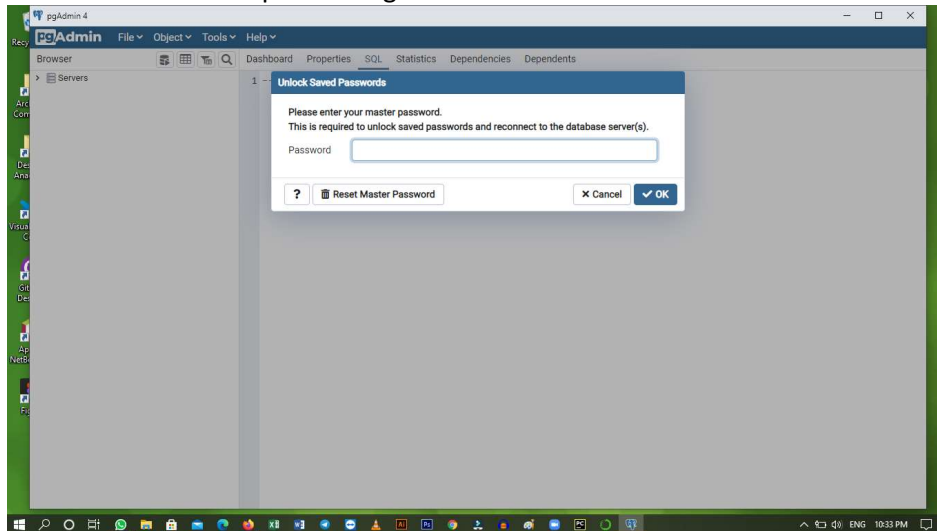
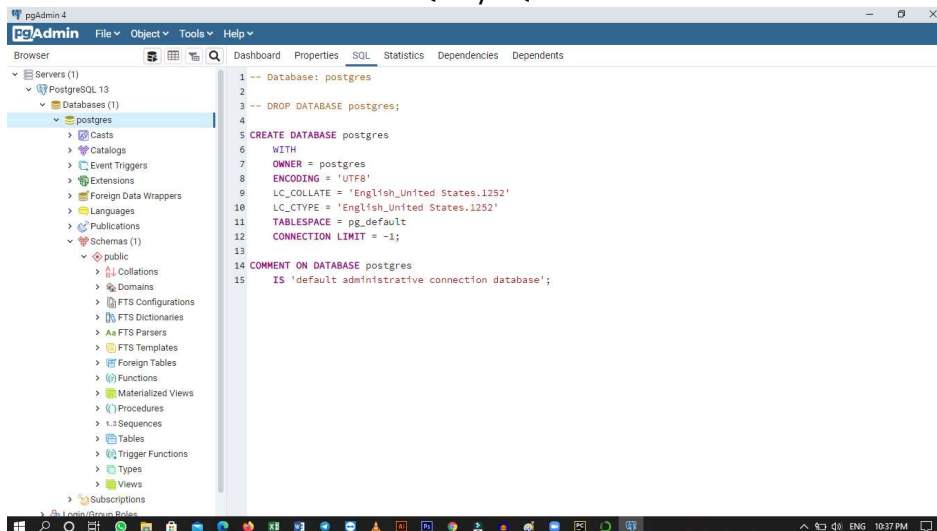


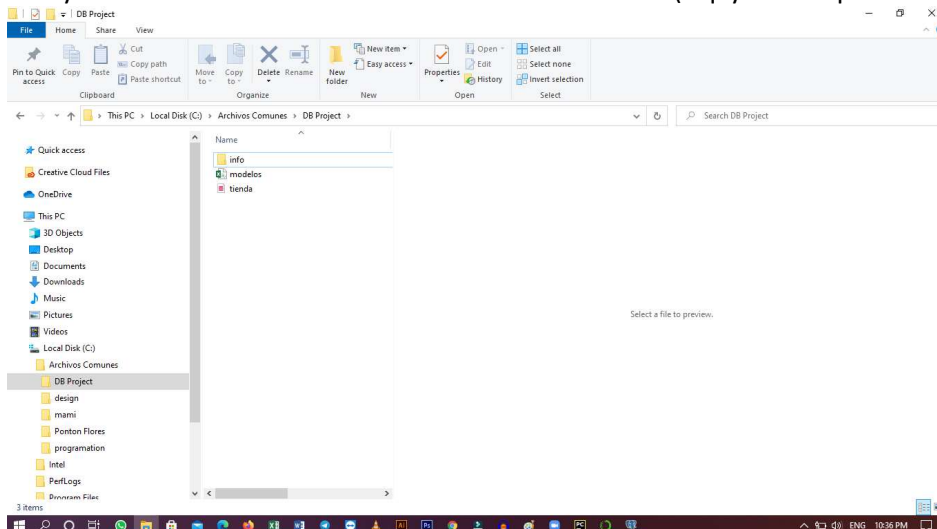
1. Luego de instalar PostgreSQL v.13, accedemos al pgAdmin y accedemos al servidor por medio de la password generada en la instalación



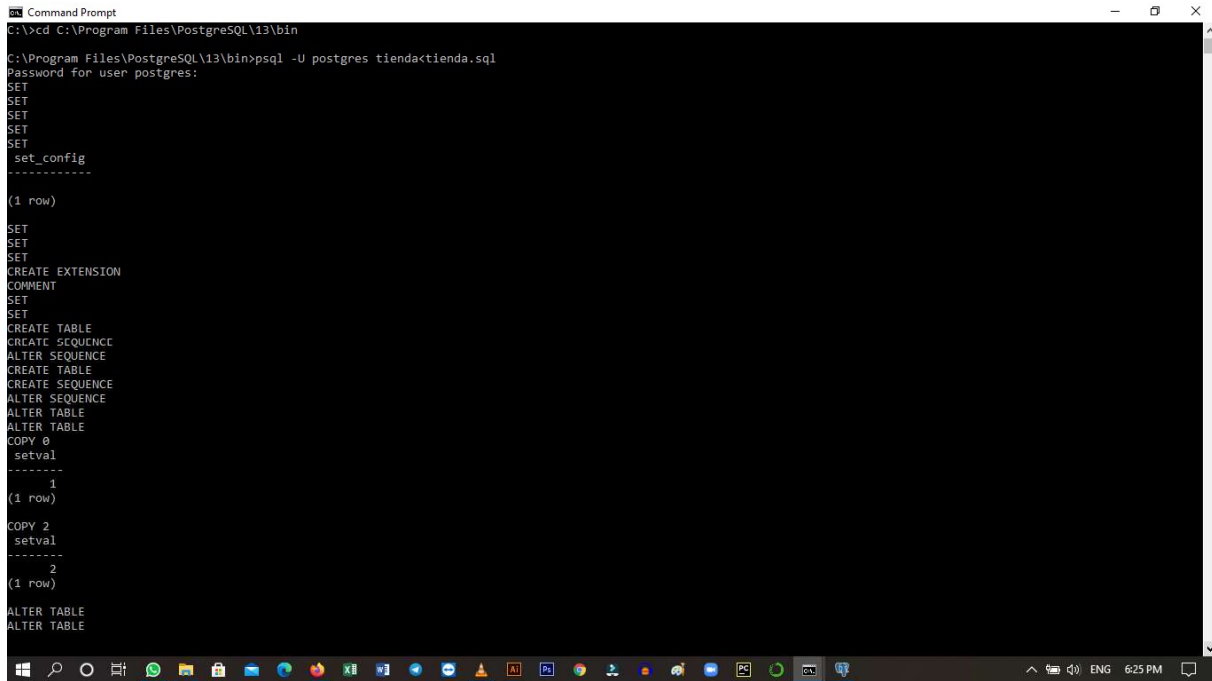
2. Accedemos al servidor y revisamos la base de datos generada ya en la instalación y podemos observar el Query SQL



3. Verificamos el lugar en donde se encuentran los archivos para recuperacion de la DB de la Tienda de Celulares ya existente asi como de los datos con los modelos (.sql y .cvs respectivamente)



4. Para el restore se copia el archivo sql a la direccion en donde se encuentra instalado postgresQL, esto es en C:\Program Files\PostgreSQL\13\bin.
- Luego a traves del Shell de windows, desde la direccion indicada anteriormente, escribimos el comando sql:
- ```
C:\Program Files\PostgreSQL\13\bin>psql -U postgres tienda<tienda.sql
```
- Se ingresa la contraseña:  
Password for user postgres:
- Y el proceso de Restore empieza generandose la restauracion de las tablas dentro de la Base de Datos tienda creado con anterioridad.



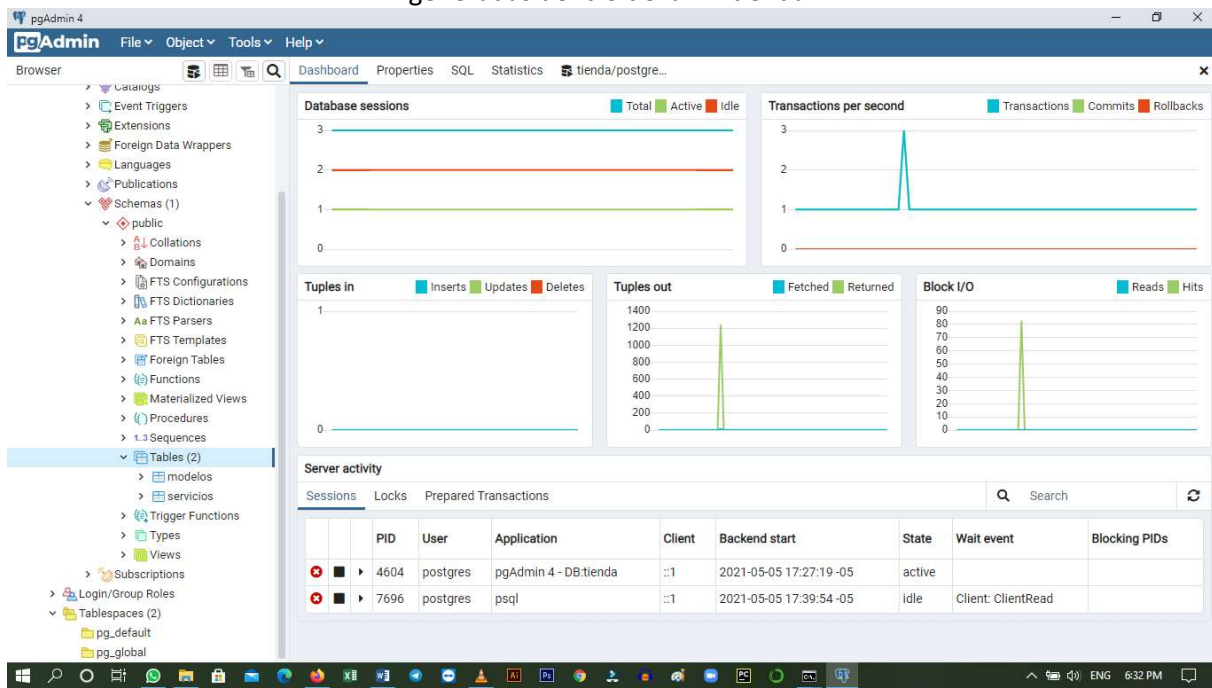
```
Command Prompt
C:\>cd C:\Program Files\PostgreSQL\13\bin
C:\Program Files\PostgreSQL\13\bin>psql -U postgres tienda<tienda.sql
Password for user postgres:
SET
SET
SET
SET
SET
SET
SET
set_config

(1 row)
SET
SET
SET
SET
CREATE EXTENSION
COMMENT
SET
SET
CREATE TABLE
CREATE SEQUENCE
ALTER SEQUENCE
CREATE TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER TABLE
ALTER TABLE
COPY 0
setval

1
(1 row)
COPY 2
setval

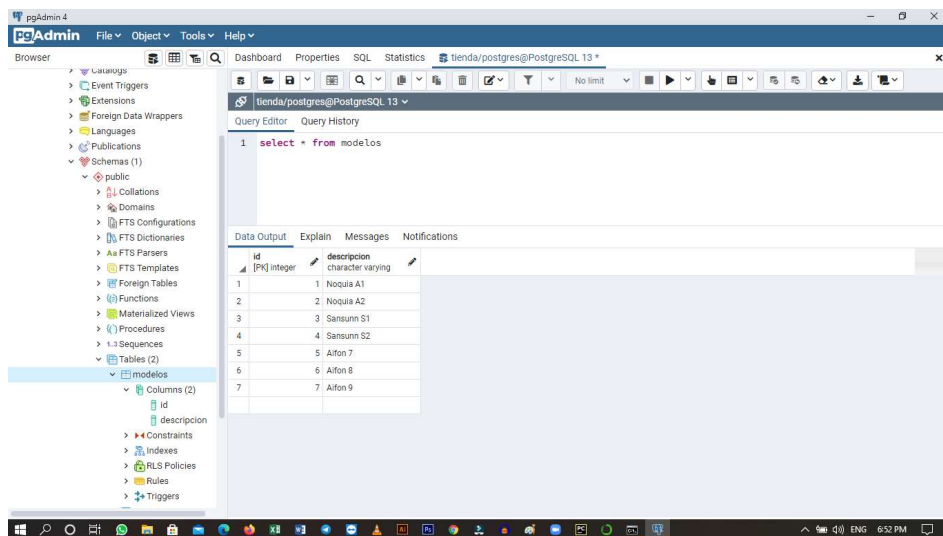
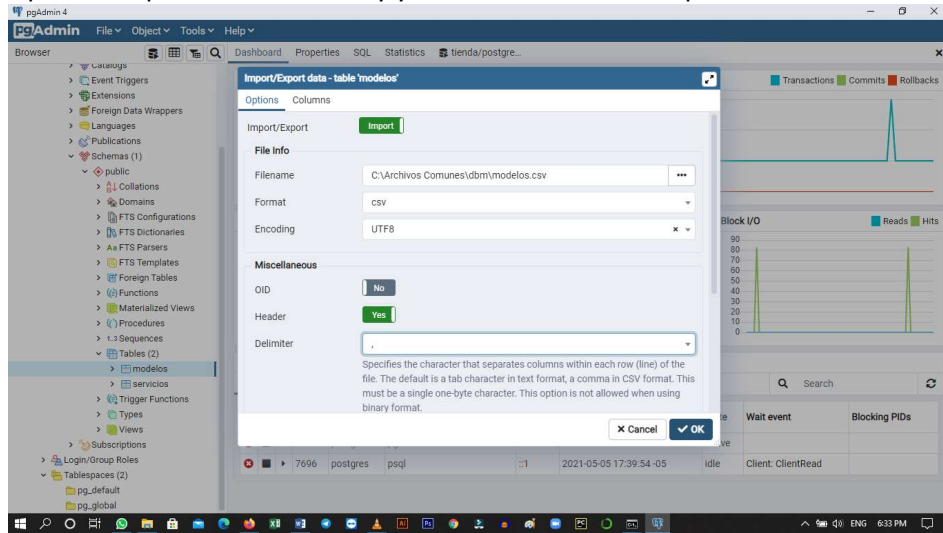
2
(1 row)
ALTER TABLE
ALTER TABLE
```

5. Ahora desde pgAdmin, el IDE de postgresQL, refrescamos el servidor, luego verificamos las tablas ya generadas dentro de la DB tienda.

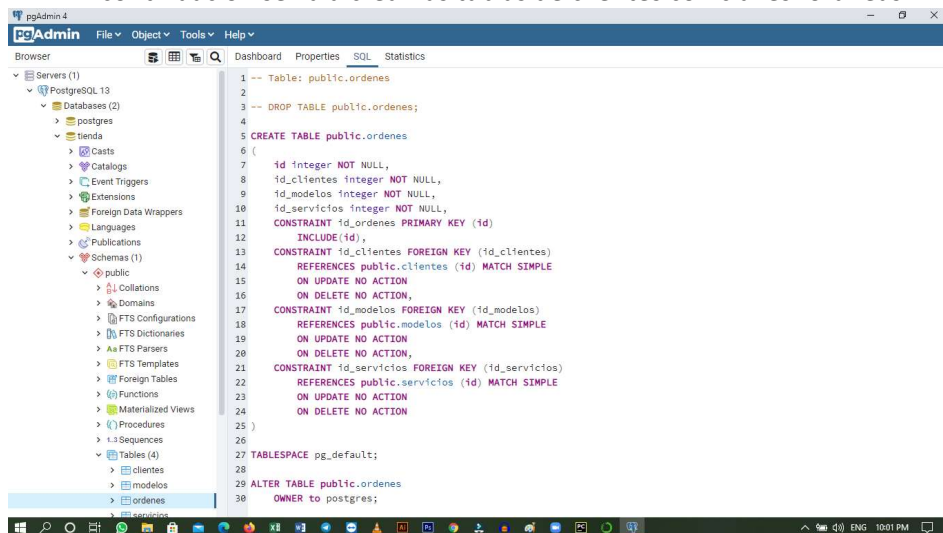


## 6. A través del pgAdmin, sobre la tabla modelos.

Accedemos a las opciones del mismo y buscamos el archivo cvs con el cual según la grafica, configuramos para la importacion de datos y ya tenemos los datos importados en la tabla.



## 7. A continuacion se va a crear las tablas de clientes con claves foraneas.



## 8. Revisamos los datos de cada tabla

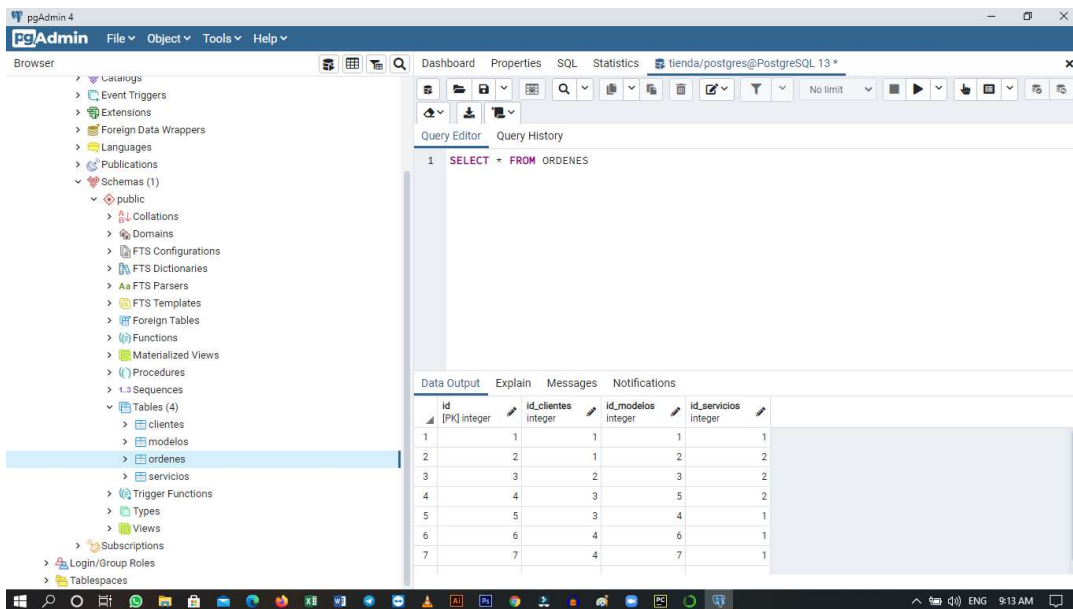
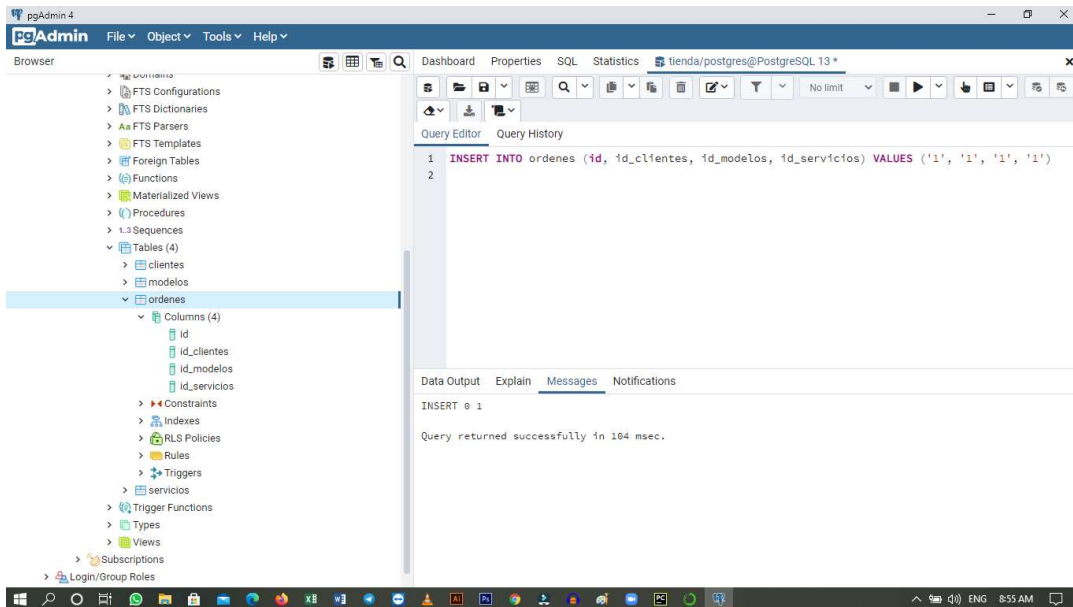
The following tables represent the data shown in the screenshots:

| id | nombre      | telefono  |
|----|-------------|-----------|
| 1  | Luis Torres | 55555555  |
| 2  | Ana         | 123123123 |
| 3  | Maria       | 11111111  |
| 4  | Pedro       | 22222222  |

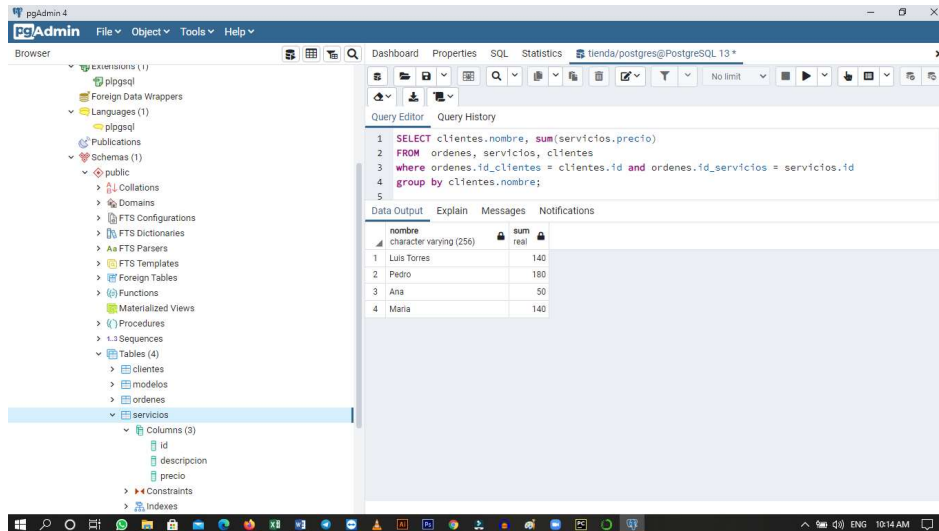
| id | descripcion |
|----|-------------|
| 1  | Noquia A1   |
| 2  | Noquia A2   |
| 3  | Sansunn S1  |
| 4  | Sansunn S2  |
| 5  | Aifon 7     |
| 6  | Aifon 8     |
| 7  | Aifon 9     |

| id | descripcion     | precio |
|----|-----------------|--------|
| 1  | cambio pantalla | 90     |
| 2  | cambio bateria  | 50     |

8. A continuacion vamos a llenar la tabla de ordenes que el cliente tenia en una hoja a traves de Query Tool. Según el diseno escogido, es esta tabla unicamente se va a ingresar los datos usando los id de cada tabla (Usuarios, Modelos y Servicios) lo que, según diseno personalizado, es atraves de llaves foraneas. La orden seria la siguiente según foto para rellenar la tabla de ordenes. Se realiza un select \* sobre la tabla ordenes mostrando el contenido de los datos.



9. Realizamos la primera consulta SQL en donde el resultado debe ser el nombre del cliente y el costo total de todos sus servicios.



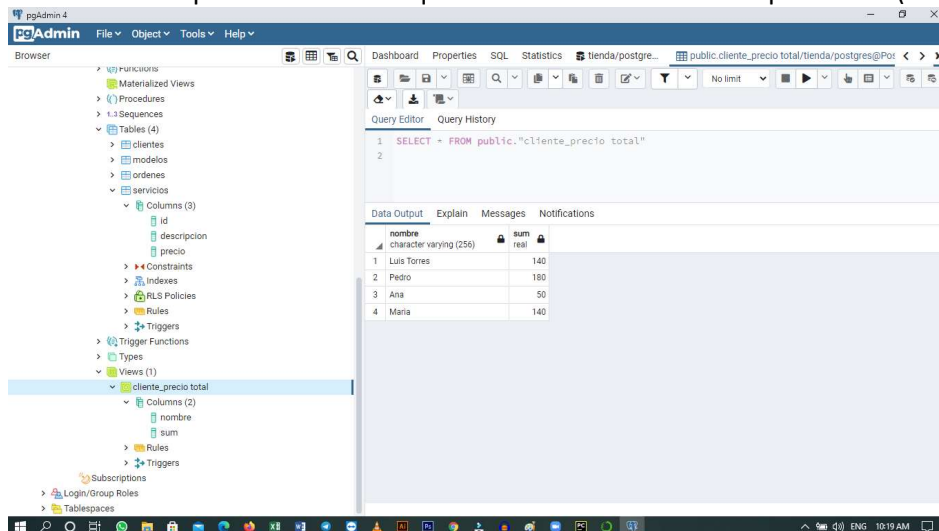
The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'servicios' table under the 'public' schema selected. The main window shows a SQL query in the Query Editor:

```
1 SELECT clientes.nombre, sum(servicios.precio)
2 FROM ordenes, servicios, clientes
3 where ordenes.id_clientes = clientes.id and ordenes.id_servicios = servicios.id
4 group by clientes.nombre;
```

The Data Output tab shows the results of the query:

| nombre                  | sum  |
|-------------------------|------|
| character varying (256) | real |
| 1 Luis Torres           | 140  |
| 2 Pedro                 | 180  |
| 3 Ana                   | 50   |
| 4 Maria                 | 140  |

9.1 Se genera una vista (cliente\_preciototal) para que se genere ya que es una consulta recurrente sobre cartera de la empresa dirigida a los clientes y sus valores pendientes (podríamos ingresar en la tabla de ordenes el estado con un campo boolean con el que sabríamos si el valor esta pendiente (true) o no (false)).



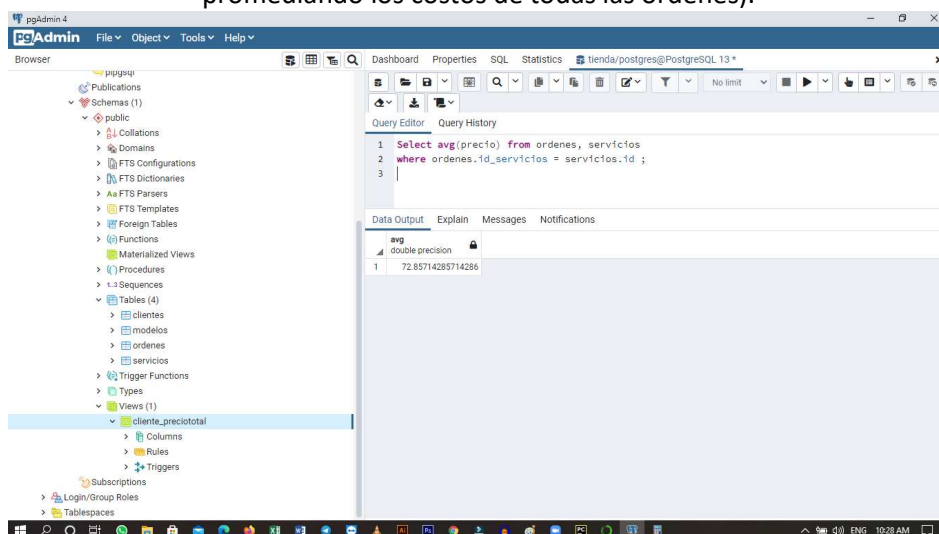
The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'cliente\_preciototal' view under the 'public' schema selected. The main window shows a SQL query in the Query Editor:

```
1 SELECT * FROM public."cliente_precio total"
```

The Data Output tab shows the results of the query:

| nombre                  | sum  |
|-------------------------|------|
| character varying (256) | real |
| 1 Luis Torres           | 140  |
| 2 Pedro                 | 180  |
| 3 Ana                   | 50   |
| 4 Maria                 | 140  |

10. Ahora generamos un SQL cuyo resultado sea el costo promedio actual de toda la tienda (que se obtiene promediando los costos de todas las órdenes).



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'cliente\_preciototal' view under the 'public' schema selected. The main window shows a SQL query in the Query Editor:

```
1 Select avg(precio) from ordenes, servicios
2 where ordenes.id_servicios = servicios.id ;
3
```

The Data Output tab shows the results of the query:

| avg                 |
|---------------------|
| double precision    |
| 1 72.85714285714286 |