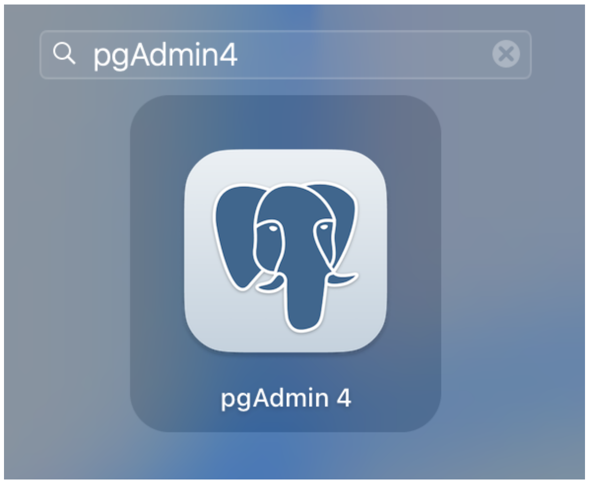
Lesson-13 – DB Setup

Steps to connect PostgreSQL with JDBC API

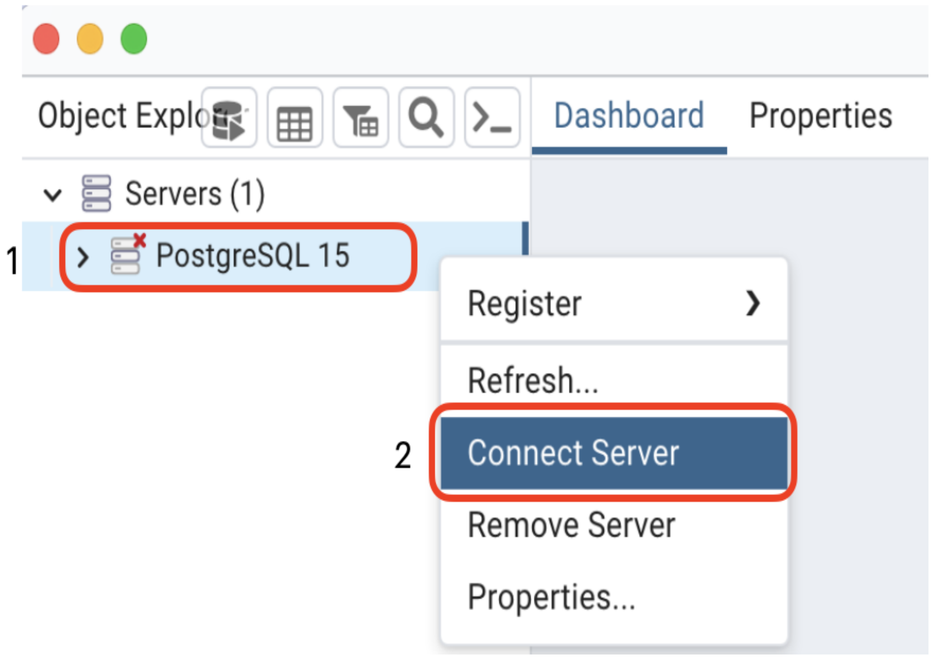
1. Download PostgresSQL based on your OS
   https://www.postgresql.org/download/

   Click on the downloaded file and follow the installation instructions.

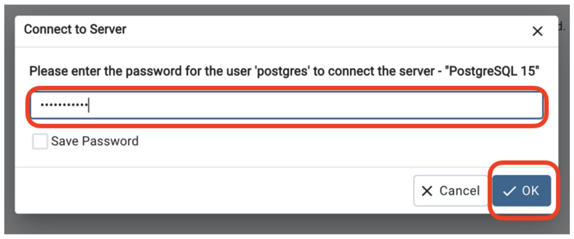
2. Go to Application, click "pgAdmin 4"

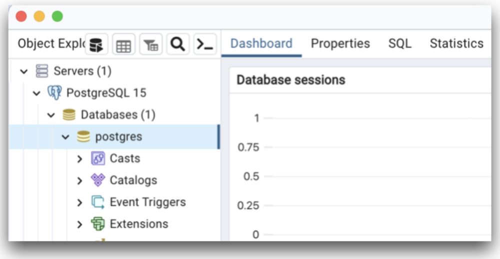


3. Right-click PostgreSQL and click "Connect Server"

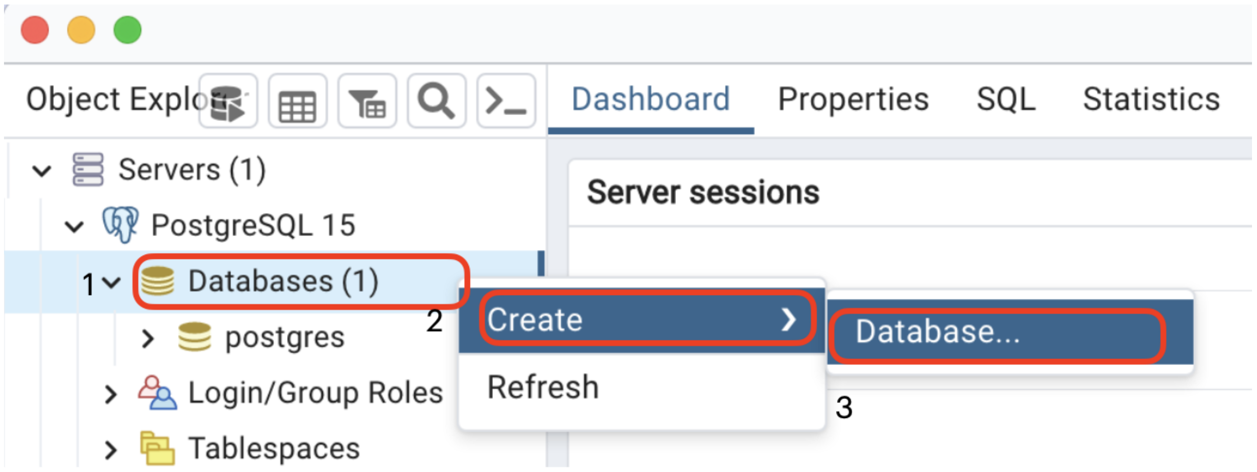


4. Enter the password and click OK.

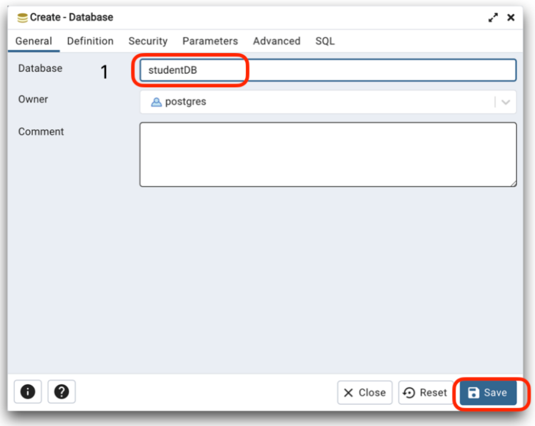

Connect to Server
Please enter the password for the user 'postgres' to connect the server - "PostgreSQL 15"
1
Save Password
Cancel
OK
2

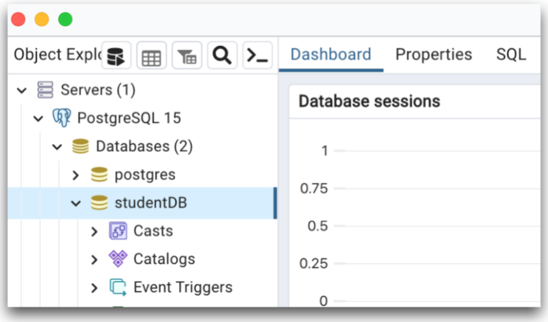


5. Right-click Database and create a new database.

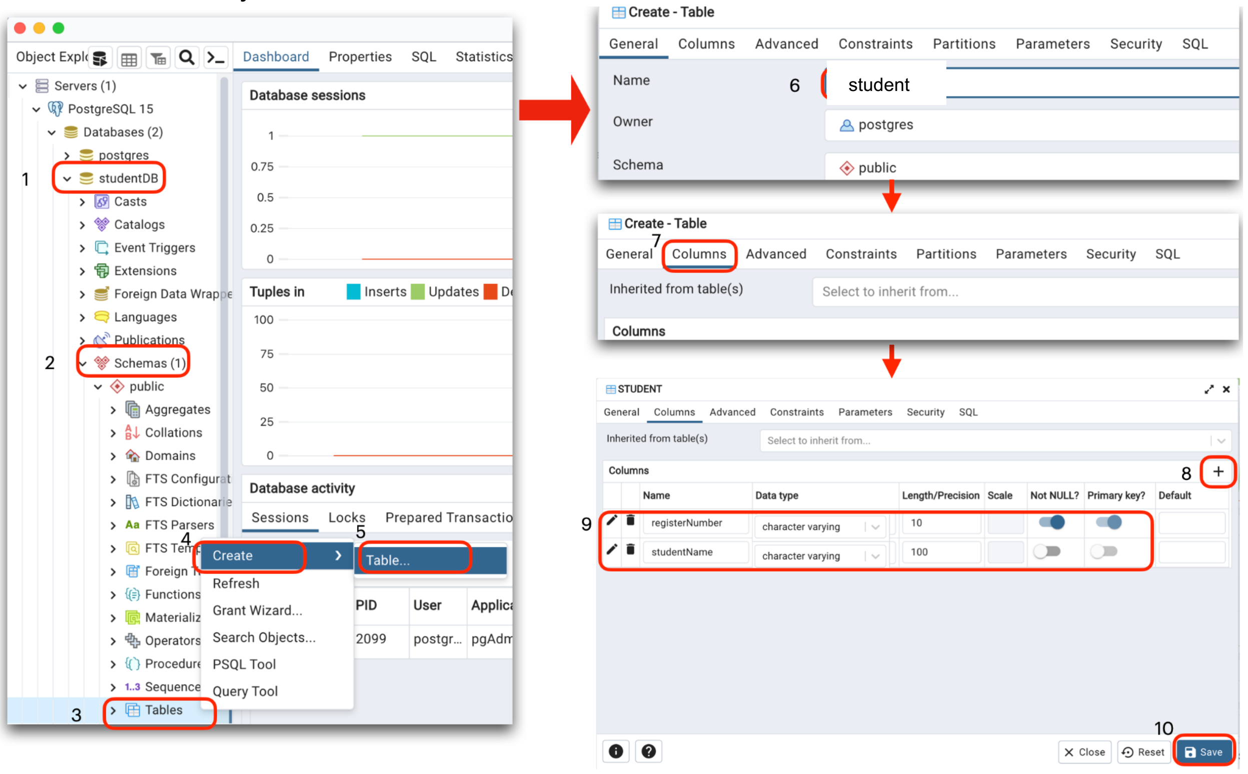


6. Enter a database name and click the button, Save.



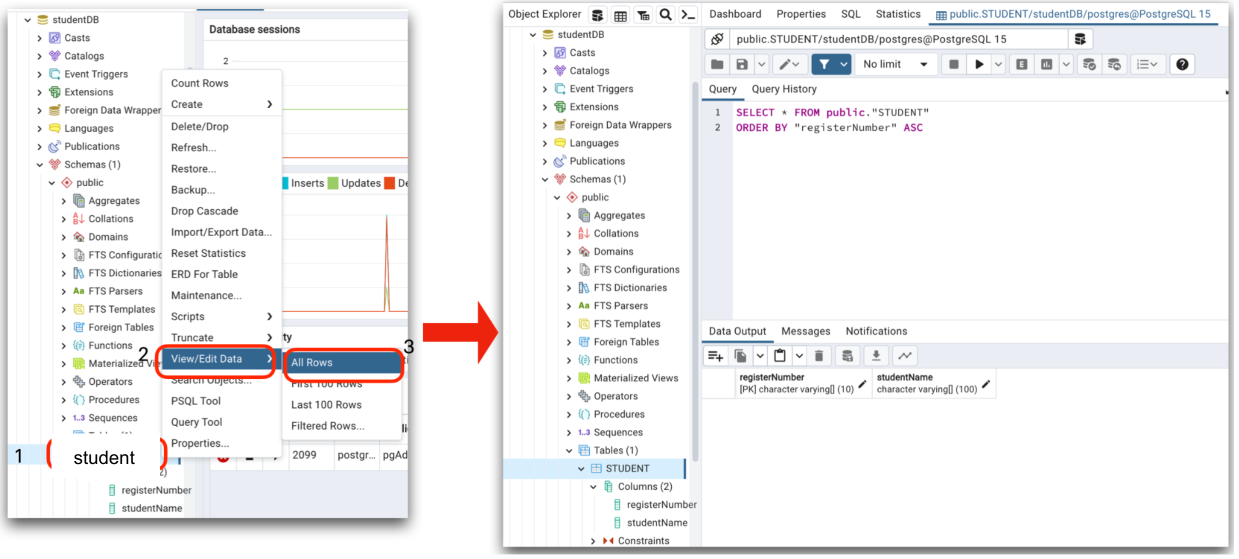


7. Create a table under Schemas as follows with necessary columns. Prefer to use lowercase for the table name(avoid case sensitive issue). Choose Columns and click + to add fields. Finally save it.



Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
registerNumber	character varying	10		✓	✓	
studentName	character varying	100				

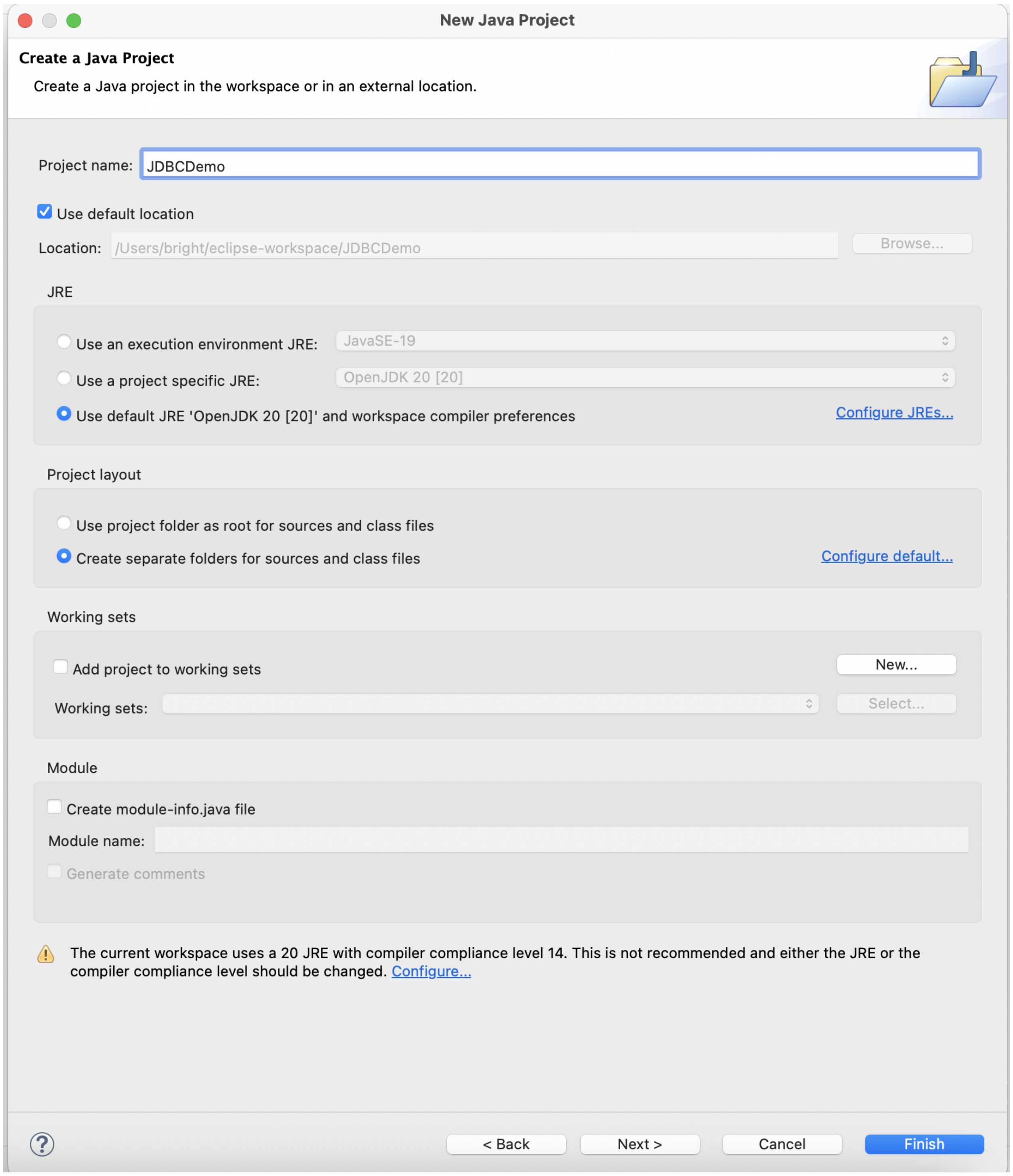
8. To view the table



```
SELECT * FROM public."STUDENT"
ORDER BY "registerNumber" ASC
```

registerNumber	studentName
[PK] character varying] (10)	character varying] (100)

9. Create a new Java Project.



Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: JDBCDemo

[✓] Use default location

Location: /Users/bright/eclipse-workspace/JDBCDemo

JRE

( ) Use an execution environment JRE: JavaSE-19

( ) Use a project specific JRE: OpenJDK 20 [20]

(●) Use default JRE 'OpenJDK 20 [20]' and workspace compiler preferences

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files

Working sets

[ ] Add project to working sets

Working sets:

Module

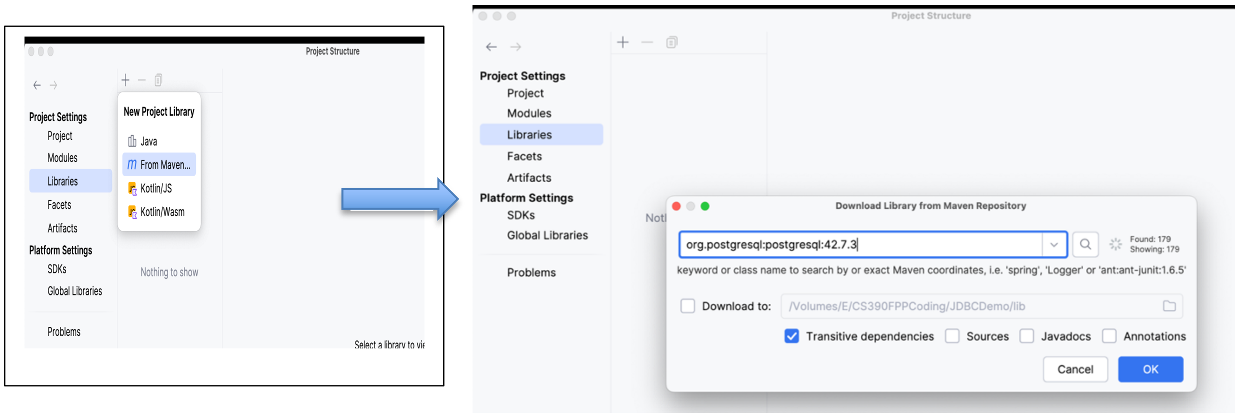
[ ] Create module-info.java file

Module name:

[ ] Generate comments

The current workspace uses a 20 JRE with compiler compliance level 14. This is not recommended and either the JRE or the compiler compliance level should be changed. Configure...

10. Add postgresql jar file into your project follows. Right click project > Open Module Settings > Libraries, Click +, type the driver file needs to add as mentioned. Finally Apply and Click OK.



org.postgresql:postgresql:42.7.3

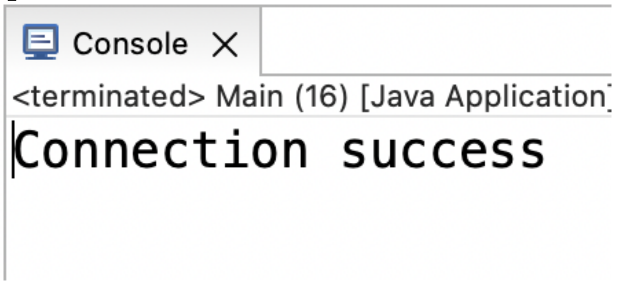
keyword or class name to search by or exact Maven coordinates, i.e. 'spring', 'Logger' or 'ant:ant-junit:1.6.5'

☐ Download to: /Volumes/E/CS390FPPCoding/JDBCDemo/lib

☑ Transitive dependencies ☐ Sources ☐ Javadocs ☐ Annotations

11. Sample code to get the connection through JDBC API as follows.

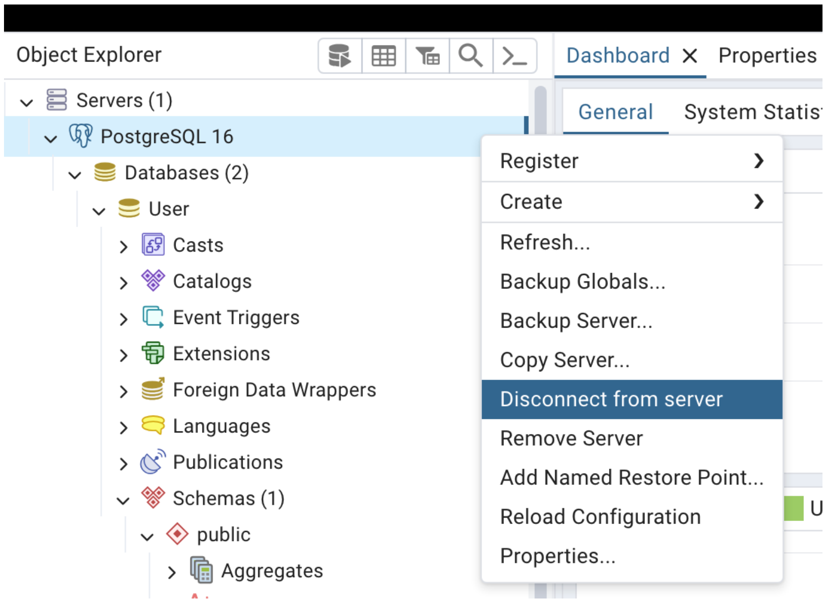
```
public static void main(String[] args) {
// User is your Database name
    String url = "jdbc:postgresql://localhost:5432/User";
    String uname = "postgres"; // Username - Common
    String pass = "1234"; // Password you created
    try {
        Connection con =
DriverManager.getConnection(url,uname,pass);
        System.out.println("Connection Success");
            } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
Output for the successful connection.
```



```
<terminated> Main (16) [Java Application]
Connection success
```

Refer: JDBCDemo Project

Step 12: After completion, you can disconnect the server.



Database

A **database** is a structured collection of related data, stored and managed by a **Database Management System (DBMS)**for easy retrieval, insertion, and management.

Table

A **table** is a database object that organizes data into **rows and columns**, where each row is a record and each column is a field/attribute.

Schema

A **schema** is the logical structure of the database. It defines how the data is organized, including tables, columns, data types, keys, and relationships.

SQL (Structured Query Language) is a **standard programming language** designed to manage and manipulate data in a **relational database management system (RDBMS)**.

It is used to:

- **Define** database structures (`CREATE`, `ALTER`, `DROP`).
- **Query** data (`SELECT`).

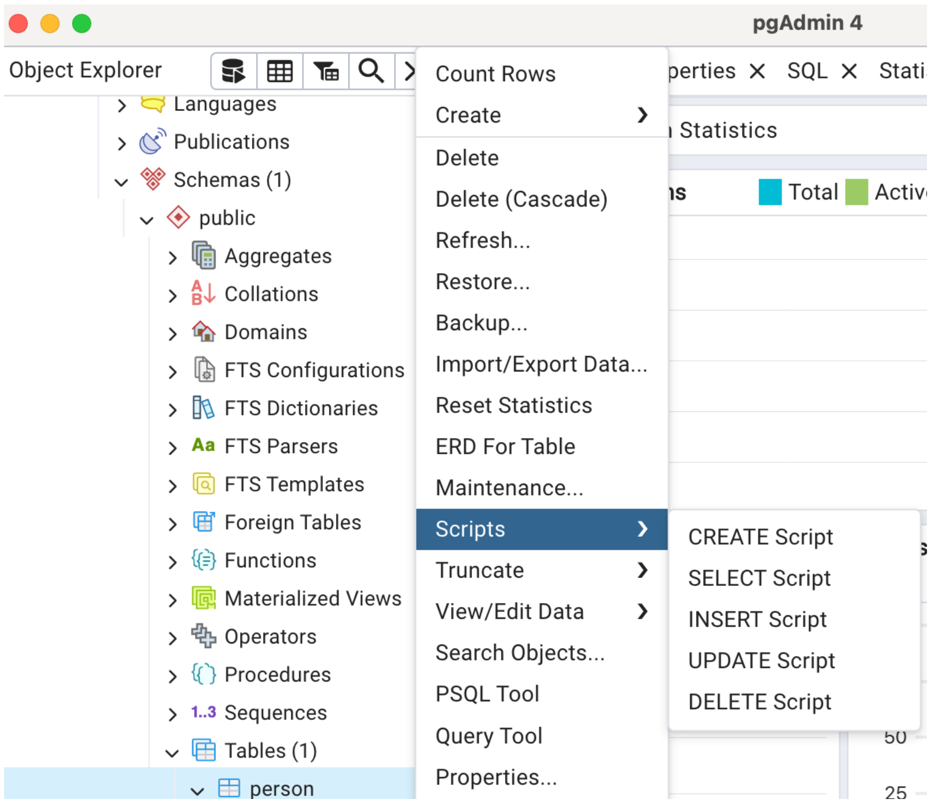
- **Manipulate** data (`INSERT`, `UPDATE`, `DELETE`).
- **Control** access and permissions (`GRANT`, `REVOKE`).

Creating a Table inside PostgreSQL GUI using the below query.

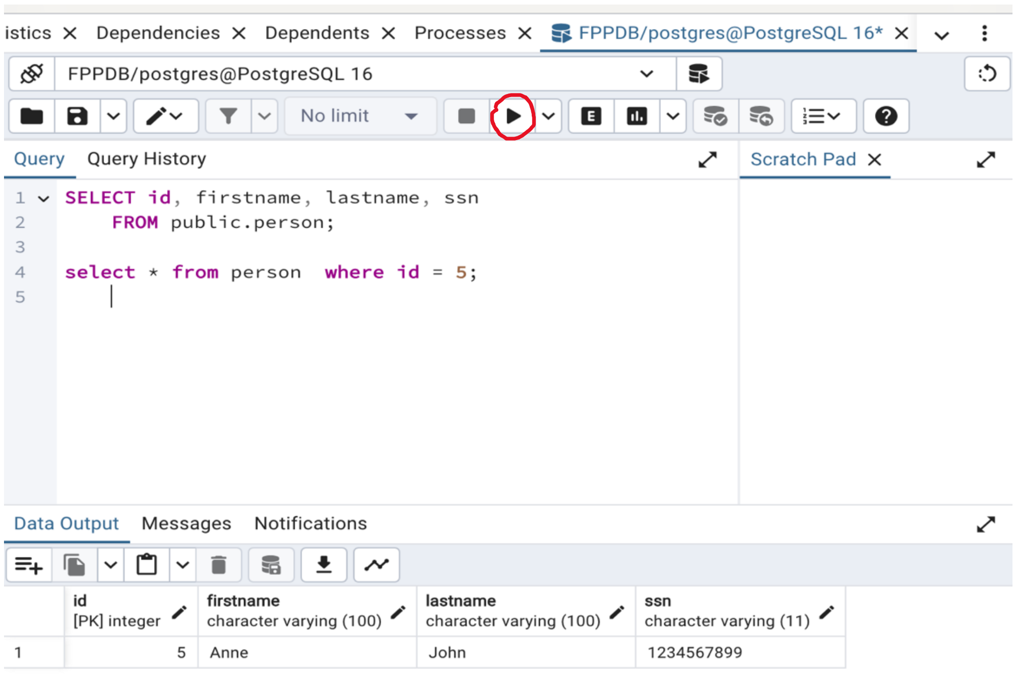
SERIAL for Auto generation ID 1, 2, 3 etc.,
UNIQUE – Does not allow duplicates

```
CREATE TABLE person (
    id SERIAL PRIMARY KEY,
    firstname VARCHAR(100) NOT NULL,
    lastname VARCHAR(100) NOT NULL,
    ssn VARCHAR(10) NOT NULL UNIQUE
);
```

You can manually perform CRUD operation inside GUI, by right clicking the table name.



SELECT Query, Type your queries and select the query, click the Play/Execute button to run the query.



```
SELECT id, firstname, lastname, ssn
	FROM public.person;

select * from person  where id = 5;
```

```
   id            firstname                 lastname                  ssn
   [PK] integer  character varying (100)   character varying (100)   character varying (11)
1             5  Anne                      John                      1234567899
```

Insert Query

```
INSERT INTO public.person(
	 firstname, lastname, ssn)
	VALUES ( 'Joe', 'Bruen', '1122334455');
```

Update Query

```
UPDATE public.person
	SET firstname='Joe', lastname='Lester', ssn='1122334455'
	WHERE id=8;
```

Delete Query

```
DELETE FROM person
	WHERE id=4;
```