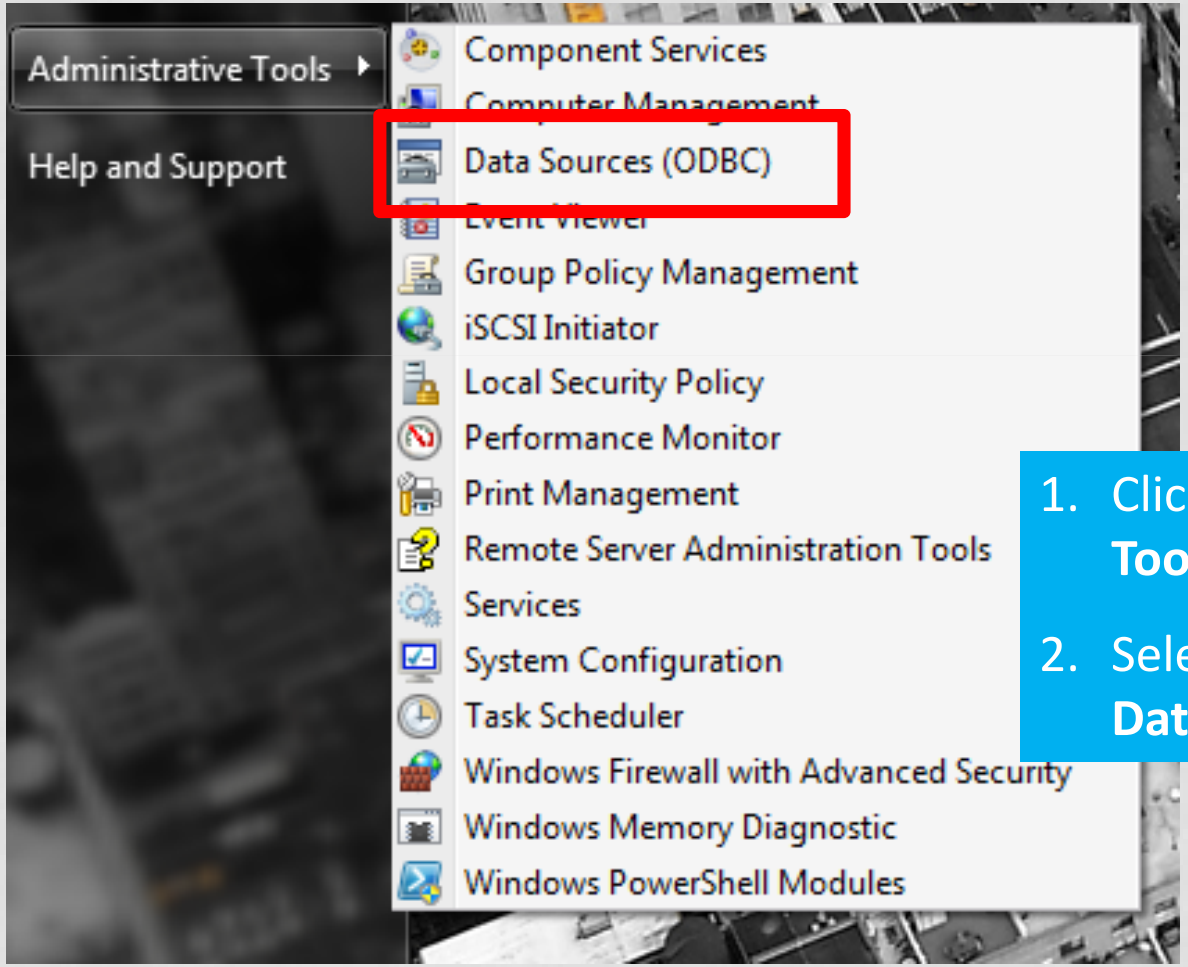
CONNECTING JAVA PROJECT TO DATABASE

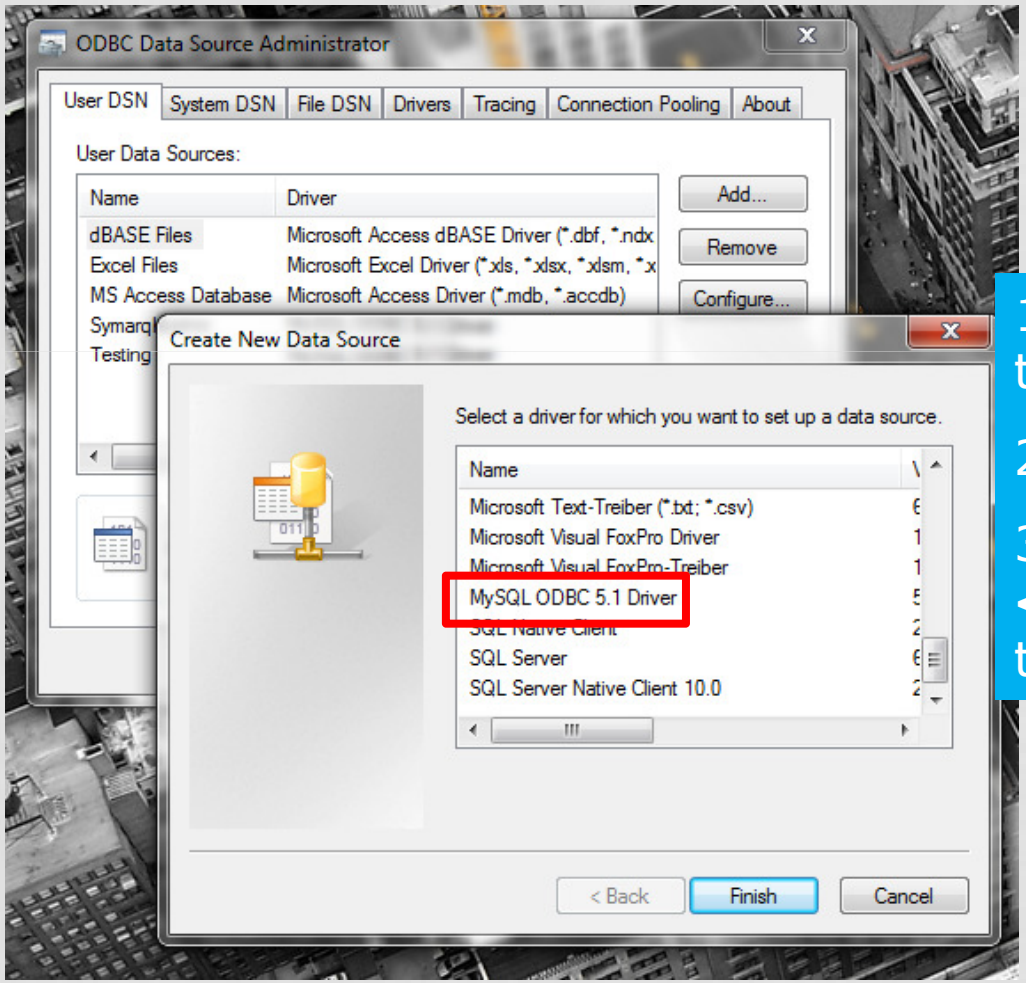
STEP 1: SETUP THE ODBC DRIVER



Make sure your SQL Server is running

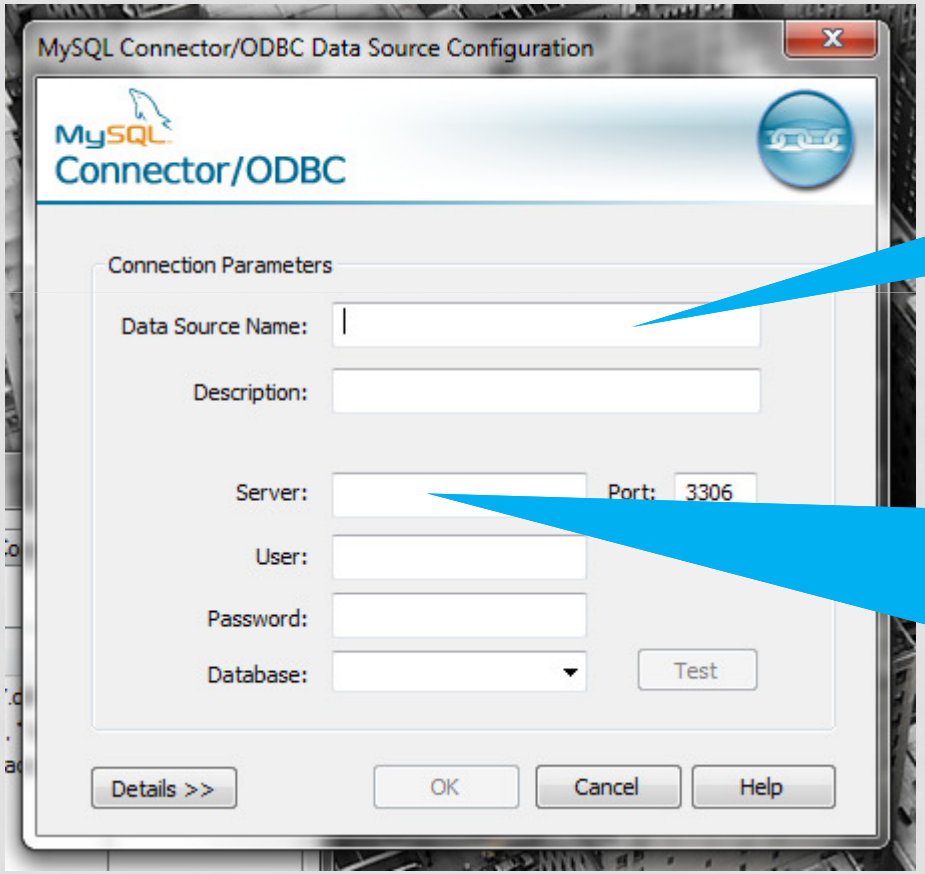
1. Click **Start -> Administrative Tools**
2. Select **Administrative Tools –> Data Sources (ODBC)**

STEP 1A: SETUP THE ODBC DRIVER



1. You should see the figure on the left in your screen
2. Click on the **Add** button
3. Select My**SQL ODBC <version> Driver** from the list, then click the **Finish** button

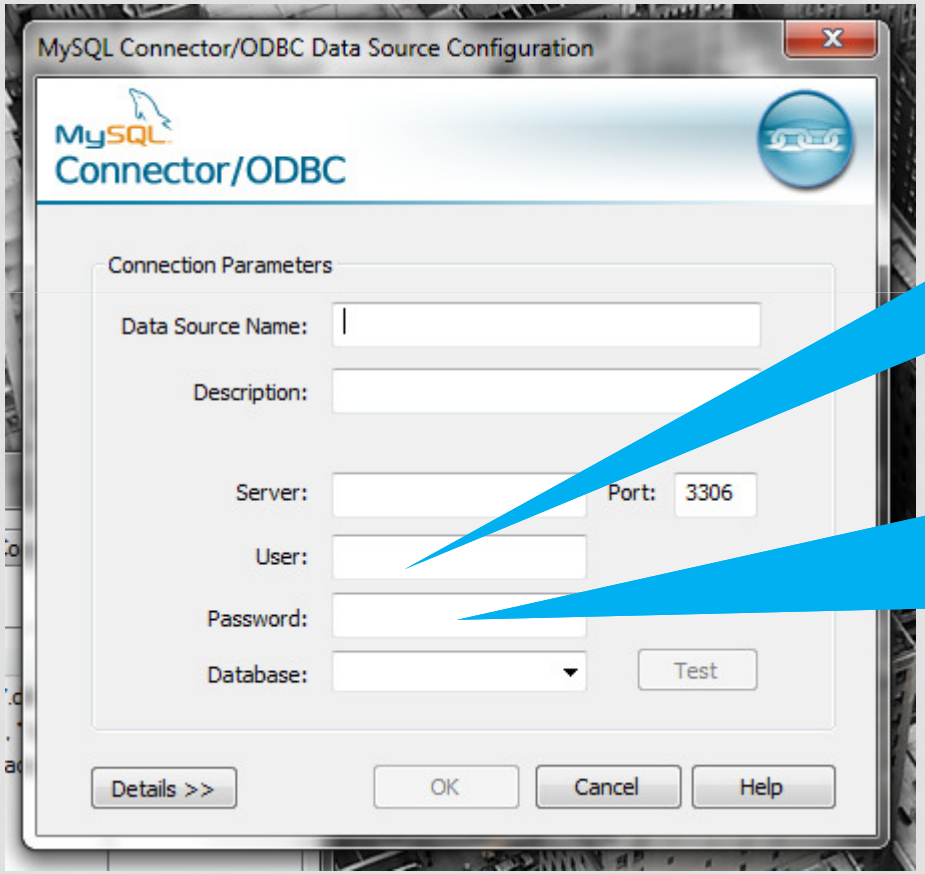
STEP 1B: SETUP THE ODBC DRIVER



Enter a data source name, e.g., Introdb_Airline

Enter the name of the SQL Server, e.g., your computer name is hosting your SQL Server. You may use localhost if the server is located locally or enter the IP address or hostname of the DB server

STEP 1C: SETUP THE ODBC DRIVER

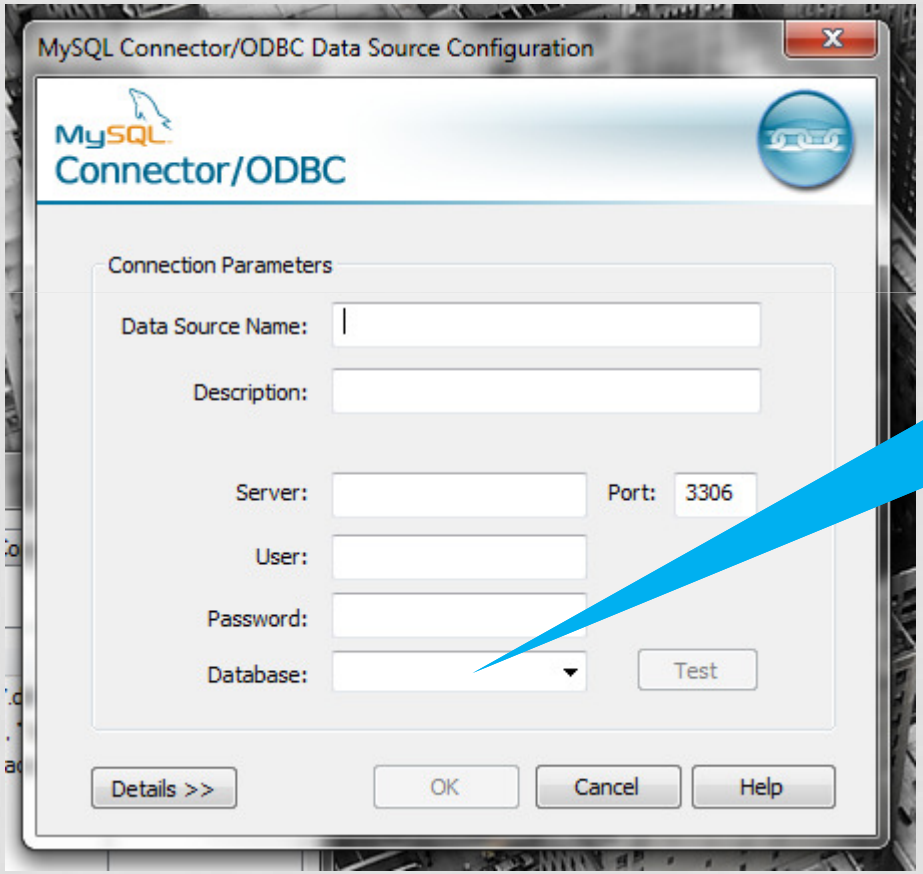


Enter "root" (for root access) or another username based on the accounts found in Server Administrator

Enter the password that you've entered during installation for root or the corresponding password for customized account

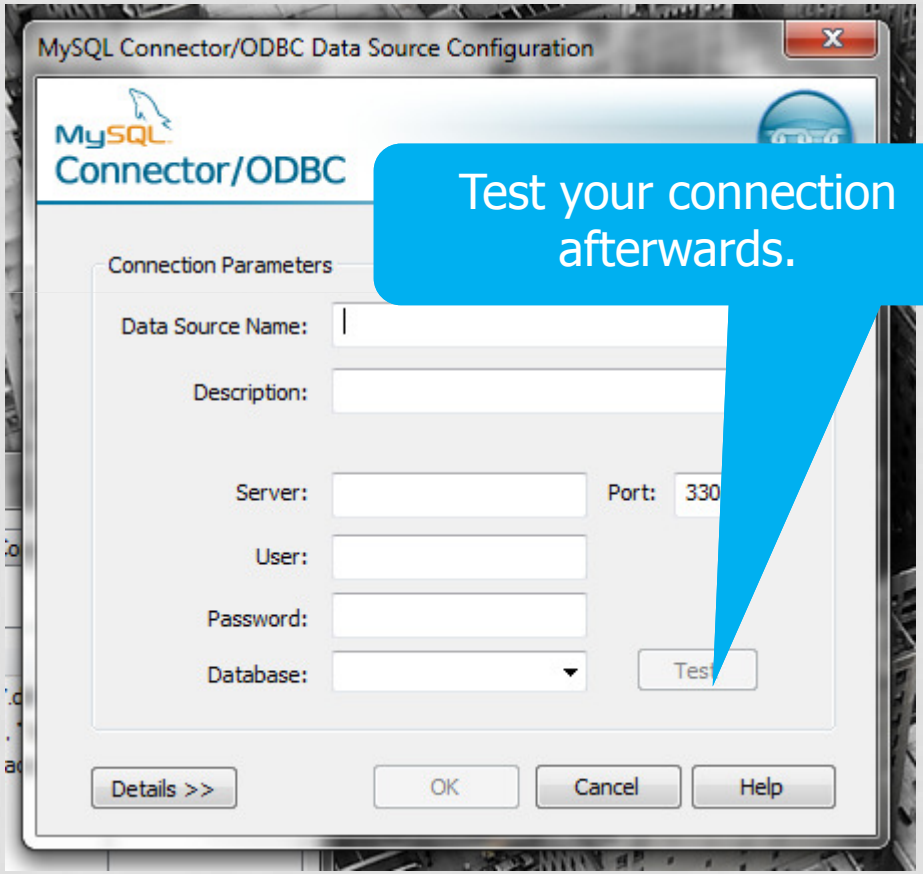
Note: It is recommended that you create your another account and **not use the root account**

STEP 1D: SETUP THE ODBC DRIVER



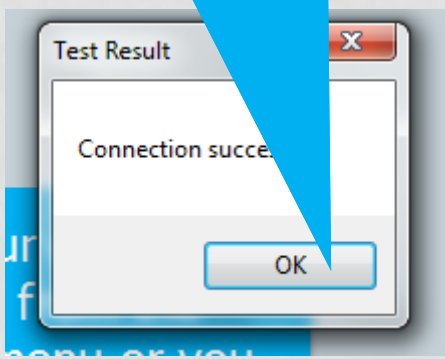
Select your created database from the dropdown menu or you may simply type it directly here

STEP 1F: SETUP THE ODBC DRIVER



Test your connection afterwards.

Connection to the database server has been tested; now you can write your application code



STEP 2: LOAD THE ODBC DRIVER

```
import java.sql.*;


Class.forName("com.mysql.jdbc.Driver");
```

Alternative syntax:

```
DriverManager.registerDriver(new sun.jdbc.odbc.JdbcOdbcDriver());
```

STEP 3: DEFINE THE CONNECTION URL

```
String url, uname, pword;


url = "jdbc:mysql://localhost/db_name";
uname = "root";
pword = "root";
```

STEP 4: ESTABLISH THE CONNECTION

```
Connection con =
DriverManager.getConnection (url, uname,
pword);
```

OR

```
Connection con =
DriverManager.getConnection("jdbc:mysql://local
host/devweb", "root", "root");
```

DriverManager.getConnection attempts to establish a connection with the data source through the specified url, and using the given username and password.

STEP 5: CREATE A STATEMENT

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

Then instantiate a Statement class that is required in order to execute a query. The Connection class returns a Statement object in its createStatement method that links the opened connection to the Statement object.

STEP 6: EXECUTE A QUERY

```
String query;
query = “SELECT * FROM products”;
ResultSet rs;
rs = stmt.executeQuery(query);
```

The ResultSet is linked to the connection to the data source via the Statement class. The Statement class contains the executeQuery method that submits the query to the JDBC driver for execution and returns a ResultSet object that contains the results.

STEP 7: PROCESS THE QUERY RESULTS

The next method of the ResultSet class fetches one row at a time

```
while (rs.next())
{
  System.out.println(
      "Product ID: "+rs.getInt("ProductID"));
  System.out.println(
      "Product Name: "+rs.getString("ProductName"));
  System.out.println(
      "Unit Price: " +rs.getFloat("UnitPrice"));
  System.out.println(
      "Units In Stock: " +rs.getInt("UnitsInStock"));
  System.out.println("---------------------------");
}
```

STEP 8: CLEAN UP

- `stmt.close();`
- `con.close();`

Closing the Statement object, in effect, closes the input-output query connection streams, but stay connected to the data source.

The last part involves terminating the connection to the data source.

STEP 9: EXCEPTION HANDLING

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

All statements from Steps 2 – 8 can throw exceptions if there was a problem with the driver, connection (establishing, maintaining), query, or retrieval of results.
So we have to enclose the statements in a try…catch block to catch any exceptions that may be thrown.

Save, compile, execute your java class

CODE 2: INSERTING A RECORD

```
String query = ("INSERT INTO `table_name`
(`column_1`,`column_2`, `column_3`) VALUES
('"+input1+"','" + input2 + "', '" +
input3+ "')");

Statement stmt = con.createStatement();
int code = stmt.executeUpdate(query);
```

CODE 3: MODIFYING RECORD(S)

```
String query = ("UPDATE `table_name` SET
`column_1`='" + input_1 + "' WHERE
`column_2`='" + input_2 + "';");


Statement stmt = con.createStatement();
int code = stmt.executeUpdate(query);
```

CODE 4: DELETING RECORDS

```
String query = ("DELETE FROM `table_name`
WHERE `column_1`='" + input_1 + "' ;");

Statement stmt = con.createStatement();
int code = stmt.executeUpdate(query);
```