(Write all SQL Strings inside " ")

JDBC – PART 1 – SEM 2 – JAVA II

Java Connection:

```
Import java.sql.*; XAMPP > start Apache & MySQL > start MySQL admin > create db > create Table
public class App { public static void main(String args[]) throws Exception {
   Class.forName("com.mysql.cj.jdbc.Driver");  //optional line
   Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dblju", "root", "");
   System.out.println((con != null) ? "Connection Sucessful" : "Connection Failed"); }
}
```

student		
sid	sname	smark
1	Riya	23.5
2	Diya	21
3	Priya	20.5
4	Jiya	24

Statement for DML (I,U,D): **- as they do changes in data so DML**

```
Statement st = con.createStatement();
String q1 = INSERT INTO student (sid, sname, smark) VALUES (5, 'Siya', 22);
int r = st.executeUpdate(q1);
System.out.println((r>0)? "insertion success" : "insertion failed");
---------------------------------------------------------------------------------------------
String q2= UPDATE student SET smark = 25 WHERE sid = 4;
System.out.println((st.executeUpdate(q2)>0) ? "updation success" : "updation failed");
---------------------------------------------------------------------------------------------
String q3 = DELETE FROM student WHERE sid = 3;
System.out.println((st.executeUpdate(q3)>0) ? "deletion success" : "deletion failed");
---------------------------------------------------------------------------------------------
```

Both DDL & DML:
st.execute(sql) -> DDL: true, DML: false
Only DML:
st.executeUpdate(sql) -> int r
Only DDL:
st.executeQuery(sql) -> ResultSet rs

Statement for DDL (SELECT): **-as it only gives data in return so DDL, no changes done in data by SELECT query.**

```
String q4 = SELECT * FROM student WHERE sid = 1;
ResultSet rs = st.executeQuery(q4);
while(rs.next()){
        System.out.println(rs.getInt(1) + " " + rs.getString("sname")+" "+rs.getDouble(3)); }
```

YOU MAY USE column Index OR column name ANYTHING IN GETTER

PreparedStatement for DML (I,U,D): **[Prepared means Placeholder (?)] //never use ? for column name or table name**

```
String q1 = INSERT INTO student (sid, sname, smark) VALUES (? , ?, ?);  // use ? for value/data only
PreparedStatement pst = con.prepareStatement(q1);
pst.setInt(1, 7);              OR  pst.setInt(1, sc.nextInt()); //bcz 1st placeholder is for column sid – INT so setInt
pst.setString(2, "HDS");       OR  pst.setString(2, sc.next());//bcz 2nd placeholder is for sname – VARCHAR so setString
pst.setDouble(3,25);           OR  pst.setDouble(3, sc.nextDouble()); //bcz 3RD is for smark – DOUBLE so setDouble
int r = pst.executeUpdate();
System.out.println((r>0)? "insertion success" : "insertion failed");
-----------------------------------------------------------------------------------
String q2= UPDATE student SET smark = ? WHERE sid = ?; // use setTYPE() for every ? for setting value/user data
PreparedStatement pst = con.prepareStatement(q2);
pst.setDouble(1,21);      //bcz 1st  is for smark – DOUBLE so setDouble
pst.setInt(2, 3);         //bcz 2nd placeholder is for column sid – INT so setInt
System.out.println((pst.executeUpdate()>0) ? "updation success" : "updation failed");
-----------------------------------------------------------------------------------------------------
String q3 = DELETE FROM student WHERE sid = ?;  // invalid if used as ''WHERE ?=?''
PreparedStatement pst = con.prepareStatement(q3);
pst.setInt(1, 4);         //bcz 1st placeholder is for column sid – INT so setInt
System.out.println((pst.executeUpdate()>0) ? "deletion success" : "deletion failed");
-----------------------------------------------------------------------------------------------------
```

PreparedStatement for DDL (SELECT):

```
String q4 = SELECT * FROM student WHERE sid = ?;
PreparedStatement pst = con.prepareStatement(q4);
pst.setInt(1, 2);         //bcz 1st placeholder is for column sid – INT so setInt
ResultSet rs = pst.executeQuery();
while(rs.next()){
        System.out.println(rs.getInt("sid") + " " + rs.getString(2)+" "+rs.getDouble(3)); }
```

YOU MAY USE column Index OR column name ANYTHING IN GETTER

Create Stored Procedure :

click on your Database (lju) > Routines Tab > create New Routine > give name > Keep "Procedure" in dropdown > add parameters only if placeholders are used in call query > write SQL query in definition between BEGIN & END > Save

without placeholder (fixed values)

CallableStatement for DML (I,U,D): **TABLE: faculty (fid INT 10, fname VARCHAR 20, fsal DOUBLE)**

```
String sql1 = "{call insertFac()}"; // Procedure: BEGIN insert into faculty(fid,fname,fsal) values (3,'hardik',433); END
CallableStatement cst = con.prepareCall(sql1);
int r = cst.executeUpdate();
System.out.println((r>0)? "insertion success" : "insertion failed");
---------------------------------------------------------------------------------------------------
String sql2 = "{call updateFac()}"; // Procedure: BEGIN update faculty set fname='hds' WHERE fid=1; END
CallableStatement cst = con.prepareCall(sql2);
System.out.println((cst.executeUpdate()>0) ? "updation success" : "updation failed");
---------------------------------------------------------------------------------------------------
String sql3 = "{call deleteFac()}"; // Procedure: BEGIN delete from faculty where fid=3; END
CallableStatement cst = con.prepareCall(sql3);
System.out.println((cst.executeUpdate()>0) ? "deletion success" : "deletion failed");
---------------------------------------------------------------------------------------------------
```

CallableStatement for DDL (SELECT):

```
String sql4 = "{call selectFac()}"; // Procedure: BEGIN SELECT * FROM faculty; END
CallableStatement cst = con.prepareCall(sql4);
ResultSet rs = cst.executeQuery();
while(rs.next()){
        System.out.println(rs.getInt(1) + " " + rs.getString("fname")+" "+rs.getDouble(3)); }
```

with placeholder (user values)

CallableStatement for DML (I,U,D): **TABLE: faculty(fid INT 10, fname VARCHAR 20, fsal DOUBLE)**

```
String sql5 = "{call insertion(?,?,?)}";
CallableStatement cst = con.prepareCall(sql5);
cst.setInt(1, 10);      OR (1,sc.nextInt())
cst.setString(2, "Hardik Shah"); OR (2,sc.next())
cst.setDouble(3, 50000); OR (3,sc.nexDouble())
int r = cst.executeUpdate();
System.out.println((r>0)?"inserted":"not inserted");
```

Procedure: insertion

```
Parameters:     IN – id INT 10
                IN – name VARCHAR 20
                IN – sal DOUBLE
BEGIN
    INSERT INTO faculty (fid,fname,fsal) VALUES (id,name,sal);
END
```

```
String sql6 = "{call updation(?,?)}";
CallableStatement cst = con.prepareCall(sql6);
cst.setString(1, "hardik");  OR (1,sc.next())
cst.setInt(2, 1);  OR (2,sc.nextInt())
int r = cst.executeUpdate();
System.out.println((r>0)?"updated":"not updated");
```

Procedure: updation

```
Parameters:     IN – name VARCHAR 20
                IN – id INT 10
BEGIN
    UPDATE faculty SET fname = name WHERE fid = id;
END
```

```
String sql7 = "{call deletion(?)}";
CallableStatement cst = con.prepareCall(sql7);
cst.setInt(1, 10);      OR (1,sc.nextInt())
int r = cst.executeUpdate();
System.out.println((r > 0)?"deleted":"not deleted");
```

Procedure: deletion

```
Parameters:     IN – id INT 10
BEGIN
      DELETE FROM faculty WHERE fid = id;
END
```

CallableStatement for DDL (SELECT):

```
String sql8 = "{call selection(?,?,?)}";
CallableStatement cst = con.prepareCall(sql8);
cst.setInt(1, 7);
cst.executeQuery();
System.out.println(cst.getString(2));
System.out.println(cst.getDouble(3));
```

Procedure: selection

```
Parameters:     IN – id INT 10;
                OUT – name VARCHAR 20
                OUT – sal DOUBLE
BEGIN
    SELECT fname,fsal INTO name,sal FROM faculty WHERE fid = id;
END
```

RSMD (table properties)

```
String sql = "SELECT * FROM faculty";
PreparedStatement pst = con.prepareStatement(sql);
ResultSet rs = pst.executeQuery();
ResultSetMetaData rsmd = rs.getMetaData();
System.out.println("Total Columns = "+rsmd.getColumnCount());
System.out.println("1st Column = " + rsmd.getColumnName(1));
System.out.println("2nd Column Type = " +
rsmd.getColumnTypeName(2));

System.out.println("Table Name = "+rsmd.getTableName(1));
```

DBMD(database properties)

```
DatabaseMetaData dbmd = con.getMetaData();
System.out.println("Driver Name: " + dbmd.getDriverName());
System.out.println("Driver Version: " + dbmd.getDriverVersion());
System.out.println("UserName is: " + dbmd.getUserName());
System.out.println("Product Name: " + dbmd.getDatabaseProductName());
System.out.println("Product Version: " + dbmd.getDatabaseProductVersion())
 String table[] = { "TABLE" };
    ResultSet rs = dbmd.getTables(null, null, null, table);
    while (rs.next())
        { System.out.println("Table Name = "+rs.getString(3)); }
```