Day 4

In the previous session, we had seen how we can take input from the user to complete the query and for that, we make use of the PreparedStatement interface.
But what if I wanted to take user input multiple times and have to insert the query into the database multiple times?
For that, we can make use of a loop as shown below-

```
package jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class Demoj {

	public static void main(String[] args)  {
		String url = "jdbc:mysql://localhost:3306/employee";
		String un = "root";
		String pwd = "root";
		PreparedStatement pstmt = null;
		ResultSet res = null;
		Connection con = null;

		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			System.out.println("Driver successfully loaded");

			con=DriverManager.getConnection(url, un, pwd);
			System.out.println("Connection established");
			String query ="insert into
emp(`id`,`name`,`desig`,`salary`) values (?,?,?,?)";

			pstmt = con.prepareStatement(query);

			Scanner scan =new Scanner(System.in);
			int n = scan.nextInt();
```

```
                for(int i = 1;i <= n;i++)
                {
                      int id = scan.nextInt();
                      String name = scan.next();
                      String desig = scan.next();
                      int salary = scan.nextInt();
                      pstmt.setInt(1, id);
                      pstmt.setString(2, name);
                      pstmt.setString(3, desig);
                      pstmt.setInt(4, salary);
                      pstmt.execute();
                }

          } catch (ClassNotFoundException  e) {
                e.printStackTrace();
          }
          catch(SQLException e) {
                e.printStackTrace();
          }


          try {
                pstmt.close();
                con.close();
          } catch (SQLException e) {
                e.printStackTrace();
          }

    }

}
```

When we execute the above program and we enter the following data-

```
Demoj [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe  (Jul 13, 2021, 11:12:43 AM)
Driver successfully loaded
Connection established
3
4
Arun
editor
25000
```

One row would have been inserted into our database-

	id	name	dsig	salary
▶	1	Rohit	CEO	10000
	2	Somanna	CTO	12000
	3	Rakshit	CMO	15000
	4	Arun	editor	25000
*	NULL	NULL	NULL	NULL

Since we are in a loop, the editor is waiting for the next input and we can keep inserting the data, till the loop is terminated.
But what if there was a problem due to which few rows were not inserted? This could result in an incomplete database and this could lead to a problem in the future.
We have to follow the property in DBMS, which is known as the ACID(Atomicity, Consistency, Isolation, Durability) and every database must possess this property.
So according to this property, we should have inserted 3 rows into our database. But due to some issue if all 3 rows could not be inserted, then None of the rows should be inserted.
We have to follow the All or None rule. I.e., either all the rows should be inserted or none of the rows should be inserted.
So how can we achieve this?
Even if we insert a row into the table, until and unless all rows are inserted, data should not be committed into the database. To achieve this, we can make use of a method known as **setAutoCommit()**
This method takes a boolean value and by default it's true. So by setting its value as false, we can make sure that the data will not be committed into the database, until we call commit() it will not be committed into the database.

```
pstmt = con.prepareStatement(query);

Scanner scan =new Scanner(System.in);
int n = scan.nextInt();
con.setAutoCommit(false);
```

```
for(int i = 1;i <= n;i++)
{
	int id = scan.nextInt();
	String name = scan.next();
	String desig = scan.next();
	int salary = scan.nextInt();
	pstmt.setInt(1, id);
	pstmt.setString(2, name);
	pstmt.setString(3, desig);
	pstmt.setInt(4, salary);
	pstmt.execute();
}

con.commit();
```