

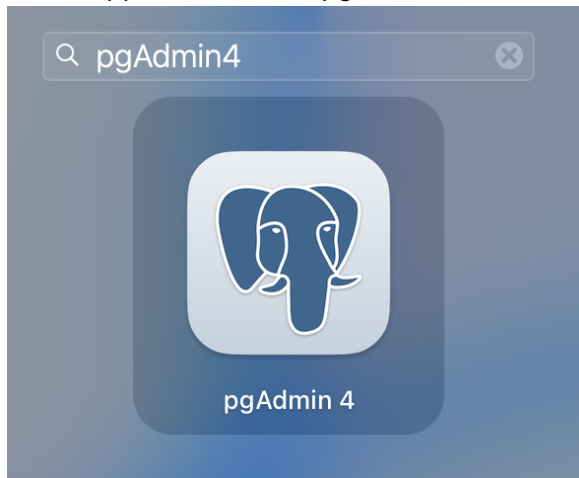
Lesson-13 – DB Setup

Steps to connect PostgreSQL with JDBC API

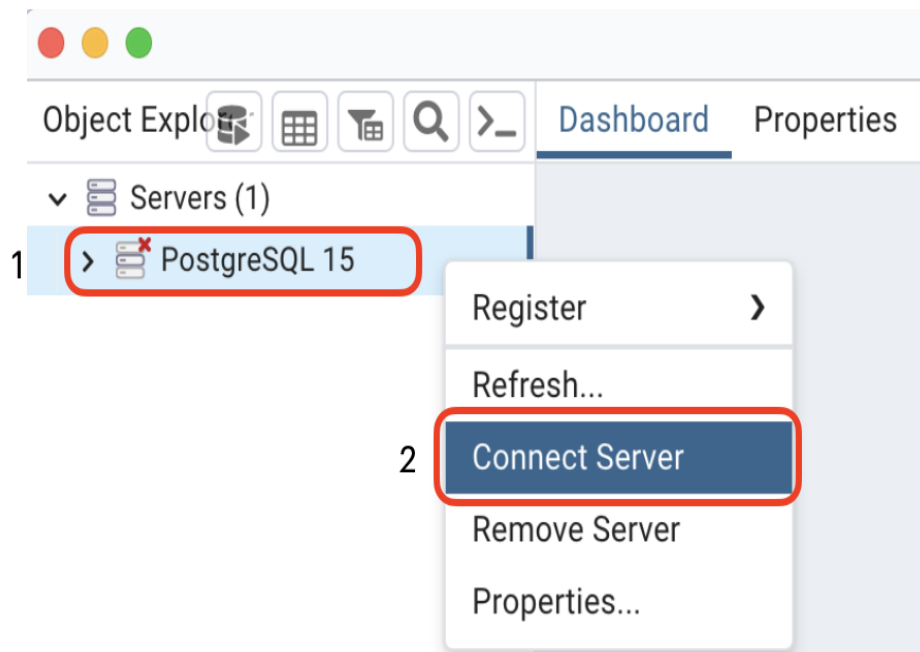
1. Download PostgreSQL based on your OS
<https://www.postgresql.org/download/>

Click on the downloaded file and follow the installation instructions.

2. Go to Application, click “pgAdmin 4”



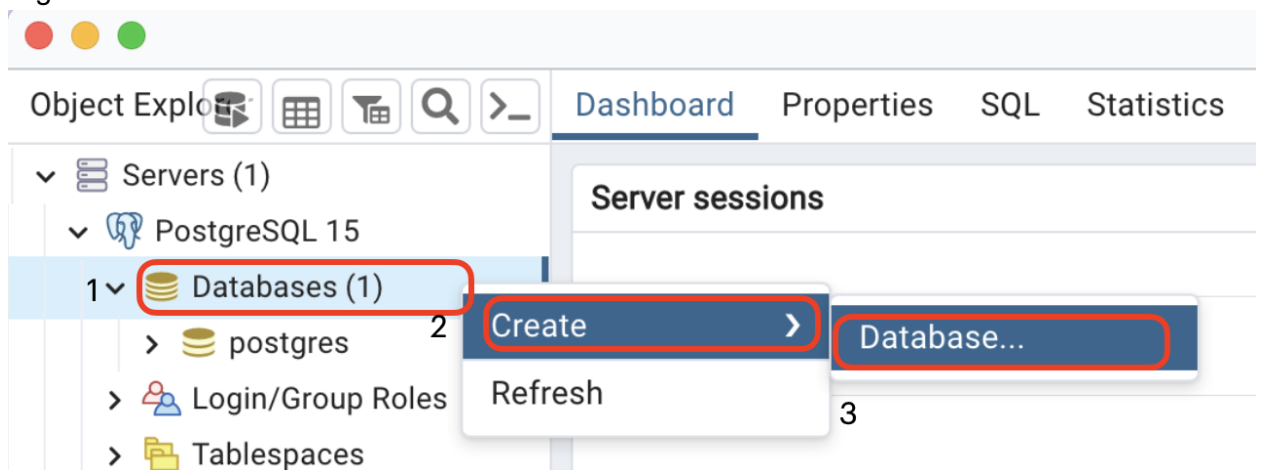
3. Right-click PostgreSQL and click “Connect Server”



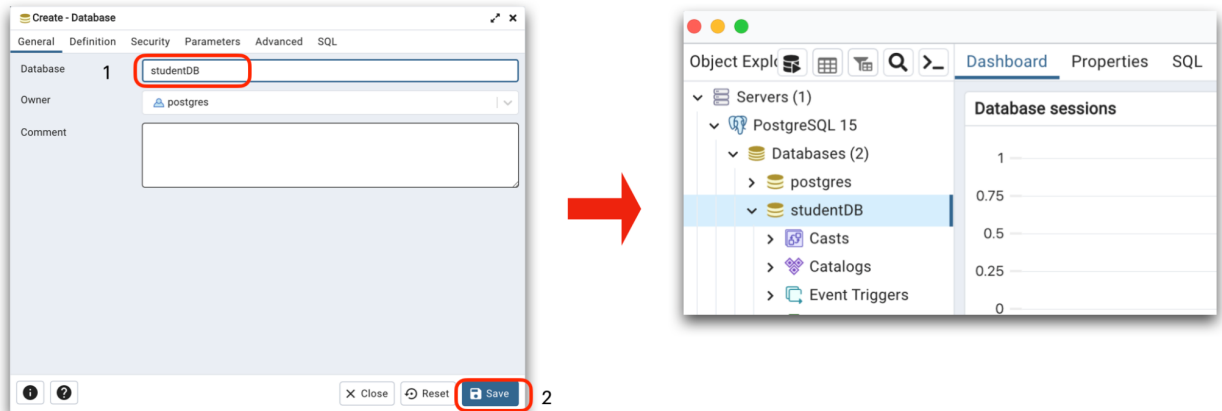
4. Enter the password and click OK.



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.

The screenshot shows the DBeaver interface with the following steps:

1. In the Object Explorer, expand the 'studentDB' database.
2. Expand the 'Schemas (1)' folder under 'studentDB'.
3. Right-click on the 'Tables' folder and select 'Create' > 'Table...'. A red box highlights the 'Table...' option.
4. In the 'Create - Table' dialog, the 'Name' field is set to 'student'.
5. The 'Owner' is set to 'postgres' and the 'Schema' is set to 'public'.
6. Click the 'Columns' tab in the 'Create - Table' dialog.
7. In the 'Columns' tab, click the '+' button to add a new column.
8. In the 'Columns' tab, add two columns: 'registerNumber' (character varying, 10) and 'studentName' (character varying, 100). A red box highlights the column definitions.
9. Click the 'Save' button at the bottom right of the dialog.
10. The 'STUDENT' table is created and visible in the Object Explorer.

8. To view the table

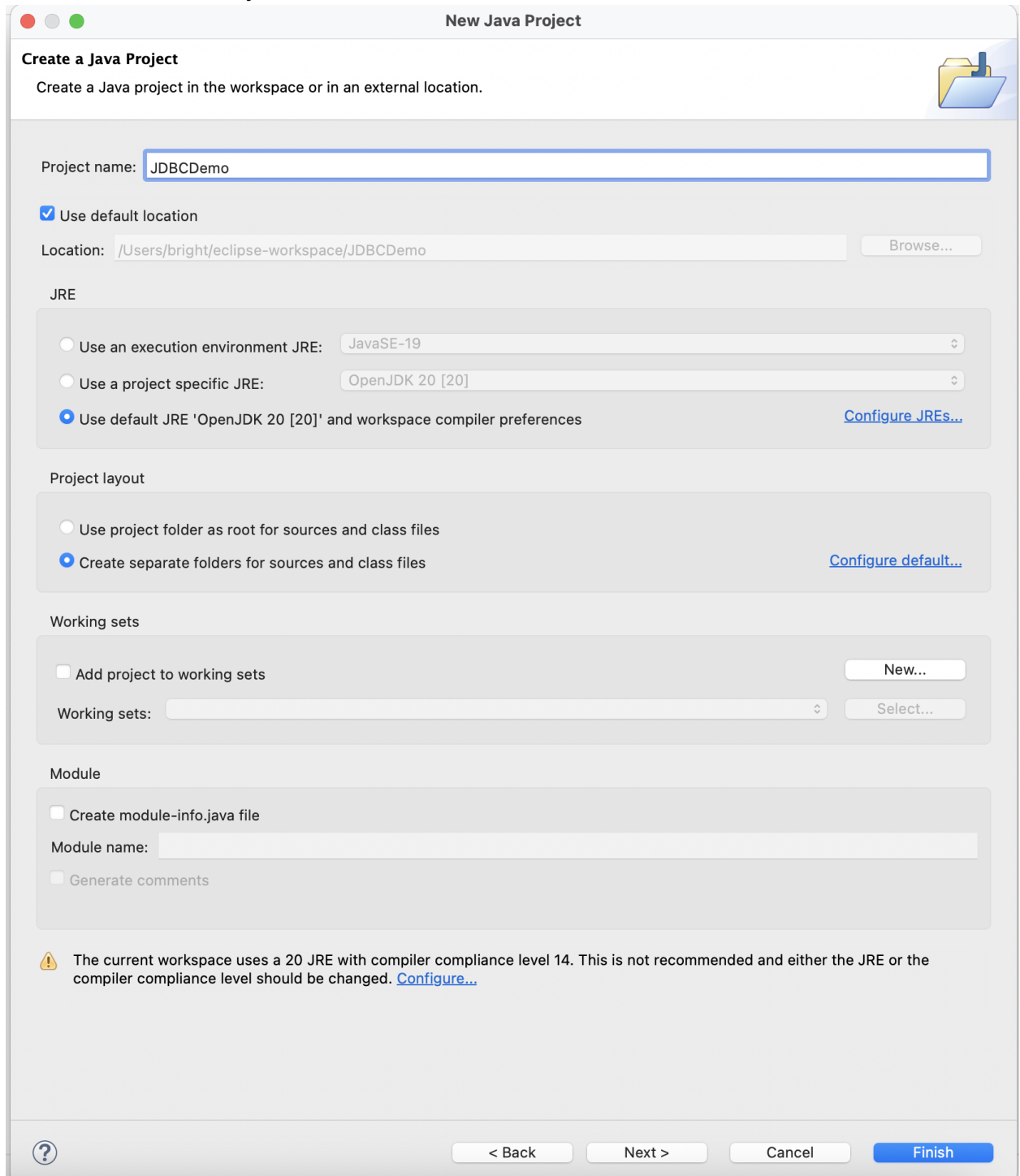
The screenshot shows the DBeaver interface with the following steps:

1. In the Object Explorer, expand the 'studentDB' database and the 'Schemas (1)' folder.
2. Right-click on the 'student' table and select 'View/Edit Data' > 'All Rows'. A red box highlights the 'All Rows' option.
3. The 'Query' tab is selected, showing the SQL query:

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

. The 'Data Output' tab shows the results of the query, displaying the 'registerNumber' and 'studentName' columns.

9. Create a new Java Project.



The screenshot shows the 'New Java Project' dialog box in the Eclipse IDE. The dialog has a title bar with standard macOS window controls (red, yellow, green buttons) and the title 'New Java Project'. Below the title bar, there's a section titled 'Create a Java Project' with a sub-instruction: 'Create a Java project in the workspace or in an external location.' To the right of this text is a folder icon.

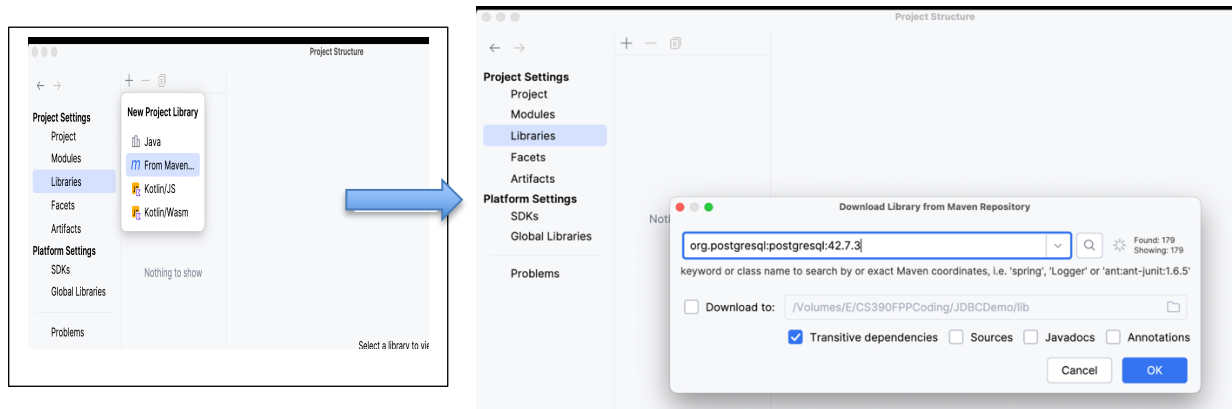
The main content area is divided into several sections:

- Project name:** A text field containing 'JDBCDemo'.
- Location:** A text field showing the default path '/Users/bright/eclipse-workspace/JDBCDemo' and a 'Browse...' button.
- JRE:** A section with three radio buttons:
 - ☐ Use an execution environment JRE: JavaSE-19
 - ☐ Use a project specific JRE: OpenJDK 20 [20]
 - ☒ Use default JRE 'OpenJDK 20 [20]' and workspace compiler preferencesA 'Configure JREs...' link is visible to the right.
- Project layout:** A section with two radio buttons:
 - ☐ Use project folder as root for sources and class files
 - ☒ Create separate folders for sources and class filesA 'Configure default...' link is visible to the right.
- Working sets:** A section with a checkbox 'Add project to working sets' and a 'New...' button. Below it is a 'Working sets:' dropdown menu and a 'Select...' button.
- Module:** A section with a checkbox 'Create module-info.java file', a 'Module name:' text field, and a checkbox 'Generate comments'.

At the bottom of the dialog, there's a warning message: '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...](#)'

The bottom of the dialog features a footer bar with a help icon (question mark) on the left and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish'.

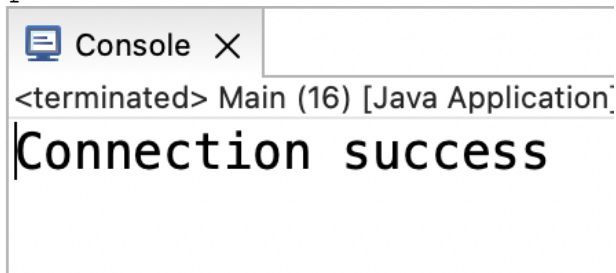
10. Add postgresql jar file into your project follows. Right click project > Open Module Settings > Libraries, Click +, type the driver file needs to add as mentioned. Finally Apply and Click OK.



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

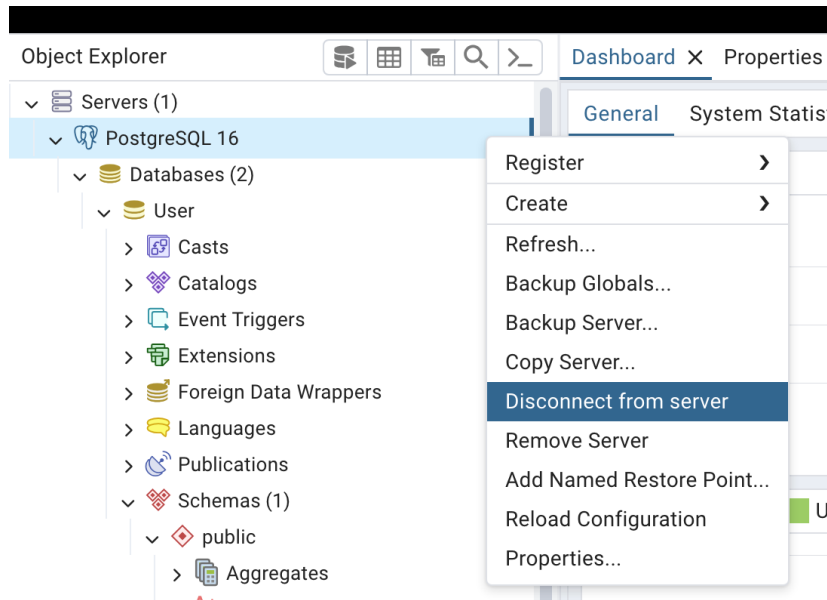
```
public static void main(String[] args) {  
    // User is your Database name  
    String url = "jdbc:postgresql://localhost:5432/User";  
    String uname = "postgres"; // Username - Common  
    String pass = "1234"; // Password you created  
    try {  
        Connection con =  
        DriverManager.getConnection(url,uname,pass);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

Output for the successful connection.



Refer: JDBC Demo 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).

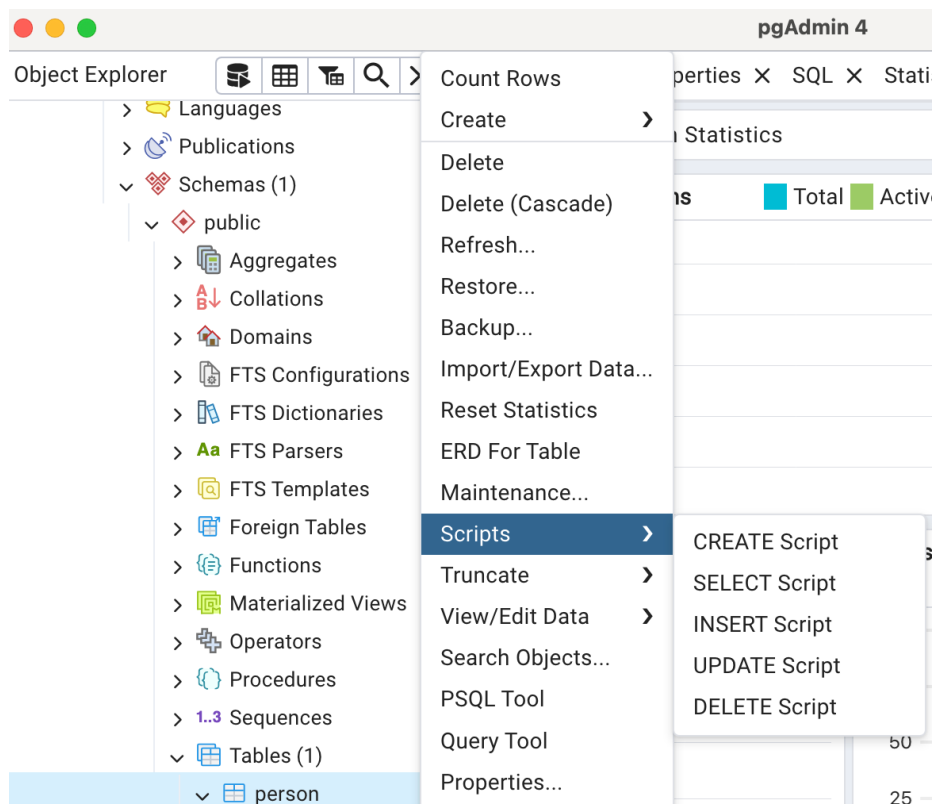
Creating a Table inside PostgreSQL GUI using the below query.

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.

ististics X Dependencies X Dependents X Processes X FPPDB/postgres@PostgreSQL 16* X

FPPDB/postgres@PostgreSQL 16

No limit

Query Query History Scratch Pad X

```
1 SELECT id, firstname, lastname, ssn
2 FROM public.person;
3
4 select * from person where id = 5;
5
```

Data Output Messages Notifications

	id [PK] integer	firstname character varying (100)	lastname character varying (100)	ssn 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;
```
