JDBC (Java DataBase Connectivity)

In this tutorial we guys are going to know how to connect your java application to the database and the Databases may vary according to your organisation so we just need to know how to setup the connection.It's. not that difficult as we think so let's dive deep in this and explore it !

we are going to to use the PostgreSQL in this as a database (**PostgreSQL is an open source database which support both the model(relational and non relational (object oriented), It combines relational database features with object-oriented design elements),Offers a wide range of features for data management, including advanced concurrency control, data integrity, and more.**)

Before we connect to the our java application to the database we need to have the jar files . Now the ? Is what are jar files .....
A JAR file is a package file format typically used to aggregate many Java class files and associated metadata and resources into one file for distribution. JAR files are archive files that include a Java-specific manifest file. They are built on the ZIP format and typically have a .jar file extension. It's a file format based on the popular ZIP file format and is used for **aggregating many files into one.**

this is the most important file we need to connect to db basically we have connectors that we are going to use further but initially let's see what are the steps to connect ....
1. Import the packages that are essential for this
    import java.sql
2. Load the Driver
    this driver is coming from the jar file and every jar file has it's own driver
3. Register the Driver
4.Create Connection
    in this we need to make a connection with the db
5. Create a statement
6.Execute the statement
7.Close the connection

Demo Code for the connection for mySql

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
public class MySQLDemo {

    // DB connection info
    static final String DB_URL = "jdbc:mysql://localhost:3306/testdb"; // Replace
`testdb` with your DB name
    static final String USER = "root";      // Your DB username
    static final String PASS = "password";  // Your DB password

    public static void main(String[] args) {
        Connection conn = null;

        try {
            // Load MySQL JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Open connection
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connected to database!");

            // Create table (if not exists)
            String createTableSQL = "CREATE TABLE IF NOT EXISTS users (id INT
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100))";
            conn.createStatement().execute(createTableSQL);

            // Insert data
            String insertSQL = "INSERT INTO users (name) VALUES (?)";
            PreparedStatement insertStmt = conn.prepareStatement(insertSQL);
            insertStmt.setString(1, "John Doe");
            insertStmt.executeUpdate();
            System.out.println("Data inserted.");

            // Read data
            String selectSQL = "SELECT * FROM users";
            ResultSet rs = conn.createStatement().executeQuery(selectSQL);

            while (rs.next()) {
                System.out.println("User ID: " + rs.getInt("id") + ", Name: " +
rs.getString("name"));
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // Close connection
            try {
                if (conn != null) conn.close();
```

```
        } catch (SQLException se) {
            se.printStackTrace();
        }
    }
  }
}
```

This above code is for the mysql connection to the db

now lets check it out for the PostgreSQL

First of all download the jar file for the PostgreSQL using the following link ....
https://jdbc.postgresql.org/
After downloading the file we need to include this jar file in the project in the external libraries section and after adding the jar file our jar file adding step done

```
// Load PostgreSQL JDBC Driver
Class.forName("org.postgresql.Driver");

//Create Connection
String url = "jdbc:postgresql://localhost:5432/demo";
//url syntax = jdbc:+db_name(mysql, oracle, postgres):+your path or local address/+your db_name in post
String user = "postgres";
String password = “postgres”;

Connection con = DriverManager.getConnection(url,user,password);
```

upto this what we have done is that we have completed our first 4 steps and in the above code I have also mentioned that the how to make the ur for making the connection now lets talk about the other 3 steps

```
Statement stmt = con.createStatement(); //Create a statement
stmt.executeQuery(query); // execute the query
```

after this we need to keep one thing in our mind is that we have to call the execute the queries on the basis of the operation that we guys are performing if we are not not updating anything then we need to use the execute query and if we are using to update something then we have ti use the executeQuery() for that so we need to keep that in mind while using this
and the execute query will return something like resultSet(it’s an interface)

```
ResultSet rs = stmt.executeQuery(query); // execute the query

while(rs.next()){
    System.out.println(rs.getString("sname"));
}
```

this will return the resultSet and we have to store and need to print the data using the column name in that the next function will return the boolean values if we have the next value in the db yes or not if yes it will execute the query behind the scenes and get the data for us
and using the resultSet we can print the data inside the db and it contains multiple function which help us to print the data
con.close(); // close the connection

Now upto this we have done with R part of the CRUD operations and now we have to do for the create part and the remaining one
so for that we have to create a insert Query and now the statement execution part will change as we discussed earlier

now we will use execute() function and this will return a count for the inset and the update query let's see this how this will behave in the code

Now as we are done the basic step or done with CRUD operations now lets discuss the difficulties of the statement
what if the user enter the data manually then how we have to pass the variable in the query this is the Main problem ""??

No Worries !!!!!
We will use this format to inset the variables in the query **"+ variable_name+"** ===. For the string values but for the string values we will use the single quotes before this **'"+variable_name+"'**

String query = "insert into student values ('"+sname+"',"+sid+",'"+dept+"')";

This is how the query will look like
But but but the story does not ends here we have issues with this query because this leads to memory leakage and leads to the unsafe ways in the database so we need to prepare a statement regarding these how can we do it let's check this one "

now we don't know the data so we can add the ? inplace of the parameter in the

query and then we can use the preparedStatement in place of the statement and then using the inbuilt methods that preparedStatement provide us like for the int type data we use setInt(), setString() and much more so how the code will look like ……..

A Statement in Java is used for executing SQL queries directly, which can be vulnerable to SQL injection if not handled carefully. On the other hand, a PreparedStatement is a precompiled SQL statement with placeholders, offering better security and performance as it can be reused with different parameter values, preventing SQL injection attacks.

A Statement in Java is used for executing SQL queries directly, which can be vulnerable to SQL injection if not handled carefully. On the other hand, a PreparedStatement is a precompiled SQL statement with placeholders, offering better security and performance as it can be reused with different parameter values, preventing SQL injection attacks.

String query = "insert into student values (?,?,?)"; //using query for preparedStatement

```
 PreparedStatement stmt = con.prepareStatement(query);
stmt.setString(1,sname);
stmt.setInt(2,81223);
stmt.setString(3,dept);

boolean check = stmt.execute();//For the Prepared Statement
System.out.println(check);
```