozzarella Cheese	5,5	2
5	3	Chocolate Cookies	4,2	5
6	3	Whole Milk	3	2
7	4	HP Pavilion Laptop	750	1
8	4	Samsung Galaxy Smartphone	600	1
9	4	Wireless Mouse	20	2
10	5	Samsung Galaxy Smartphone	600	1
11	5	Bluetooth Headphones	55	3
12	6	Red Apple	3,8	2
13	6	Orange	3,6	2

5. Para cada compra, muestre todas las categorías de productos comprados en esta compra. Muestre cada categoría sólo una vez por cada compra.

Script Editor: *<postgres> Script-1

```
-- quinta consulta
select pi.purchase_id, c.name as category_name from purchase_item pi
join product p on pi.product_id = p.product_id
join category c on p.category_id = c.category_id group by category_name, pi.purchase_id order by pi.purchase_id;
```

purchase_item(+) 1

select pi.purchase_id, c.name as category_name

Grilla

	123 purchase_id	A-Z category_name
1	1	Cleaning
2	1	Fruits and Vegetables
3	2	Dairy
4	2	Meat
5	3	Dairy
6	3	Snacks
7	4	Technology
8	5	Technology
9	6	Fruits and Vegetables

6. Mostrar los apellidos, nombres y fechas de nacimiento de los empleados. Ordene los resultados por la edad del empleado en orden ascendente

Script Editor: *<postgres> Script-1

```
-- sexta consulta
select e.last_name, e.first_name, e.birth_date from employee e order by e.birth_date asc;
```

employee 1

select e.last_name, e.first_name, e.birth_date

Grilla

	A-Z last_name	A-Z first_name	birth_date
1	Garcia	Luis	1982-02-10
2	Smith	John	1985-06-12
3	Johnson	Emily	1990-09-20

7. Cuento cuántos clientes viven en cada ciudad, excepto en Knoxville y Stockton. Ordene los resultados por el nombre de la ciudad en orden ascendente. Muestre dos columnas: city y customers_quantity.

The screenshot shows a PostgreSQL IDE with two tabs: "<postgres> Script-5" and "*<postgres> Script-6". The active tab is "Script-6", which contains the following SQL query:

```
-- septima consulta
select c.city, count(c.city) as customers_quantity from customer c
where c.city not in ('Knoxville', 'Stockton') group by c.city order by c.city asc
```

Below the query editor, there is a section labeled "customer 1" with a search bar containing "select c.city, count(c.city) as customers_quant". Below the search bar, there is a table with 5 rows and 2 columns. The first column is labeled "city" and the second column is labeled "customers_quantity". The table shows the following data:

	city	customers_quantity
1	Atlanta	2
2	Dallas	1
3	Houston	1
4	Miami	1
5	Orlando	1

8. Para cada categoría, encuentre el número de productos descatalogados. Muestre sólo las categorías con al menos tres productos descatalogados. Ordene las filas por el número de productos descatalogados en orden descendente. Muestre dos columnas: name (el nombre de la categoría) y discontinued_products_number.

The screenshot shows a PostgreSQL IDE with two tabs: "<postgres> Script-5" and "*<postgres> Script-6". The active tab is "Script-6", which contains the following SQL query:

```
-- octava consulta
select c.name as category_name, count(p.discontinued) as discontinued_products_number from product p
join category c on p.category_id = c.category_id where p.discontinued = true group by c.name
having count(p.discontinued) >= 3 order by discontinued_products_number desc
```

Below the query editor, there is a section labeled "category 1" with a search bar containing "select c.name as category_name, count(p.dis". Below the search bar, there is a table with 2 rows and 2 columns. The first column is labeled "category_name" and the second column is labeled "discontinued_products_number". The table shows the following data:

	category_name	discontinued_products_number
1	Snacks	4
2	Fruits and Vegetables	3