Fundamental Steps in JDBC in the process of connecting to a database and executing a query consist of the following:

1. **Import JDBC packages.**

This is for making the JDBC API classes immediately available to the application program.

->> import java.sql.\*;

1. **Load and register the JDBC driver.**

This is for loading and registering the JDBC drivers with Oracle database. This is done by using the static registerDriver() method of the DriverManager class of the JDBC API. For the entire Java application, the JDBC driver is registered only once per each database that needs to be accessed. This is true even when there are multiple database connections to the same data server.

->> DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

1. **Open a connection to the database.**

Once the required packages are imported and the Oracle JDBC driver is loaded and registered, a database connection must be established. This is done by using the getConnection() method of the DriverManager class. A call to this method creates an object instance of the java.sql.Connection class. The getConnection() requires three input parameters- URL, username and a password, all 3 are String.

The getConnection() method is an overloaded method that takes-

1. Three parameters, one each for the URL, username, and password.

->>Connection conn = DriverManager.getConnection(URL, username, passwd);

1. Only one parameter for the database URL. In this case, the URL contains the username and password.

->>Connection conn = DriverManager.getConnection(URL);

1. **Create a statement object to perform a query.**

In this step we try to get a Statement object which is what will allow us to run our SQL queries. A Statement object can be obtained by createStatement() method of the Connection object created above. A call to this method creates an object instance of the Statement class.

->> Statement sql\_stmt = conn.createStatement();

1. **Execute the SQL query with help of Statement object and return output as ResultSet object.**

Once a Statement object is created, next we execute our SQL query. This is done by using the executeQuery() method of the Statement object. A call to this method takes as parameter a SQL **SELECT** statement and returns a ResultSet object.

->> ResultSet result = stmt.executeQuery("SELECT empno, … FROM emp …");

OR

->> String sql = "SELECT empno, … FROM emp …";

ResultSet result = stmt.executeQuery(sql);

1. **Process the ResultSet.**

Once the query has been executed, there are two steps to be carried out:

1. Processing the output resultset to fetch the rows
2. Retrieving the column values of the current row

The first step is done using the next() method of the ResultSet object. A call to next() is executed in a loop to fetch the rows one row at a time, with each call to next() advancing the control to the next available row. The next() method returns the Boolean value true while rows are still available for fetching and returns false when all the rows have been fetched.

The second step is done by using the getXXX() methods of the ResultSet object. Here getXXX() corresponds to the getInt(), getString() etc with XXX being replaced by a Java datatype.

String str;

**while** (result.next()){

str = result.getInt(1)+ " "+ result.getString(2)+ "

"+result.getFloat(3)+ " "result.getInt(4)+ "\n";

}

**byte** buf[] = str.getBytes();

OutputStream fp = **new** FileOutputStream("query1.lst");

fp.write(buf);

fp.close();

1. **Close the ResultSet and Statement objects.**

Once the ResultSet and Statement objects have been used, they must be closed explicitly. This is done by calls to the close() method of the ResultSet and Statement classes

->> rset.close();

sql\_stmt.close();

1. **Close the connection.**

The last step is to close the database connection opened in the beginning after importing the packages and loading the JDBC drivers. This is done by a call to the close() method of the Connection class. Closing the ResultSet and Statement objects does not close the connection. It should be closed by explicitly invoking the close() method of the Connection class.

->> conn.close();

**Processing the Results of a Database Query That Returns a Single Row**

The above sections show the processing of a query that returned multiple rows. This section highlights the processing of a single-row query.

NO\_DATA\_FOUND exception in PL/SQL is simulated in JDBC by using the return value of the next() method of the ResultSet object. A value of false returned by the next() method identifies a NO\_DATA\_FOUND exception.

(result is obtained from above query)

if (result.next()) // Process the row returned

else System.out.println("Does not exist");