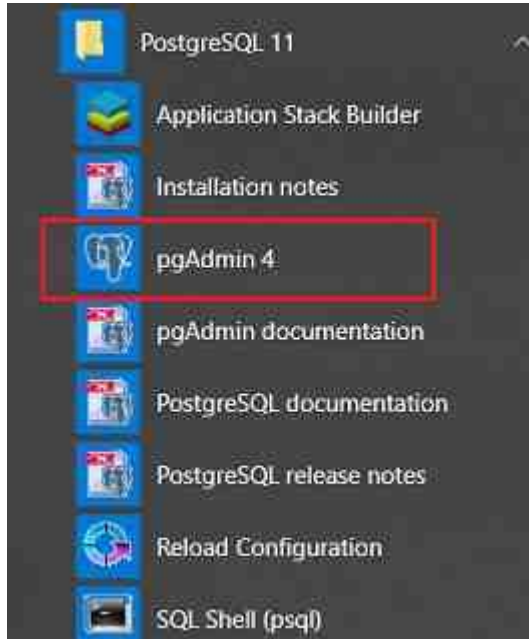
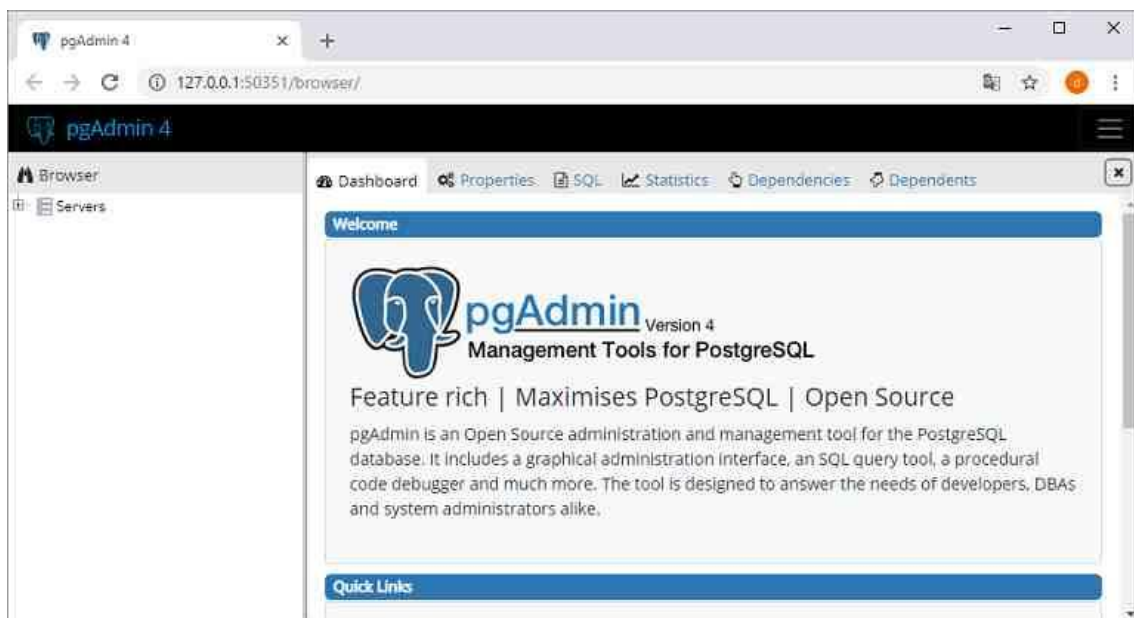


Creación de una base de datos y una tabla desde el programa pgAdmin.

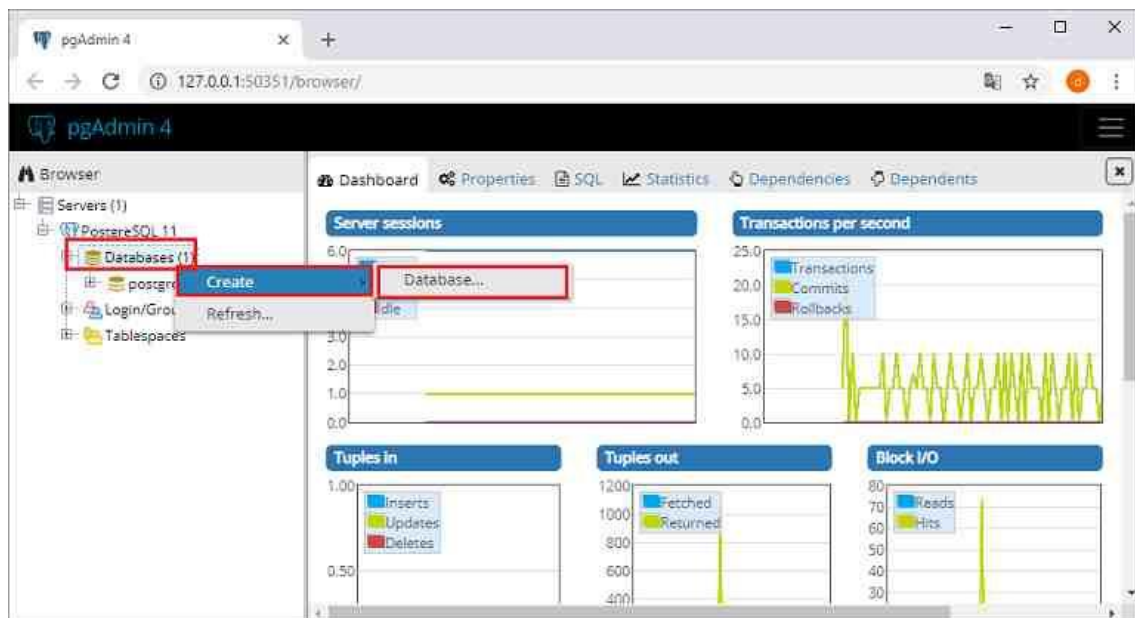
Ejecutemos el programa pgAdmin accediendo desde el menú de Windows:



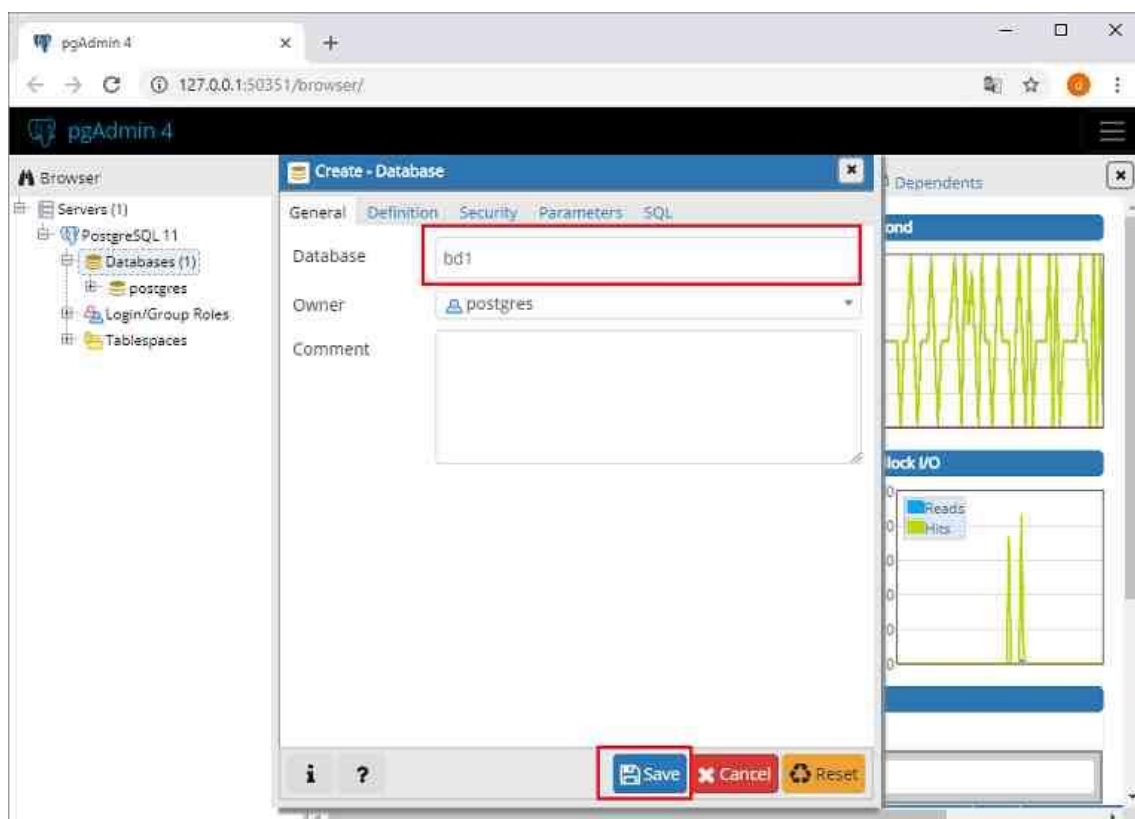
Se abre el navegador con la aplicación:



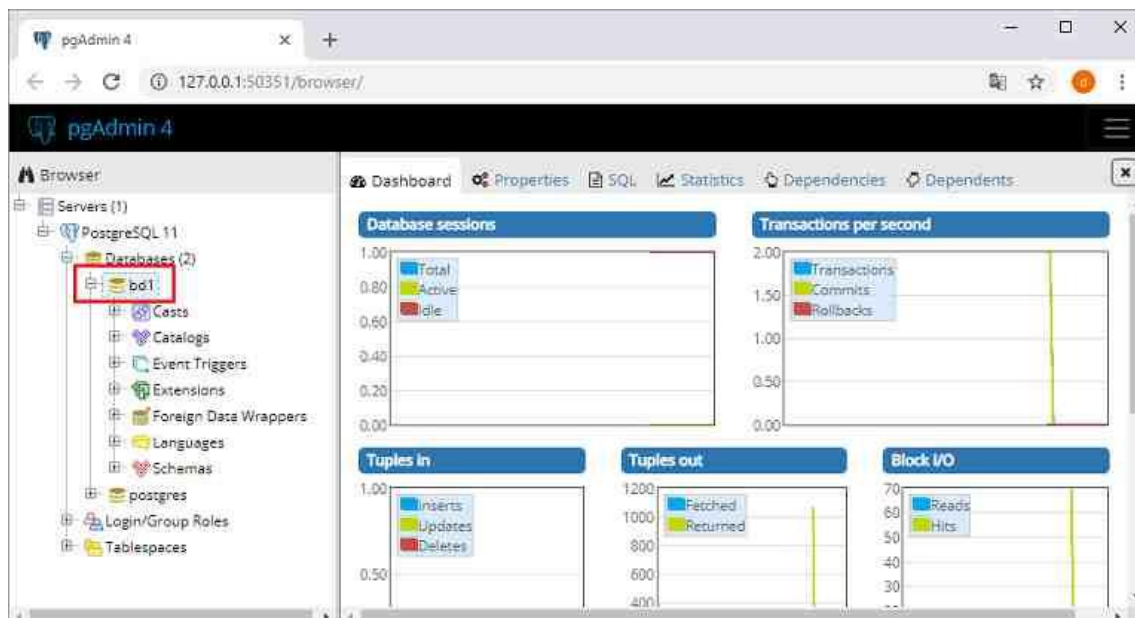
Para crear una base de datos presionamos el botón derecho del mouse donde dice "Databases" y seleccionamos la opción "Create" -> "Database...":



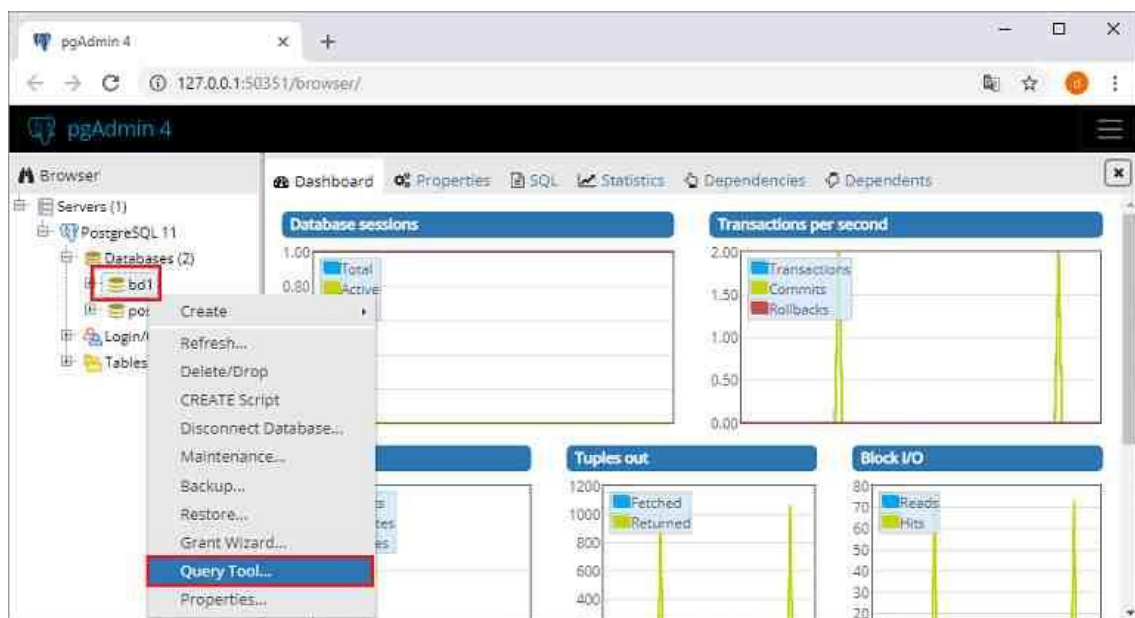
Aparece un diálogo donde debemos ingresar el nombre de la base de datos a crear, la llamaremos "bd1":



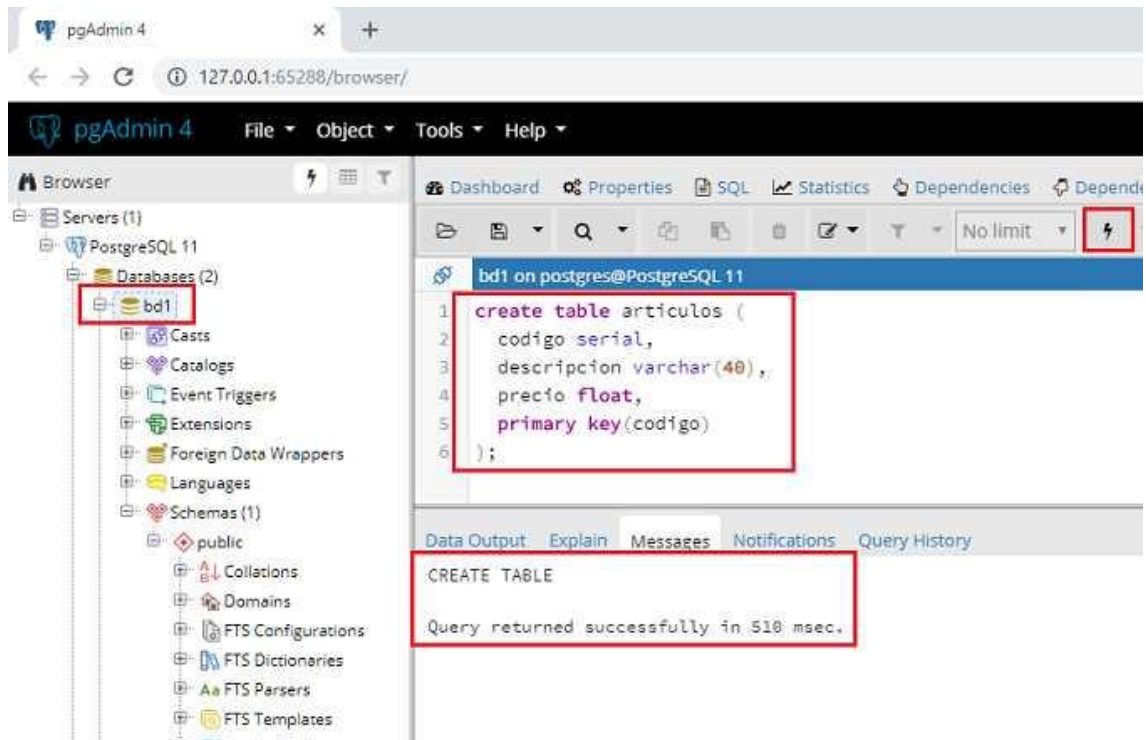
Ahora podemos seleccionar en la ventana de la izquierda la base de datos "bd1" que acabamos de crear:



Para poder ejecutar comandos SQL debemos presionar el botón derecho del mouse sobre el nombre de la base de datos "bd1" y seleccionar la opción "Query Tool..":



Ahora crearemos la tabla articulos en la base de datos "bd1":



Paquete de Python necesario para conectarnos a PostgreSQL.

Utilizaremos el programa 'pip' que vimos anteriormente para instalar el paquete necesario para interconectar 'Python' y 'PostgreSQL'.

Desde la línea de comandos ejecutamos el programa pip con el siguiente paquete a instalar:

```
pip install psycopg2
```

Luego de ejecutar el programa pip podemos ver que nos informa que la instalación del paquete se efectuó correctamente:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

C:\programaspython>pip install psycopg2
Collecting psycopg2
  Using cached https://files.pythonhosted.org/packages/86/e9/3165d3f4023d9c91a116286bdba0aa2bdde72e787acf6161f4929d9f1
Installing collected packages: psycopg2
Successfully installed psycopg2-2.7.6.1

C:\programaspython>
```

Conexión con el servidor de PostgreSQL.

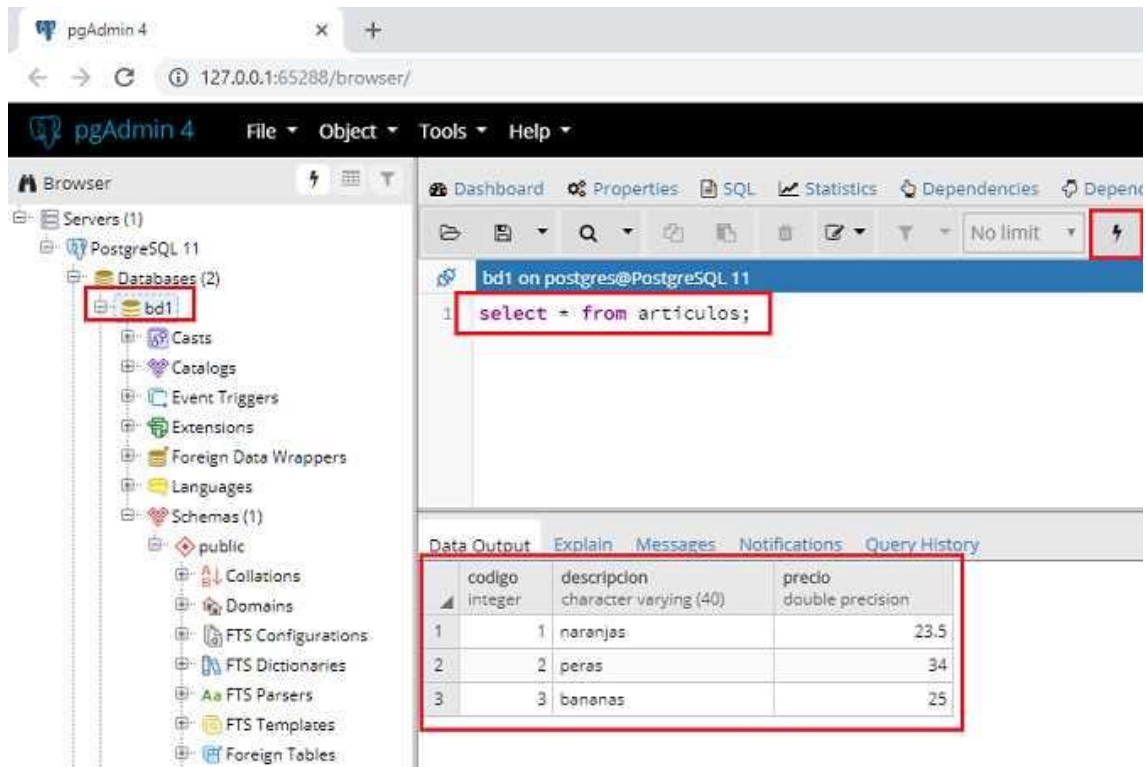
El primer programa que implementaremos nos conectaremos con el servidor de PostgreSQL e insertaremos un par de filas en la tabla 'articulos' que creamos desde el programa 'pgAdmin'.

```
import psycopg2

conexion1 = psycopg2.connect(database="bd1", user="postgres", password="heladera")
cursor1=conexion1.cursor()
sql="insert into articulos(descripcion, precio)
values (%s,%s)"
datos=("naranjas", 23.50)
cursor1.execute(sql, datos)
datos=("peras", 34)
cursor1.execute(sql, datos)
datos=("bananas", 25)
cursor1.execute(sql, datos)
conexion1.commit()
conexion1.close()
```

Ejecutemos este programa para que se efectúe la carga de las tres filas en la tabla 'articulos' de la base de datos 'bd1'.

Por el momento si queremos controlar que se han cargado las tres filas en la tabla 'articulos' podemos abrir el 'pgAdmin' que viene con PostgreSQL y ver el contenido de la tabla:



Lo primero que hacemos es importar el módulo que nos permite conectarnos con PostgreSQL:

```
import psycopg2
```

Del módulo importado llamamos a la función connect pasando la ubicación con el nombre de la base de datos, nombre de usuario y la clave de dicho usuario:

```
conexion1 = psycopg2.connect(database="bd1", user="postgres", password="helad  
era")
```

Si por ejemplo el servidor de PostgreSQL no se encuentra en ejecución el programa se detendrá en esta línea informando un error.

Luego a partir del objeto 'conexion1' llamamos al método 'cursor':

```
cursor1=conexion1.cursor()
```

Definimos un string con el comando SQL insert disponiendo la máscara %s donde queremos que se sustituya por un valor que le pasaremos al método execute:

```
sql="insert into articulos(descripcion, precio) values (%s,%s)"
```

La variable datos es una tupla que contiene los datos que se utilizarán en la sustitución %s:

```
datos=("naranjas", 23.50)
```

Finalmente llamamos al método 'execute' y le pasamos las dos variables que acabamos de crear:

```
cursor1.execute(sql, datos)
```

Es fundamental llamar al final al método 'commit' para que queden firmes los comandos SQL 'insert':

```
conexion1.commit()
```

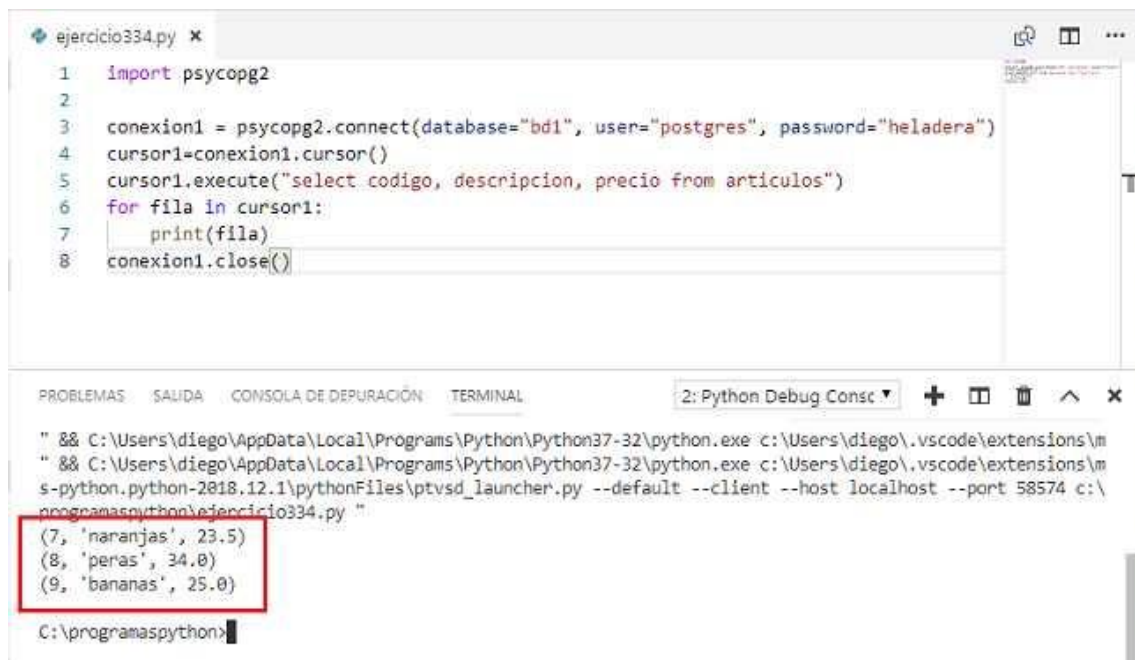
Recuperar todas las filas de una tabla.

Implementaremos un programa que solicite ejecutar un 'select' en la tabla 'articulos' de la base de datos 'bd1' y nos retorne todas sus filas.

```
import psycopg2

conexion1 = psycopg2.connect(database="bd1", user="postgres", password="123456")
cursor1=conexion1.cursor()
cursor1.execute("select codigo, descripcion, precio from articulos")
for fila in cursor1:
    print(fila)
conexion1.close()
```

Cuando ejecutamos el programa podemos ver que se recuperan todas las filas de la tabla 'articulos':



The screenshot shows a VS Code editor window with a file named 'ejercicio334.py'. The code in the editor is as follows:

```
1 import psycopg2
2
3 conexion1 = psycopg2.connect(database="bd1", user="postgres", password="heladera")
4 cursor1=conexion1.cursor()
5 cursor1.execute("select codigo, descripcion, precio from articulos")
6 for fila in cursor1:
7     print(fila)
8 conexion1.close()
```

Below the editor, the 'TERMINAL' panel is open, showing the command used to run the script and its output:

```
" && C:\Users\diego\AppData\Local\Programs\Python\Python37-32\python.exe c:\Users\diego\.vscode\extensions\ms-python.python-2018.12.1\pythonFiles\ptvsd_launcher.py --default --client --host localhost --port 58574 c:\programaspython\ejercicio334.py "
```

The output of the script is displayed in the terminal, with the last three lines highlighted by a red box:

```
(7, 'naranjas', 23.5)
(8, 'peras', 34.0)
(9, 'bananas', 25.0)
```

The terminal prompt is 'C:\programaspython>'.

Luego de conectarnos y crear un cursor procedemos a ejecutar el comando 'select', recorremos con un for el 'cursor1':

```
cursor1=conexion1.cursor()
cursor1.execute("select codigo, descripcion, precio from articulos")
for fila in cursor1:
    print(fila)
```

Borrado y modificación de filas.

Las otras dos actividades fundamentales que podemos hacer con una tabla es borrar filas y modificar datos.

Desarrollaremos un pequeño programa que borre el artículo cuyo código sea el 1 y modifique el precio del artículo cuyo código sea 3.

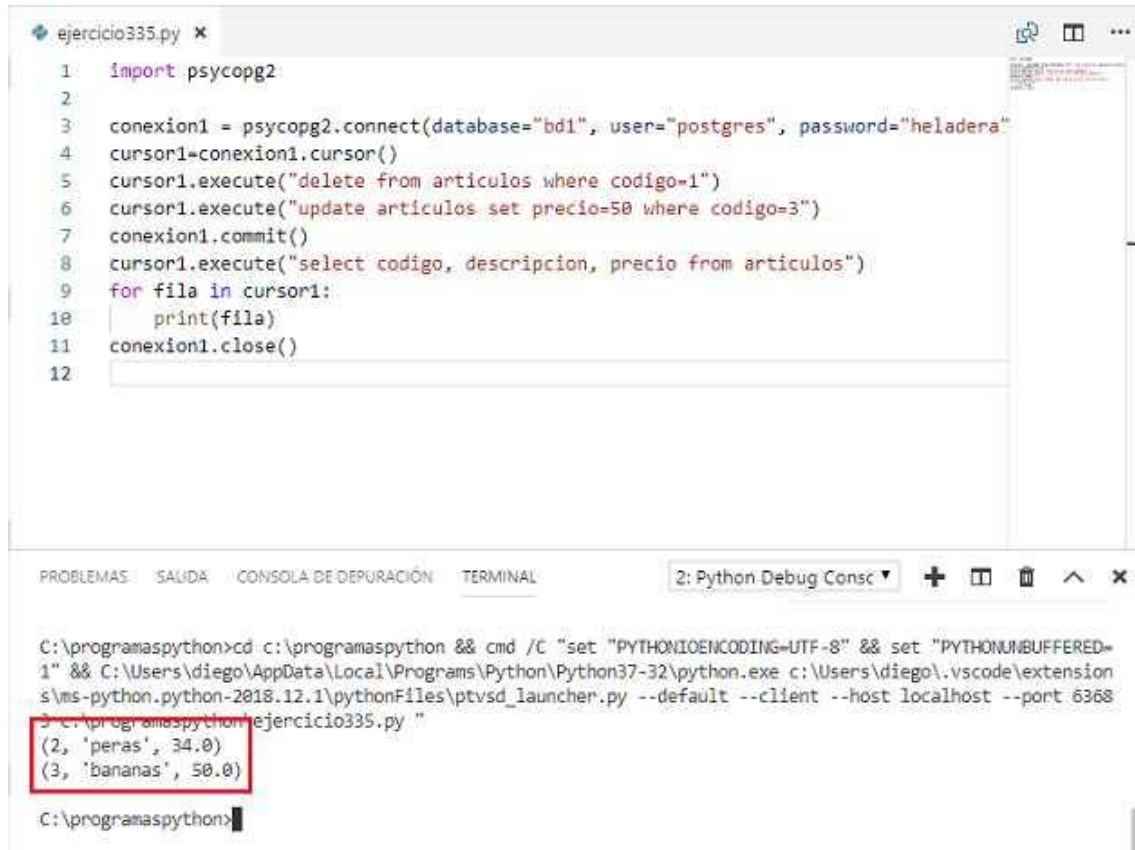
```
import psycopg2

conexion1 = psycopg2.connect(database="bd1", user="postgres", password="123456")
cursor1=conexion1.cursor()
cursor1.execute("delete from articulos where codigo=1")
cursor1.execute("update articulos set precio=50 where codigo=3")
conexion1.commit()
```



```
cursor1.execute("select codigo, descripcion, precio from articulos")
for fila in cursor1:
    print(fila)
conexion1.close()
```

Cuando ejecutamos el programa podemos ver que se eliminó el artículo cuyo código es 1 y se modificó el precio del artículo con código 3:



```
ejercicio335.py x
1 import psycopg2
2
3 conexion1 = psycopg2.connect(database="bd1", user="postgres", password="heladera")
4 cursor1=conexion1.cursor()
5 cursor1.execute("delete from articulos where codigo=1")
6 cursor1.execute("update articulos set precio=50 where codigo=3")
7 conexion1.commit()
8 cursor1.execute("select codigo, descripcion, precio from articulos")
9 for fila in cursor1:
10     print(fila)
11 conexion1.close()
12
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL 2: Python Debug Consol

```
C:\programaspython>cd c:\programaspython && cmd /c "set "PYTHONIOENCODING=UTF-8" && set "PYTHONUNBUFFERED=1" && C:\Users\diego\AppData\Local\Programs\Python\Python37-32\python.exe c:\Users\diego\.vscode\extension s\ms-python.python-2018.12.1\pythonFiles\ptvsd_launcher.py --default --client --host localhost --port 6368"
C:\programaspython>ejercicio335.py "
(2, 'peras', 34.0)
(3, 'bananas', 50.0)
C:\programaspython>
```

Luego de crear el cursor podemos llamar al método 'execute' varias veces y pasar distintos comando SQL:

```
cursor1=conexion1.cursor()
cursor1.execute("delete from articulos where codigo=1")
cursor1.execute("update articulos set precio=50 where codigo=3")
```

Siempre que pasemos un comando SQL: insert, delete o update debemos llamar al método commit para que quede firme los cambios en la base de datos:

```
conexion1.commit()
```

Ejecutamos finalmente un 'select' para comprobar los cambios efectuados en la tabla 'articulos':

```
cursor1.execute("select codigo, descripcion, precio from articulos")
for fila in cursor1:
    print(fila)
```