

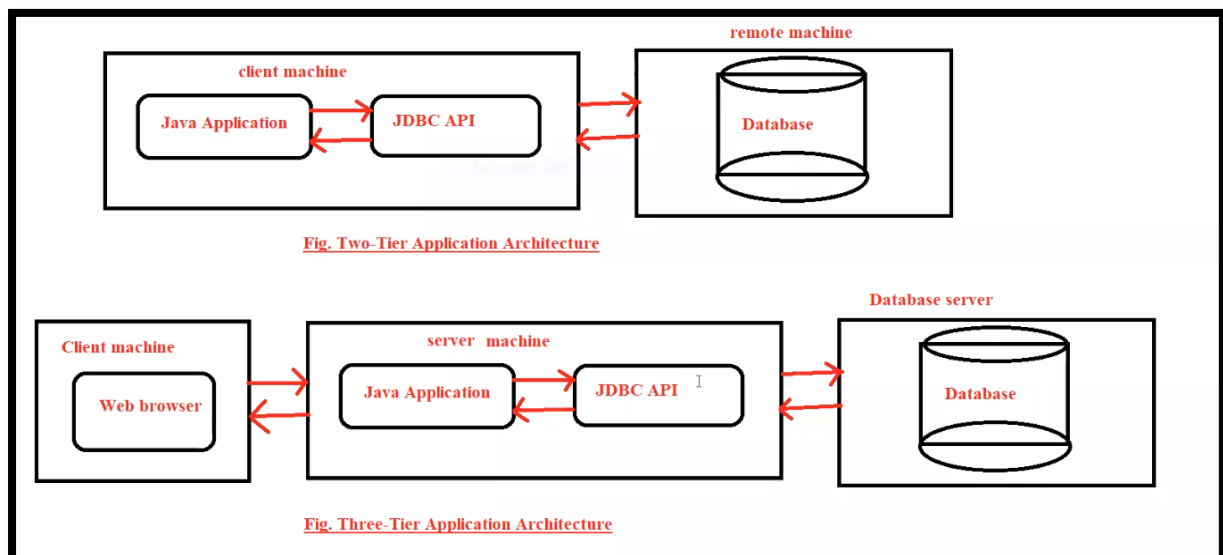
### \*\*\*\*\* Interacting with Database \*\*\*\*\*

✓ Interview Question:-How to Establish Connection Between java and Database.

- **JDBC Architecture:-**

- 1) Two-Tier Application Architecture.
- 2) Three-Tier Application Architecture.

- **Diagram:-**



- **JDBC Driver:-**

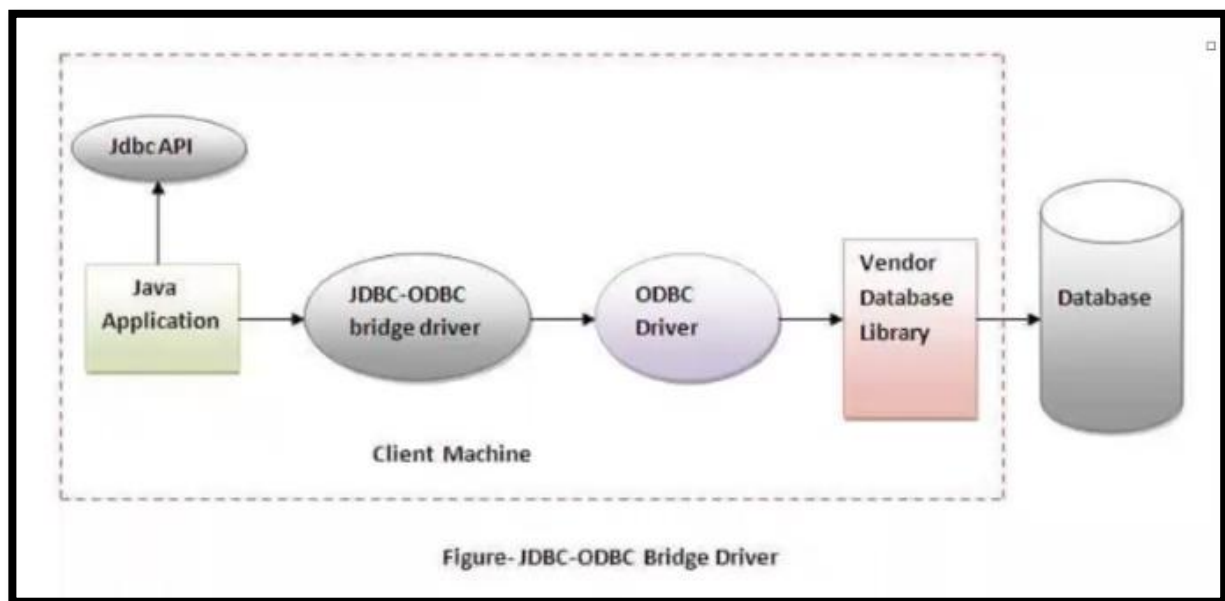
JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver).

### 1) JDBC-ODBC bridge driver:-

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

**Diagram:-**



Oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

### Advantages:

- 1)easy to use.
- 2)can be easily connected to any database.

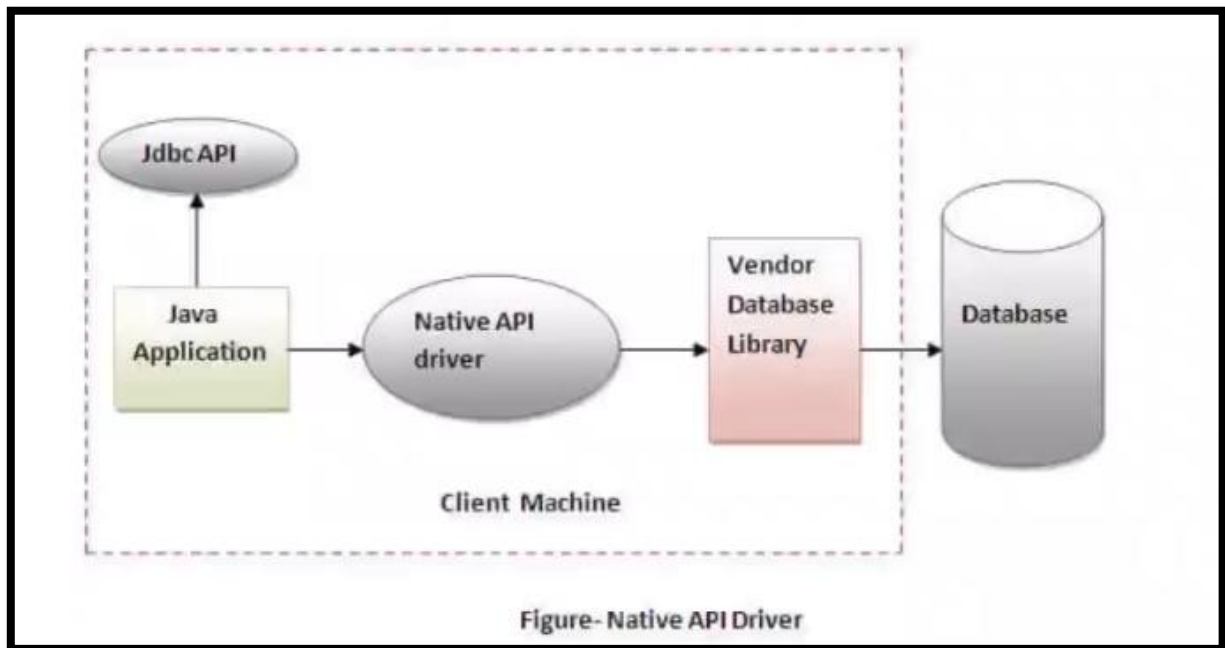
### Disadvantages:

- 1)Performance degraded because JDBC method call is converted into the ODBC function calls.
- 2)The ODBC driver needs to be installed on the client machine.

### 2) Native-API driver:-

The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

**Diagram:-**



### Advantage:

- 1) performance upgraded than JDBC-ODBC bridge driver.

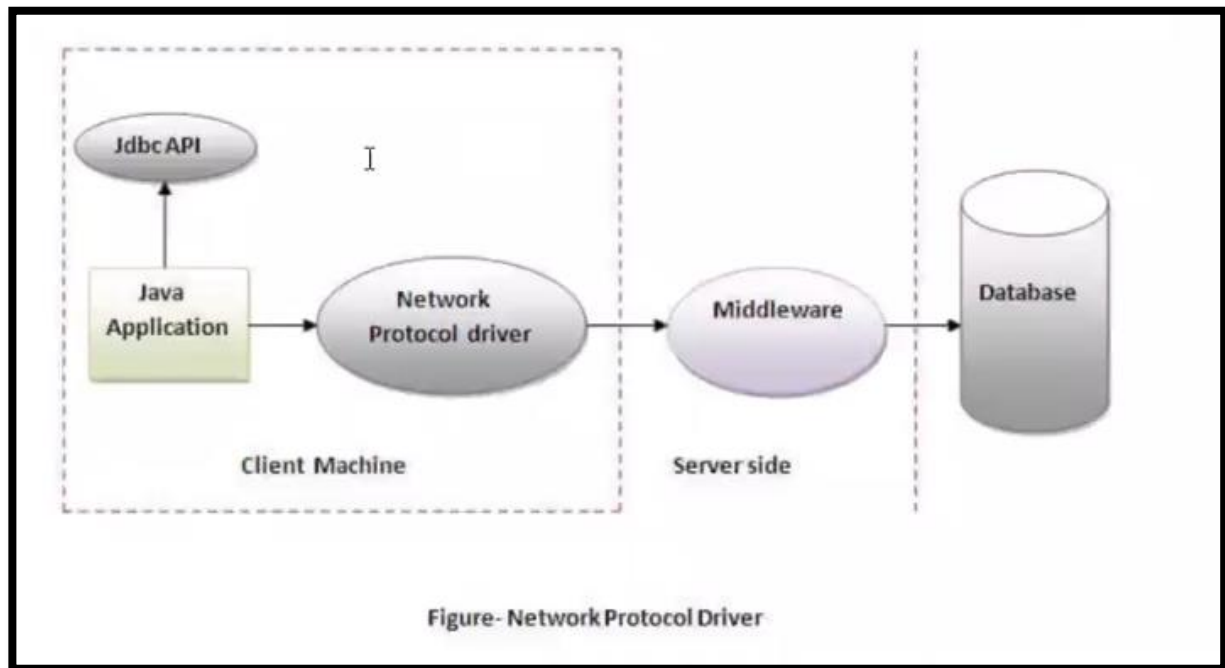
### Disadvantage:

- 1) The Native driver needs to be installed on the each client machine.
- 2) The Vendor client library needs to be installed on client machine.

**3) Network Protocol driver:-**

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

**Diagram:-**

**Advantage:**

- 1) No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

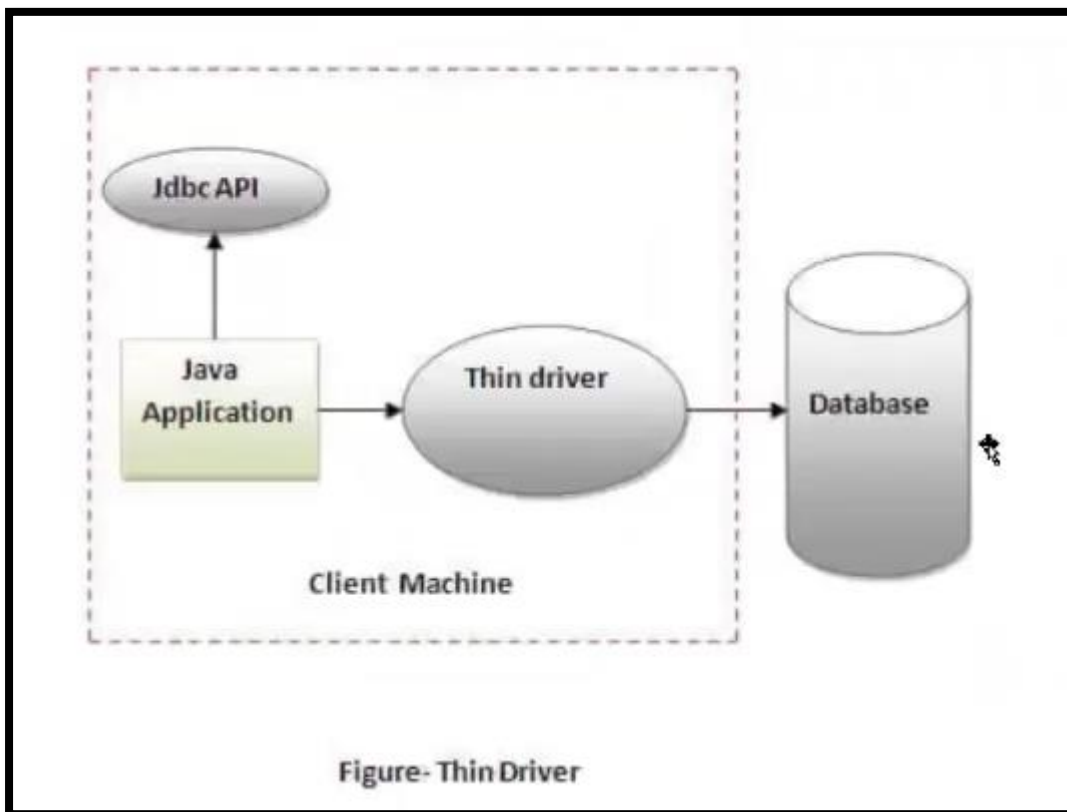
**Disadvantages:**

- 1) Network is required on client
- 2) Requires database-specific coding to be done in the middle tier.
- 3) Maintenance of Network Protocol driver becomes costly because it requires database specific coding to be done in the middle tier.

### 4) Thin driver:-

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

**Diagram:-**



### Advantage:

- 1) Better performance than all other drivers.
- 2) No software is required at client side or server side.

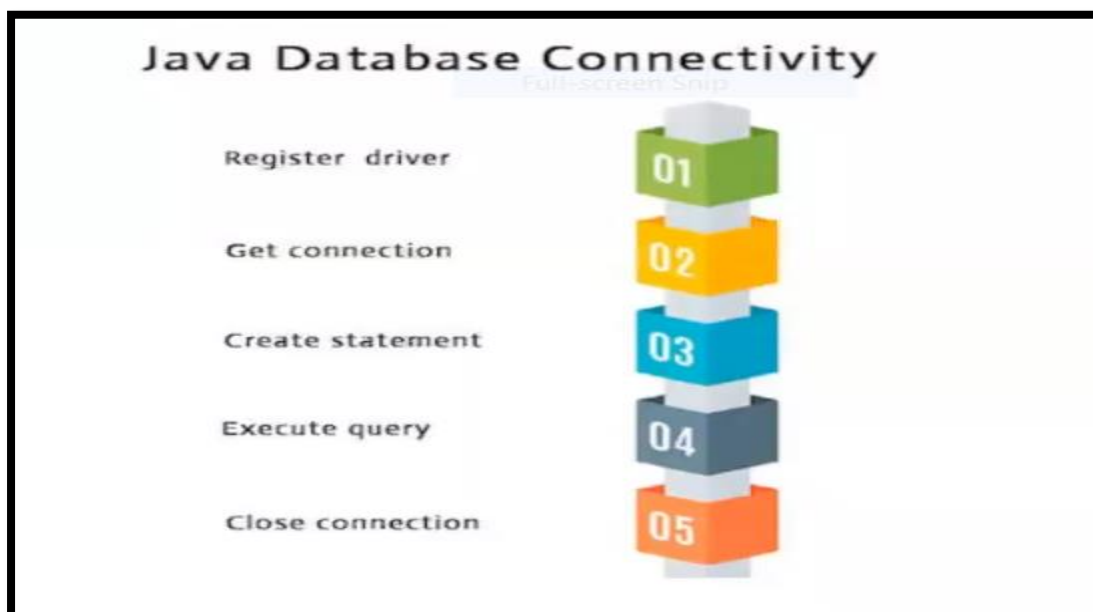
### Disadvantage:

Drivers depend on the Database.

### Java Database Connectivity with 5 Steps:-

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

- 1) Register the Driver class
- 2) Create connection
- 3) Create statement
- 4) Execute queries
- 5) Close connection



#### 1) Register the driver class:-

The **forName()** method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

#### Syntax of forName() method

```
public static void forName(String className) throws ClassNotFoundException
```

#### Example to register the OracleDriver class

Here, Java program is loading oracle driver to establish database connection.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
*****
```

### 2) Create the connection object:-

The getConnection() method of DriverManager class is used to establish connection with the database.

#### Syntax of getConnection() method

- 1) **public static** Connection getConnection(String url) throws SQLException
- 2) **public static** Connection getConnection(String url, String name, String password) throws SQLException

#### Example to establish connection with the Oracle database

```
Connection con=DriverManager.getConnection("jdbc:oracle:thin:
```

```
@localhost:1521:xe","sys t em","password");
```

```
*****
```

### 3) Create the Statement object:-

The createStatement() method of Connection interface is used to create statement.

The object of statement is responsible to execute queries with the database.

#### Syntax of createStatement() method

```
public Statement createStatement() throws SQLException
```

#### Example to create the statement object

```
Statement stmt=con.createStatement();
```

```
*****
```

**4) Execute the query:-**

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

**Syntax of executeQuery() method**

**public** ResultSet executeQuery(String sql) **throws** SQLException

**Example to execute query**

```
ResultSet rs=stmt.executeQuery("select * from emp");  
while(rs.next())  
{  
    System.out.println(rs.getInt(1)+" "+rs.getString(2));  
}
```

.....

**5) Close the connection object:-**

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

**Syntax of close() method**

**public void** close() **throws** SQLException

**Example of Close Connection**

```
Con.close();
```



- **Code:-**

```
import java.sql.*;
class JDBCdemo
{
    public static void main(String args[])throws SQLException
    {
        try
        {
            //Register Driver
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Example to establish connection with the Oracle database
            Connection con=DriverManager.getConnection("jdbc:oracle:thin:
            @localhost:1521:xe","system","password");
            //Create Statement Object
            Statement stmt=con.createStatement();
            ResultSet rs=stmt.executeQuery("select *from Student");
            while(rs.next())
            {
                System.out.println(re.getInt(1)+" "+rs.getString(2));
            }
            //Close The Connection
            con.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

\*\*\*\*\*

- **\*\*\*Class:-**

- > static Class forname(String className)

- **\*\*\*DriverManager class:-**

- > static Connection getConnection(String URL)

- > static Connection getConnection(String URL,String username, String password)

- > static void deregisterDriver(Driver driver)

- **\*\*\*Connection interface:**

- > Statement createStatement();

- > void close();

- > void commit();

- > CallableStatement prepareCall(String sql);

- > PreparedStatement prepareStatement(String sql)

- **\*\*\*Statement Interface:-**

- > void close();

- > void cancel();

- > boolean execute(String sql)

- > ResultSet executeQuery(String sql) //select query

- > int executeUpdate(String sql) //insert,delete,update

- **\*\*\*PreparedStatement Interface:**

- > boolean execute()
- > ResultSet executeQuery() //select query
- > int executeUpdate() //insert,delete,update
- > void setInt(int index, int val);
- > void setFloat(int index, float val);
- > void setDouble(int index, double val);
- > void setShort(int index, short val);
- > void setLong(int index, long val);
- > void setString(int index, String val);
- > void setByte(int index, byte val);
- > void clearParameters();

- **\*\*\*ResultSet Interface:**

- > boolean next()
- > boolean previous()
- > boolean first()
- > boolean last()
- > void afterLast()
- > void beforeFirst()
- > void deleteRow()
- > int getInt(int column\_no)
- > int getInt(String column\_name)

-> float getFloat(int column\_no)  
-> float getFloat(String column\_name)  
-> float getDouble(int column\_no)  
-> float getDouble(String column\_name)  
-> String getString(int column\_no)  
-> String getString(String column\_name)

- **Code:-**

```
//Database Connection.
```

```
import java.sql.*;
```

```
class CreateTableDemo
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        try
```

```
        {
```

```
            Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
            System.out.println("Driver Loaded Successfully!!!");
```

```
            Connection
```

```
            con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521/XE","system","system");
```

```
            System.out.println("Connection Established Successfully!!!");
```

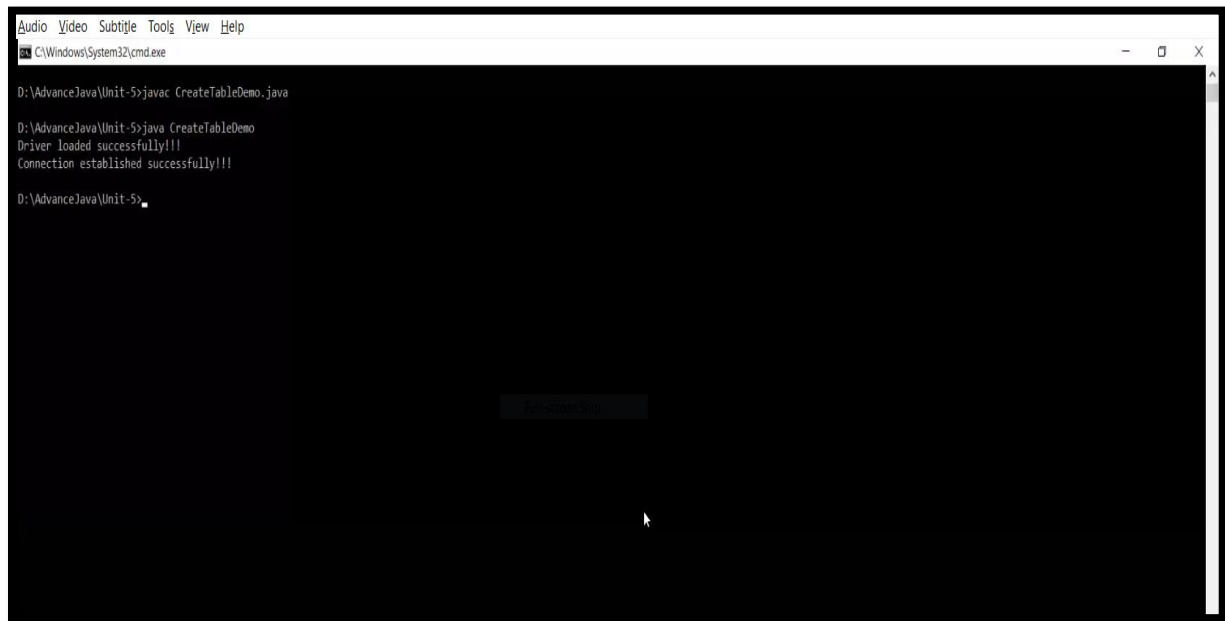
```
        }catch(Exception e)
```

```
        {
```

```
            System.out.println(e);
```

```
}  
  
}  
  
}
```

- **Output:-**



```
Audio Video Subtitle Tools View Help  
C:\Windows\System32\cmd.exe  
D:\AdvanceJava\Unit-5>javac CreateTableDemo.java  
D:\AdvanceJava\Unit-5>java CreateTableDemo  
Driver loaded successfully!!!  
Connection established successfully!!!  
D:\AdvanceJava\Unit-5>
```

\*\*\*\*\*

- **Code 1):-**

```
//Create Table Program.  
import java.sql.*;  
class CreateTableDemo  
{  
    public static void main(String args[])  
    {  
        try  
        {  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            System.out.println("Driver Loaded Successfully!!!");  
            Connection con=DriverManager.getConnection("jdbc:oracle:thin:  
@localhost:1521/XE","system","system");  
            System.out.println("Connection Established Successfully!!!");  
        }  
        catch (Exception e)  
        {  
            e.printStackTrace();  
        }  
    }  
}
```

```
Statement stmt=con.createStatement();
stmt.execute("create table vjtech(rollno number(5),name
varchar(10),marks number(3,2))");
System.out.println("Table Created Successfully!!!");
}
catch(Exception e)
{
    System.out.println(e);
}
}
```

### Output:-



```
C:\Windows\System32\cmd.exe
D:\AdvanceJava\Unit-5\javac CreateTableDemo.java
D:\AdvanceJava\Unit-5>java CreateTableDemo
Driver loaded successfully!!!
Connection established successfully!!!
D:\AdvanceJava\Unit-5>
D:\AdvanceJava\Unit-5>java CreateTableDemo
Driver loaded successfully!!!
Connection established successfully!!!
Table created successfully!!!
D:\AdvanceJava\Unit-5>
```

2)

```
import java.sql.*;
class InsertRowsDemo
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Driver Loaded Successfully!!!");
        }
    }
}
```

```
Connection con=DriverManager.getConnection("jdbc:oracle:thin:
@localhost:1521/XE","system","system");
System.out.println("Connection Established Successfully!!!");
Statement stmt=con.createStatement();
int x=stmt.executeUpdate("insert into vjtech values(3030,'James',9.9)");
System.out.println(" No of Rows Inserted into table:"+x);
}
catch(Exception e)
{
    System.out.println(e);
}
}
```

\*\*\*\*\*

**3)**

```
import java.sql.*;
class UpdateRowsDemo
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Driver Loaded Successfully!!!");
            Connection con=DriverManager.getConnection("jdbc:oracle:thin:
@localhost:1521/XE","system","system");
            System.out.println("Connection Established Successfully!!!");
            Statement stmt=con.createStatement();
            int x=stmt.executeUpdate("update vjtech set marks=5.5 where
rollno=2020");
            System.out.println(" No of Rows Updated from table:"+x);
        }
        catch(Exception e)
        {
```

```
        System.out.println(e);
    }
}
}
```

\*\*\*\*\*

**4)**

```
import java.sql.*;
class DeleteRowsDemo
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Driver Loaded Successfully!!!");
            Connection con=DriverManager.getConnection("jdbc:oracle:thin:
            @localhost:1521/XE","system","system");
            System.out.println("Connection Established Successfully!!!");
            Statement stmt=con.createStatement();
            int x=stmt.executeUpdate("delete from vjtech where rollno=3030");
            System.out.println(" No of Rows Deleted from table:"+x);
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

\*\*\*\*\*

**5)**

```
import java.sql.*;
class SelectRowsDemo
```



```

{
    public static void main(String args[])
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Driver Loaded Successfully!!!");
            Connection con=DriverManager.getConnection("jdbc:oracle:thin:
            @localhost:1521/XE","system","system");
            System.out.println("Connection Established Successfully!!!");
            Statement stmt=con.createStatement();
            ResultSet rs=stmt.executeUpdate("select *from vjtech");
            System.out.println("\n*****
            *****");
            System.out.println("\n*****VJTech Academy
            *****");
            System.out.println("\n*****
            *****");
            System.out.println("\nRollNo\tName\tmarks");
            System.out.println("\n=====");
            while(re.next())
            {

                System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+
                "\t"+rs.getFloat(3));
            }
            con.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}

```

\*\*\*\*\*

**6)**

```
import java.sql.*;
import java.util.*;

class PreparedStatementInsertRowsDemo
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Driver Loaded Successfully!!!");
            Connection con=DriverManager.getConnection("jdbc:oracle:thin:
@localhost:1521/XE","system","system");
            System.out.println("Connection Established Successfully!!!");
            PreparedStatement pstmt=con.prepareStatement("insert into vjtech
values(?,?,?)");
            Scanner sc=new Scanner(System.in);

            System.out.println("Enter Student Roll No:");
            int rn=sc.nextInt();
            System.out.println("Enter Student Name:");
            String rm=sc.next();
            System.out.println("Enter Student Marks:");
            float mrks=sc.nextFloat();

            pstmt.setInt(1,rn);
            pstmt.setString(2,nm);
            pstmt.setFloat(3,mrks);

            int x=pstmt.executeUpdate();

            System.out.println(" No of Rows Inserted into table:"+x);
        }
    }
}
```

```
catch(Exception e)
{
    System.out.println(e);
}
}
}
```

\*\*\*\*\*

7)

```
import java.sql.*;
import java.util.*;
```

```
class PreparedstmtDeleteRowsDemo
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Driver Loaded Successfully!!!");
            Connection con=DriverManager.getConnection("jdbc:oracle:thin:
            @localhost:1521/XE","system","system");
            System.out.println("Connection Established Successfully!!!");
            PreparedStatement pstmt=con.prepareStatement("delete from vjtech
            where rollno=?");
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter Roll No for Deletion:");
            int rn=sc.nextInt();
            pstmt.setInt(1,rn);

            int x=pstmt.executeUpdate();

            System.out.println(" No of Rows Deleted from table:"+x);
        }
    }
}
```

```
catch(Exception e)
{
    System.out.println(e);
}
}
}
```

\*\*\*\*\*

- **\*\*\*ResultSetMetaData**

-> Data about data is known as metadata.

```
ResultSet rs=stmt.executeQuery("select * from vjtech");
```

```
ResultSetMetaData rsmd=rs.getMedaData();
```

- 1) int getColumnCount() - return no of columns available in resultset.
- 2) String getColumnTypeName(int column\_no)- it will return SQL data type of column.
- 3) String getTableName(int column\_no)
- 4) boolean isAutoincrement(int column\_no);
- 5) boolean isNullable();

- **Code:-**

```
import java.sql.*;

class ResultSetMetaDataDemo
{
    public static void main(String args[])
    {

        try
```

```
{    Class.forName("oracle.jdbc.driver.OracleDriver");

    System.out.println("Driver Loaded Successfully!!!");

    Connection con=DriverManager.getConnection("jdbc:oracle:thin:
    @localhost:1521/XE","system","system");

    System.out.println("Connection Established Successfully!!!");

    Statement stmt=con.createStatement();

    ResultSet rs=stmt.executeUpdate("select *from vjtech");

    ResultSetMetaData rsmd=rs.getMetaData();

    System.out.println("Number
    of Coloums="+rsmd.getColumnCount());

    System.out.println("Name of
    1st Coloum="+rsmd.getColumnName());

    System.out.println("SQL Data Type of
    1st Coloum="+rsmd.getColumnTypeName(1));

    System.out.println("Name of table="+rsmd.getTableName(2));

    con.close();

}

catch(Exception e)

{

    System.out.println(e);

}

}

}
```

- **Java CallableStatement Interface:-**

CallableStatement interface is used to call the **stored procedures and functions**.

We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.

Suppose you need the get the age of the employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output.

**What is the difference between stored procedures and functions.**

The differences between stored procedures and functions are given below:

Stored Procedure	Function
is used to perform business logic.	is used to perform calculation.
must not have the return type.	must have the return type.
may return 0 or more values.	may return only one values.
We can call functions from the procedure.	Procedure cannot be called from function.
Procedure supports input and output parameters.	Function supports only input parameter.
Exception handling using try/catch block can be used in stored procedures.	Exception handling using try/catch can't be used in user defined functions.

- **How to get the instance of CallableStatement?**

The prepareCall() method of Connection interface returns the instance of CallableStatement. Syntax is given below:

1. **public** CallableStatement prepareCall("{ call procedurename(?,?,...?)}");

The example to get the instance of CallableStatement is given below:

1. CallableStatement stmt=con.prepareCall(" {call myprocedure(?,?)}");

It calls the procedure myprocedure that receives 2 arguments.

To call the stored procedure, you need to create it in the database. Here, we are assuming that stored procedure looks like this.

```
create or replace procedure "INSERTR"  
(id IN NUMBER,  
name IN VARCHAR2)  
is  
begin  
insert into user420 values(id,name);  
end;  
/
```

The table structure is given below:

```
create table user420(id number(10), name varchar2(200));
```

In this example, we are going to call the stored procedure INSERTR that receives id and name as the parameter and inserts it into the table user420. Note that you need to create the user420 table as well to run this application.

```
import java.sql.*;

public class Proc {

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

        Class.forName("oracle.jdbc.driver.OracleDriver");

        Connection con=DriverManager.getConnection(
            "jdbc.oracle:thin: @localhost:1521:xe", "system ","oracle");

        CallableStatement stmt=con.prepareCall("{call insertR(?,?)}");

        stmt.setInt(1,1011);

        stmt.setString(2,"Amit");

        stmt.execute();

        System.out.println("success");

    }

}
```

Now check the table in the database, value is inserted in the user420 table.

- **Code 1):-**

```
import java.sql.*;

import java.util.*;

class CallableStmtStoredProcedureDemo

{

    public static void main(String args[])

    {

        try

        {
```



```
Class.forName("oracle.jdbc.driver.OracleDriver");

System.out.println("Driver loaded successfully!!!");

Connectioncon=DriverManager.getConnection("jdbc:oracle:
thin:@localhost:1521/XE","system","system");

System.out.println("Connection established
successfully!!!");

CallableStatement Cstmt=con.prepareCall("{call
InsertRows(?,?,?)}");

Scanner sc=new Scanner(System.in);

System.out.println("Enter Student Roll No:");

int rn=sc.nextInt();

System.out.println("Enter Student Name:");

String nm=sc.next();

System.out.println("Enter Student Marks:");

float mrks=sc.nextFloat();


Cstmt.setInt(1,rn);

Cstmt.setString(2,nm);

Cstmt.setFloat(3,mrks);


int x=Cstmt.executeUpdate();


System.out.println("No of Rows Inserted into table through
stored procedure:"+x);
```

```
    }  
    catch(Exception e)  
    {  
        System.out.println(e);  
    }  
}  
}
```

- **Code 2):-**

```
import java.sql.*;  
import java.util.*;  
class CallableStmtFunctionsDemo  
{  
    public static void main(String args[])  
    {  
        try  
        {  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            System.out.println("Driver loaded successfully!!!");  
            Connectioncon=DriverManager.getConnection("jdbc:oracle:  
thin:@localhost:1521/XE","system","system");  
            System.out.println("Connection established  
successfully!!!");  
        }  
        catch(Exception e)  
        {  
            System.out.println(e);  
        }  
    }  
}
```

```
        CallableStatement Cstmt=con.prepareCall("{?=call  
        Sum1(?,?)}");  
  
        Cstmt.setInt(2,50);  
  
        Cstmt.setInt(3,80);  
        Cstmt.registerOutParameter(1,Types.INTEGER);  
  
        Cstmt.execute();  
  
        System.out.println("Addition of two  
        numbers="+Cstmt.getInt(1));  
    }  
    catch(Exception e)  
    {  
        System.out.println(e);  
    }  
}  
}
```

# *Inspiring Your Success*



**VJTech Academy...**