From JDBC4.X version onwards,there is a facility of "autoloading".

Q> What is autoloading in JDBC?
      Loading and register the driver is done automatically,based on the url supplied by the user.
      Behind the scenes
            a. check the url
            b. based on the url supplied, go to classpath environmental variable
            c. open the relevant jar
            d. go to META-INF/services folder
            e. open java.sql.Driver file
            f. read the file and load the class supplied in the file

Note:
    Using resultSet object, we can retrieve the records based on the column names also.
    If java pgm and database engine is running in the same program with the default port no for database then
    url can be of the following type
            String url = "jdbc:mysql:///octbatch".

Program to demonstrate select operation using JDBC
==========
```
package in.ineuron.main;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

//JDBC4.X autoloading feature is enabled.
public class SelectApp {

      public static void main(String[] args) throws ClassNotFoundException, SQLException {

            // Step2. Establish the Connection
            String url = "jdbc:mysql:///octbatch";
            String user = "root";
            String password = "root123";
            Connection connection = DriverManager.getConnection(url, user, password);
            System.out.println("CONNECTION object created...");

            // Step3. Create statement Object and send the Query
            Statement statement = connection.createStatement();
            System.out.println("STATEMENT object created...");

            // Step4. Execute the Query and Process the resultSet
            String sqlSelectQuery = "select sid,sname,sage,saddress from student";
            ResultSet resultSet = statement.executeQuery(sqlSelectQuery);
            System.out.println("RESULTSET object created...");
            System.out.println("SID\tSNAME\tSAGE\tSADDRESS");
            while (resultSet.next()) {

                  int sid = resultSet.getInt("sid");
                  String sname = resultSet.getString("sname");
```

```
                int sage = resultSet.getInt("sage");
                String saddress = resultSet.getString("saddress");
                System.out.println(sid + "\t" + sname + "\t" + sage + "\t" +
saddress);
            }

            // Step6. Close the resources
            resultSet.close();
            statement.close();
            connection.close();
            System.out.println("Closing the resources...");

      }
}
Output
Loading the driver...
CONNECTION object created...
STATEMENT object created...
RESULTSET object created...
SID   SNAME SAGE  SADDRESS
1     sachin            50                MI
2     kohli       35                RCB
3     dhoni       41                CSK
4     rahul       28                LSG
5     SKY               28                DC
Closing the resources...
```

Note
According to DBA specification, all SQL commands are categorised into following types

a. DDL(Data Definition Language)
   eg: Create table, alter table, drop table, etc...
b. DML(Data Manipulation Language)
   eg: insert,update,delete
c. DQL(Data Query Language)
   eg: Select
d. DCL(Data Control Language)
   eg: Alter password, grant access,....
e. DA command(Database Administrator commands)
   eg: start audit
       stop audit
f. TCL(Transaction Control Language)
   eg: commit, rollback,savepoint,....

According to Java Developer point of view, all SQL operations are classified into 2 types

a. Select operation(DQL)
b. Non-Select Operation(DML,DDL,...)

Through Statement Object we need to execute the Query and to exectue the Query we need to make a call
to a method as shown below.

a. executeQuery()
b. executeUpdate()
c. execute()

a. executeQuery()
   This method is used only if we perform select operation.

```
      Because of  this method execution, we will get a group of records which are
represented as "ResutlSet" object.
            public ResultSet executeQuery(String sqlSelectQuery) throws
SQLException;
                  eg: ResultSet resultSet =statement.executeQuery("select
sid,sname,sage,saddress from student");

b. executeUpdate()
      This method is used for "Non-Select Operations" like(Insert|Update|Delete)
      Because of this method execution, we won't get group of records, we will get
a numeric value which represents
      the number of rows affected. So return type of the method is "int".
                  public int executeUpdate(String sqlNonSelectQuery) throws
SQLException;
                  eg: int rowAffected = statement.executeUpdate("delete from
student where sid = 10");
                        System.out.println("No of rows affected is ::
"+rowAffected);

c. execute()
            we can use this method for both select and nonselect operation
            if we don't know the type of query at the begining and if is available
dynamically at the runtime then we need to use
            this method for execution.
                  public boolean execute(String sql) throws SQLException;
                  eg: boolean value = statement.execute(dynamicQuery);
                        if(value == true)
                        {
                                    //select Query
                                    ResultSet resultSet = statement.getResultSet();

                                    //process the resultSet
                        }
                        else
                        {
                                    //nonSelect Query
                                    int rowCount = statement.getUpdateCount();
                                    System.out.println("Number of rows affected
is :: "+rowCount);
                        }

Formatting SQlQueries using dynamic input
==================================

1st approach
==========
sname = scanner.next();
sage =scanner.nextInt();
saddress = scanner.next();
String sqlInsertQuery = "insert into
student(`sname`,`sage`,`saddress`)values('"+sname+"',"+sage+",'"+saddress+"')";

2nd approach
===========
sname = scanner.next();
sage =scanner.nextInt();
saddress = scanner.next();
sname = "'"+sname+"'";
saddress = "'"+saddress+"'";
```

```
String sqlInsertQuery = "insert into
student(`sname`,`sage`,`saddress`)values("+sname+","+sage+","+saddress+")";

3rd approach
===========
      The above 2 approaches are not recomended, to do formatting we prefer using
String class format() as shown below.
            public static  String format(String format, Object... args)

sname = scanner.next();
sage =scanner.nextInt();
saddress = scanner.next();
String sqlInsertQuery =String.format("insert into
student(`sname`,`sage`,`saddress`) values ('%s',%d,'%s')",
                                                              sname,sage,saddress);

Write a  JDBC code to take dynamic input from the user to perform insert operation
package in.ineuron.main;

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

//JDBC4.X autoloading feature is enabled.
public class InsertApp {

      public static void main(String[] args) throws SQLException {

            // Step2. Establish the Connection
            String url = "jdbc:mysql:///octbatch";
            String user = "root";
            String password = "root123";
            Connection connection = DriverManager.getConnection(url, user,
password);
            System.out.println("connection object created...");

            // Step3. Create statement Object and send the Query
            Statement statement = connection.createStatement();
            System.out.println("statement object created...");

            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter the name of the student :: ");
            String sname = scanner.next();

            System.out.print("Enter the age of the student :: ");
            int sage = scanner.nextInt();

            System.out.print("Enter the address of the student :: ");
            String address = scanner.next();

            System.out.print("Enter the gender of a student:: ");
            String gender = scanner.next();

            // Step4. Execute the Query and Process the resultSet
            String sqlInsertQuery = String.format("insert into
student(`sname`,`sage`,`saddress`,`sgender`) values ('%s',%d,'%s','%s')",
```

```
                  sname, sage, address,gender);

            System.out.println(sqlInsertQuery);

            int rowAffected = statement.executeUpdate(sqlInsertQuery);
            System.out.println("No of rows affected is :: " + rowAffected);

            // Step6. Close the resources
            statement.close();
            connection.close();
            scanner.close();
            System.out.println("closing the resources...");

      }
}
Output
connection object created...
statement object created...
Enter the name of the student :: mandana
Enter the age of the student :: 27
Enter the address of the student :: kodagu
Enter the gender of a student:: F
insert into student(`sname`,`sage`,`saddress`,`sgender`) values
('mandana',27,'kodagu','F')
No of rows affected is :: 1
closing the resources...


Note:
 While writing JDBC code, the following steps are common
            a. Establishing the connection
            b. closing the resources
            c. handling the exception
 What would vary in every code is
            a. query will be different
                  1. if it select process the ResultSet object.
                  2. if it non-select, process the integer value.

For the above 3 steps, the common util code is as shown below
package in.ineuron.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class JdbcUtil {

      private JdbcUtil() {
      }

      static {
            //Step1: loading and register the Driver
            try {
                  Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException ce) {
                  ce.printStackTrace();
```

```
			}
		}

		public static Connection getJdbcConnection() throws SQLException {
			// Step2. Establish the Connection
			String url = "jdbc:mysql:///octbatch";
			String user = "root";
			String password = "root123";
			Connection connection = DriverManager.getConnection(url, user,
password);
			System.out.println("connection object created...");
			return connection;
		}
		public static void cleanUp(Connection con, Statement statement, ResultSet
resultSet) throws SQLException {
			// Step6. Close the resources
			if (con != null) {
				con.close();
			}
			if (statement != null) {
				statement.close();
			}
			if (resultSet != null) {
				resultSet.close();
			}

		}
}
```

Note:
  In the above code, the url pattern, username and password is hardcode.
  These values would vary from user to user.
  To set these values, we use properties file approach as shown below.

```
package in.ineuron.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class JdbcUtil {

		private JdbcUtil() {
		}

		static {
			// Step1: loading and register the Driver
			try {
				Class.forName("com.mysql.cj.jdbc.Driver");
			} catch (ClassNotFoundException ce) {
				ce.printStackTrace();
			}
		}
```

```
	public static Connection getJdbcConnection() throws SQLException, IOException
{

		// Take the data from properties file
		FileInputStream fis = new FileInputStream("D:\\octbatchjdbcpgm\\
JDBCStandardApp\\application.properties");
		Properties properties = new Properties();
		properties.load(fis);

		// Step2. Establish the Connection
		Connection connection =
DriverManager.getConnection(properties.getProperty("url"),
				properties.getProperty("username"),
properties.getProperty("password"));
		System.out.println("connection object created...");
		return connection;
	}

	public static void cleanUp(Connection con, Statement statement, ResultSet
resultSet) throws SQLException {
		// Step6. Close the resources
		if (con != null) {
			con.close();
		}

		if (statement != null) {
			statement.close();
		}
		if (resultSet != null) {
			resultSet.close();
		}

	}
}
application.properties
==================
url=jdbc:mysql:///octbatch
username=root
password=root123
```