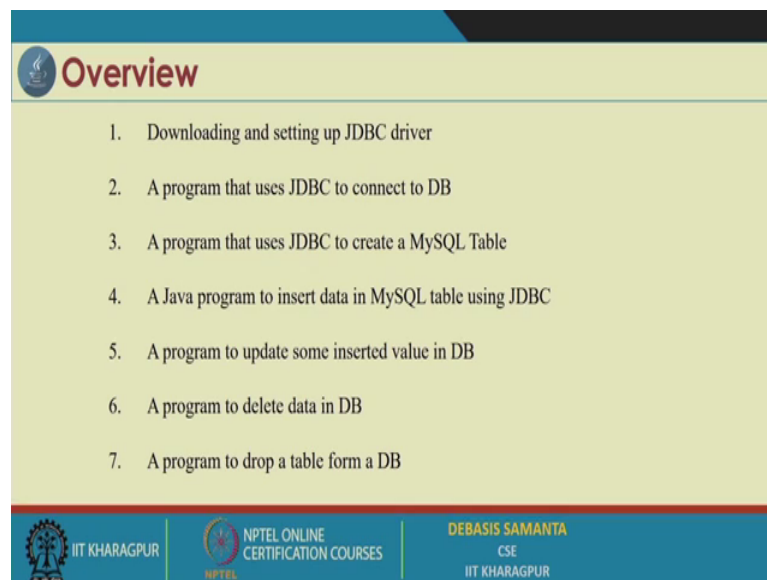


**Programming in Java**  
**Prof. Debasis Samanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 54**  
**Demonstration - XXI**

There are three things so far the JDBC is concerned; one is the database management system which is basically in this learning we have considered MySQL and then another is that JDBC driver which is basically you will connect to the server and then third component is basically the Java application.

(Refer Slide Time: 00:31)



The slide is titled "Overview" and contains a numbered list of seven topics. At the bottom, there are logos for IIT Kharagpur, NPTEL Online Certification Courses, and the instructor's name, Debasis Samanta, CSE, IIT Kharagpur.

1. Downloading and setting up JDBC driver
2. A program that uses JDBC to connect to DB
3. A program that uses JDBC to create a MySQL Table
4. A Java program to insert data in MySQL table using JDBC
5. A program to update some inserted value in DB
6. A program to delete data in DB
7. A program to drop a table form a DB

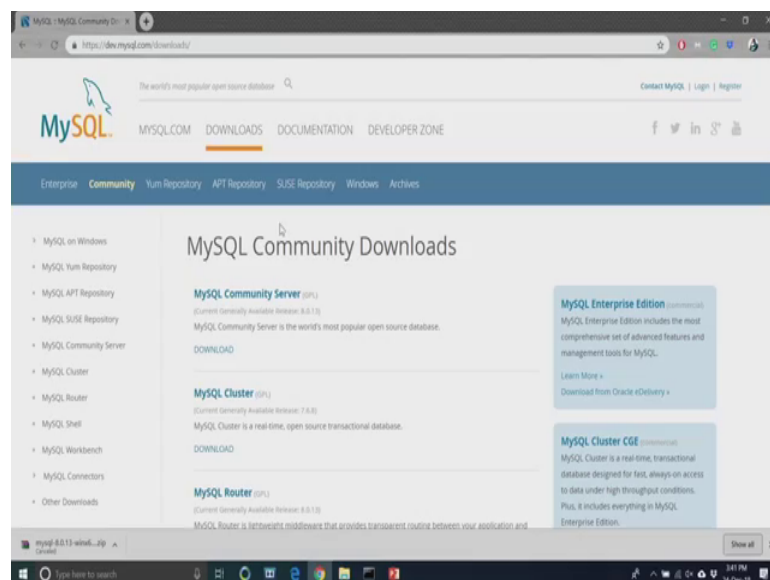
So, JDBC is the middle which will basically connect your program means Java application to server. Now in this session we will try to learn about the details about the JDBC driver.

So, first we should see exactly for this MySQL server, the proper JDBC driver needs to be installed. So, we will discuss about how the JDBC driver that will be responsible to connecting our Java application to the MySQL server. So, downloading the JDBC driver and then once the JDBC driver is installed successfully after downloading; obviously, the installation we will see exactly how this can be installed. And then finally, the through the JDBC driver and then Java program how we can execute some statement, SQL statement; and we will execute the SQL statement from our Java application Java

program, and then we will see their results from the console again; that means, through the again connecting them is MySQL server after executing. This basically gives an idea about that whatever the comment that we are intended to execute form the application whether it is successfully executed or not.

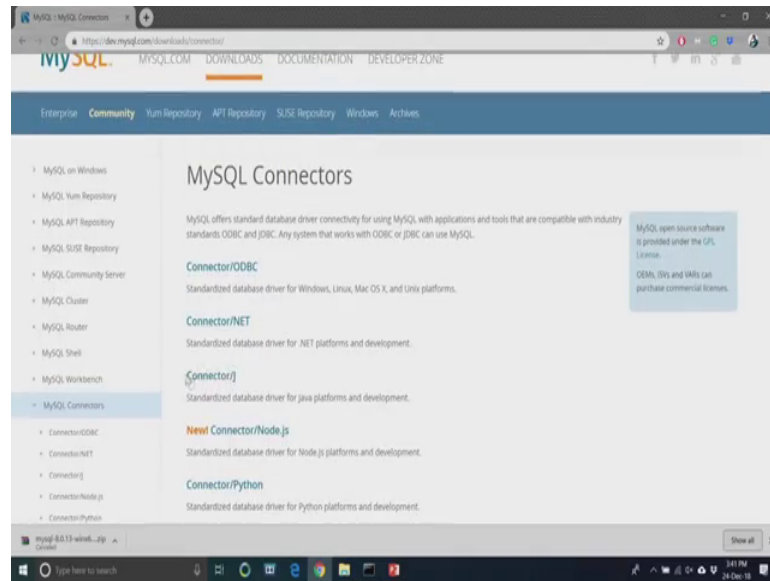
In our next session that will be held later on we will see exactly how the results also that can be obtained purely from the application itself ok. Now let us start about how the JDBC driver can be installed. The location of the JDBC driver is we have given the link is there.

(Refer Slide Time: 02:11)



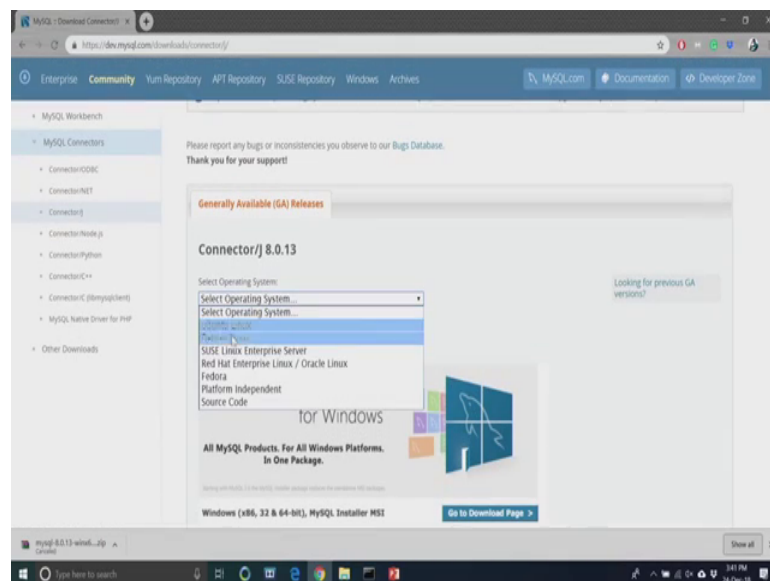
So, dev https then colon then double slash dev dot mysql dot com downloads under then slash connectors slash j slash. So, this basically is the link from where you can get the JDBC driver for the MySQL connector.

(Refer Slide Time: 02:29)



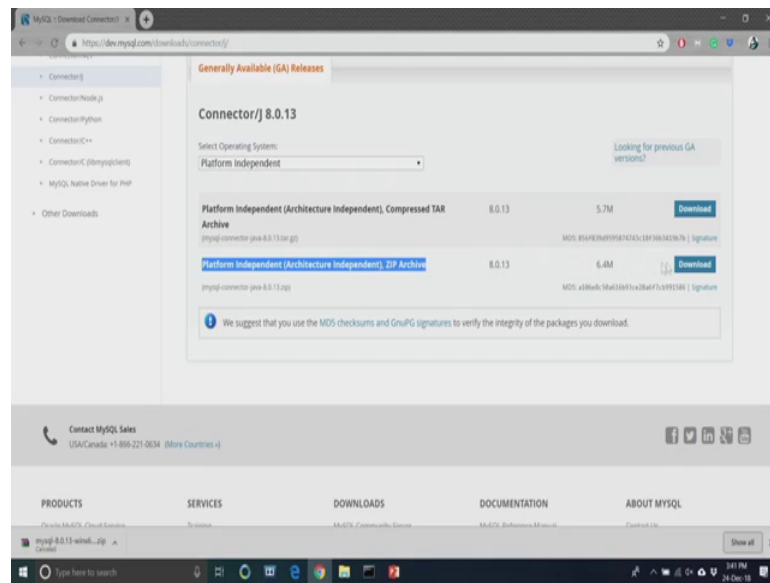
So, MySQL connector is there and there is a connector J.

(Refer Slide Time: 02:35)



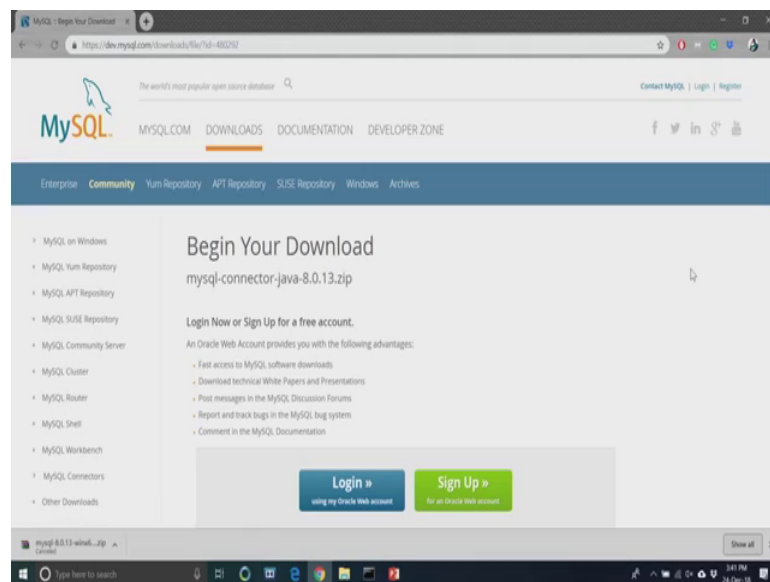
You just go to the link and finally, you will be able to set the proper program there.

(Refer Slide Time: 02:41)



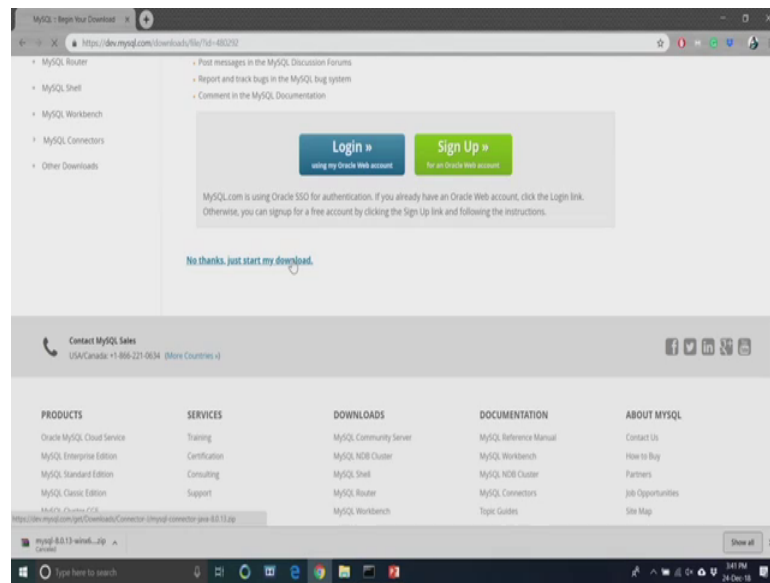
So, it is basically platform independent one version that you have to install it. And so, this is the version this is a zip file of course, and the version is for current SQL 8.0.13.

(Refer Slide Time: 02:51)



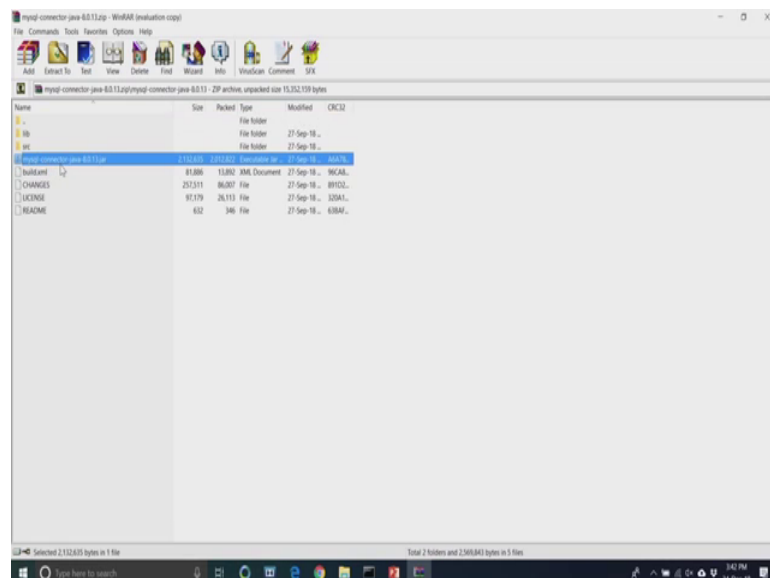
Is just simply start downloading. So, you basically download now.

(Refer Slide Time: 02:53)



So, download from the net, it is a free software. So, there is no issue. So, for the downloading this software is concerned. Once the JDBC driver is download then you have to extract the zip file, and it can we download in any directory. So, we will just advise you to download into on C drive right ok. So, we just unzip this zip file and then we, once you do the unzip, then you can find so many folders there is a one folders called jdbc mysql connector dot jar file.

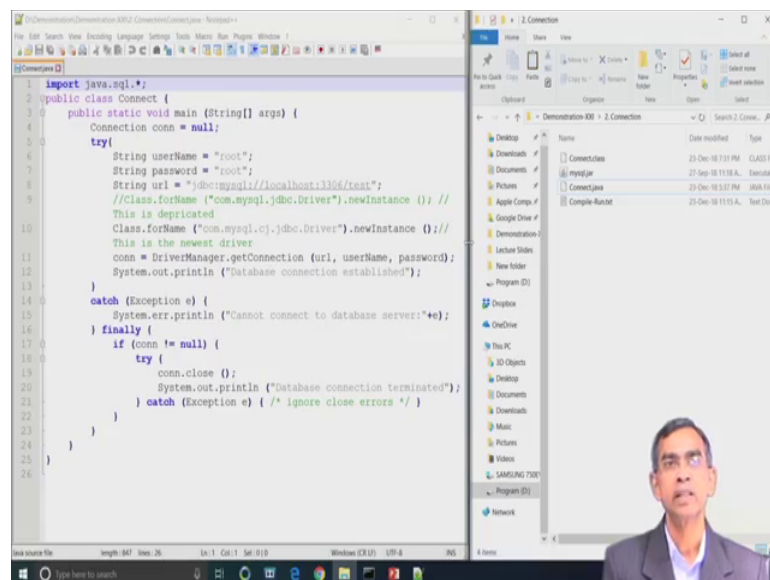
(Refer Slide Time: 03:29)



So, this file you should rename to mysql for just for simplicity, you can give any name, but for convenience you can give this one and then finally, you should place the jar file mysql dot jar file along with any other program which includes in this using jdbc, that should be done by one command from the command prompt java minus cp mysql dot jar and then what the all file name fine.

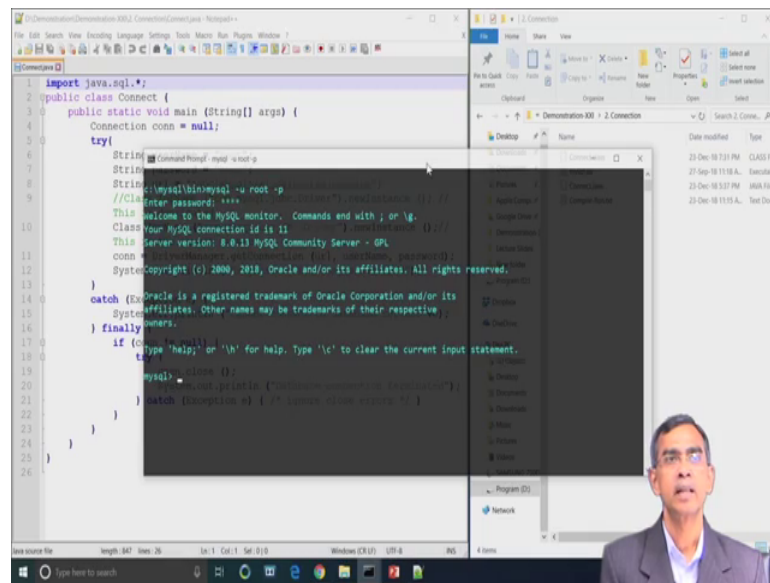
So, you just simply copy this jar file into this one right. So, we have copied it right. So, we have copied this mysql jar file and this and then we have to finally, set the class path name from the system file itself so that this JDBC driver can be identified from any location. If you execute it, so, you have to set the class path also there.

(Refer Slide Time: 04:11)



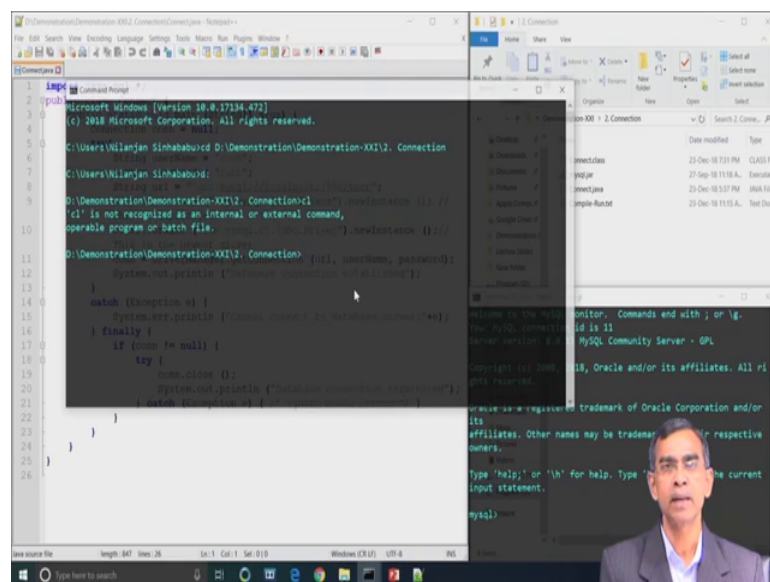
Anyway, so, our JDBC connection this is this gives you complete competition about the JDBC connection is there.

(Refer Slide Time: 04:37)



Now, you have to login to the mysqlserver because we are quit from we are exited from this one, password is root here now ok. So, the JDBC this MySQL connection is done currently it is there.

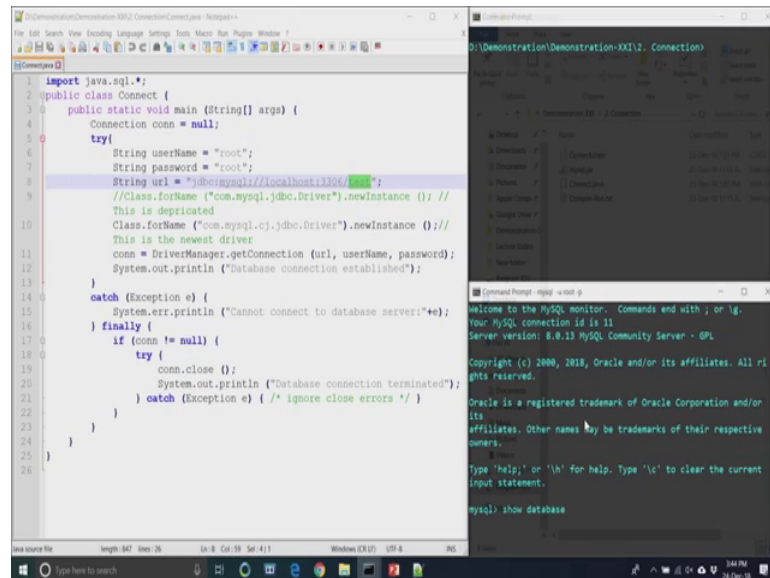
(Refer Slide Time: 04:51)



Now, we want to write some Java program by which you can execute some SQL statement and then after the execution of statement, we will be able to see the result. Now see there are few steps; obviously, involved. The first thing is that you have to create few objects; the first is that connection object we give the name as conn.

We define connection object and then we define ok. So, this is basically setting up the connection and then username is root, we have to write is username root password is root, url is basically you can note it jdbc mysql this is basically local host and this is the number that is there, is a port number is 3306 and the JDBC, the drivers should connect to a particular data base. So, the last field is basically the name of the database.

(Refer Slide Time: 05:49)

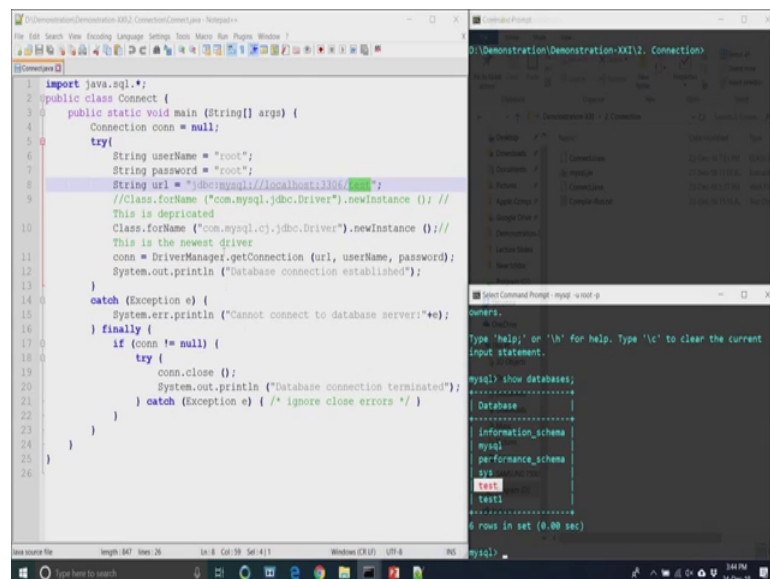


```
1 import java.sql.*;
2 public class Connect {
3     public static void main (String[] args) {
4         Connection conn = null;
5         try{
6             String userName = "root";
7             String password = "root";
8             String url = "jdbc:mysql://localhost:3306/";
9             //Class.forName ("com.mysql.jdbc.Driver").newInstance () //
10            // This is deprecated
11            Class.forName ("com.mysql.cj.jdbc.Driver").newInstance ();//
12            // This is the newest driver
13            conn = DriverManager.getConnection (url, userName, password);
14            System.out.println ("Database connection established");
15        }
16        catch (Exception e) {
17            System.err.println ("Cannot connect to database server:"+e);
18        }
19        finally {
20            if (conn != null) {
21                try {
22                    conn.close ();
23                    System.out.println ("Database connection terminated");
24                } catch (Exception e) { /* ignore close errors */ }
25            }
26        }
27    }
28 }
```

```
D:\Demonstration\Demonstration-XXI\2. Connection>
mysql> show database;
```

As we have know already we have created one database in earlier session, and the same database will be used there.

(Refer Slide Time: 05:59)



```
1 import java.sql.*;
2 public class Connect {
3     public static void main (String[] args) {
4         Connection conn = null;
5         try{
6             String userName = "root";
7             String password = "root";
8             String url = "jdbc:mysql://localhost:3306/";
9             //Class.forName ("com.mysql.jdbc.Driver").newInstance () //
10            // This is deprecated
11            Class.forName ("com.mysql.cj.jdbc.Driver").newInstance ();//
12            // This is the newest driver
13            conn = DriverManager.getConnection (url, userName, password);
14            System.out.println ("Database connection established");
15        }
16        catch (Exception e) {
17            System.err.println ("Cannot connect to database server:"+e);
18        }
19        finally {
20            if (conn != null) {
21                try {
22                    conn.close ();
23                    System.out.println ("Database connection terminated");
24                } catch (Exception e) { /* ignore close errors */ }
25            }
26        }
27    }
28 }
```

```
D:\Demonstration\Demonstration-XXI\2. Connection>
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| test |
| test1 |
+-----+
6 rows in set (0.00 sec)

mysql>
```

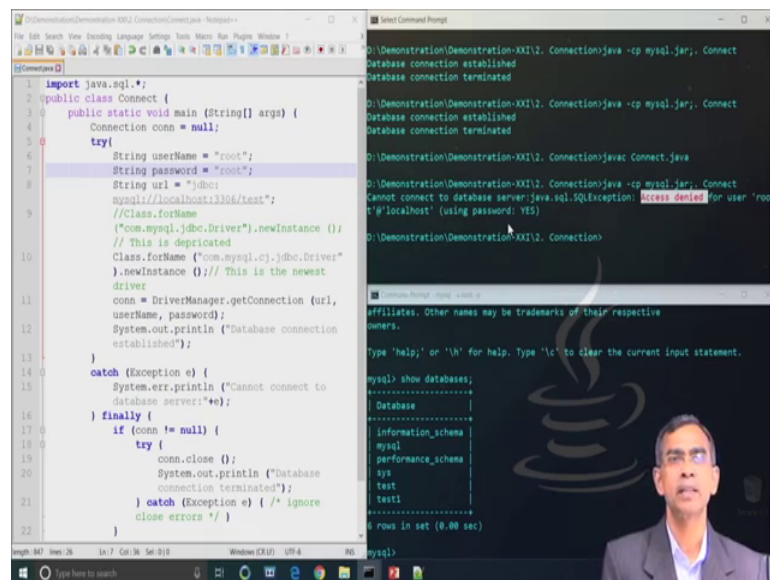


And then we have to load the JDBC driver by using the statement is basically the Class dot forName then com dot mysql dot cj dot jdbc dot Driver and all this thing within double quotes dot newInstance this is the method and so, this basically is to load the JDBC driver into your application actually ok. So, this is the right and then we have to establish the connection created instance of the collection of object for which the DriverManager is the class there.

So, DriverManager then dot getConnection and giving the url. Whatever the url you have given there you just give the argument here url, userName, password; otherwise you can in this method also you can type within double quote all the argument this also will work, but for the convenience we store it and then pass it as a method arguments.

So, this basically create the connection objects and then if the connection is successful from the program, we will get if there is no connection successful it will given an error. Now let us see whether error or not there. So, for the error handling we have already enclose everything under the try catch block. So, these are customary try catch block that we have enclosed it now ok. So, run this program java minus ct yes.

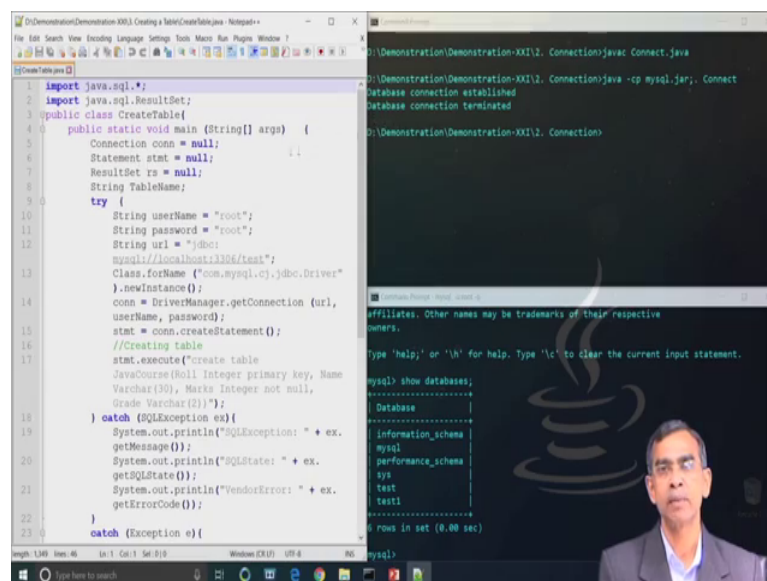
(Refer Slide Time: 07:21)



So, now we have already loaded this one here java minus cp mysql dot jar connect. Now connection is there. Now we can compile this program. So, program is successfully compiled and executed after the execution it is there. If we see the database connection established database connection is now terminated also.

Now, if I suppose there is a mistake for either say some name or some say root password is changed; so, guest g u e s t, suppose this is a wrong password then if we again try to connect it, java minus cp mysql dot jar connect as you see it basically compile it. Compilation is done and then running this program, as you see cannot connect to database server this is because this one. So, exception is caused access denied for user root localhost using password yes. So, this basically refuse to get the connection. So, root is the password that is required for the correct or successful connection at this moment.

(Refer Slide Time: 08:39)

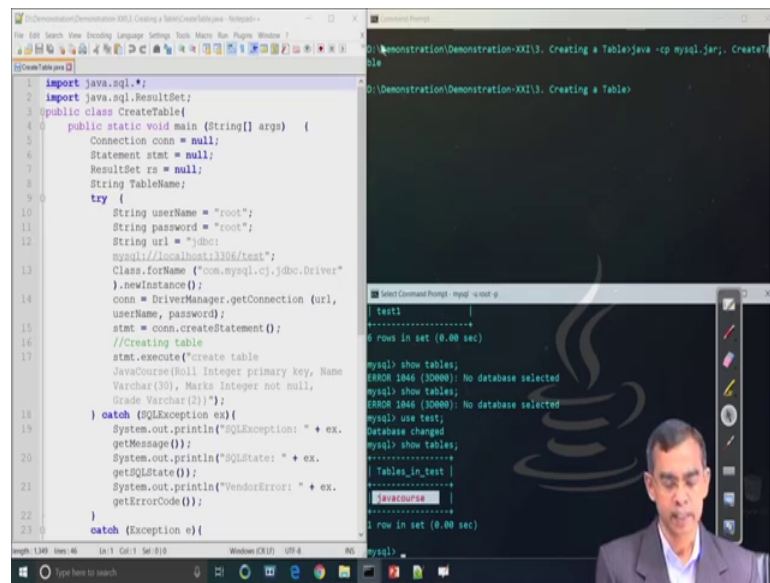


```
1 import java.sql.*;
2 import java.sql.ResultSet;
3 public class CreateTable{
4     public static void main (String[] args) {
5         Connection conn = null;
6         Statement stmt = null;
7         ResultSet rs = null;
8         String TableName;
9         try {
10            String userName = "root";
11            String password = "root";
12            String url = "jdbc:
13            mysql://localhost:3306/test";
14            Class.forName ("com.mysql.cj.jdbc.Driver"
15            ).newInstance();
16            conn = DriverManager.getConnection (url,
17            userName, password);
18            stmt = conn.createStatement();
19            //Creating table
20            stmt.executeUpdate("create table
21            JavaCourse (Roll Integer primary key, Name
22            Varchar(30), Marks Integer not null,
23            Grade Varchar(2))");
24        } catch (SQLException ex){
25            System.out.println("SQLException: " + ex.
26            getMessage());
27            System.out.println("SQLState: " + ex.
28            getSQLState());
29            System.out.println("VendorError: " + ex.
30            getErrorCode());
31        }
32        catch (Exception e){
33        }
```

```
D:\Demonstration\Demonstration-xxii\1. Connection>javac Connect.java
Database connection established
Database connection terminated
D:\Demonstration\Demonstration-xxii\1. Connection>
D:\Demonstration\Demonstration-xxii\1. Connection>
mysql show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| test |
+-----+
6 rows in set (0.00 sec)
```

So, this way you can see how the connection can be made with the JDBC to the MySQL server through the JDBC there. Now we will discussed about how the simple statement can be executed. Here the simple statement that we are going to execute is, create table java course because there is no table at the moment. So, as there is no table we will not be able to accomplish any insert, delete, update command.

(Refer Slide Time: 09:07)



```
1 import java.sql.*;
2 import java.sql.ResultSet;
3 public class CreateTable{
4     public static void main (String[] args) {
5         Connection conn = null;
6         Statement stat = null;
7         ResultSet rs = null;
8         String tableName;
9         try {
10            String userName = "root";
11            String password = "root";
12            String url = "jdbc:
13            mysql://localhost:3306/test";
14            Class.forName ("com.mysql.cj.jdbc.Driver"
15            ).newInstance();
16            conn = DriverManager.getConnection (url,
17            userName, password);
18            stat = conn.createStatement();
19            //Creating table
20            stat.executeUpdate("create table
21            JavaCourse (roll: Integer primary key, Name
22            Varchar(30), Marks Integer not null,
23            Grade Varchar(2));");
24        } catch (SQLException ex){
25            System.out.println("SQLException: " + ex.
26            getMessage());
27            System.out.println("SQLState: " + ex.
28            getSQLState());
29            System.out.println("VendorError: " + ex.
30            getErrorCode());
31        }
32    }
33 }
```

```
mysql: show tables;
ERROR 1046 (30000): No database selected
mysql: show tables;
ERROR 1046 (30000): No database selected
mysql: use test;
Database changed
mysql: show tables;
+-----+
| Tables_in_test |
+-----+
| JavaCourse      |
+-----+
1 row in set (0.00 sec)
```

So, first we have to create the table. Now this is the one program as we see. This program is basically tell how we can create a table. Now here you see this is the Connection object is created and then the Statement object is created of the class Statement, ResultSet it basically it will return result set; in this case we will not consider all though, but it can give the results. And here is basically userName; root and then password also root we have stored and then url is jdbc here; jdbc mysql localhost this is basically url for the our MySQL server in this case.

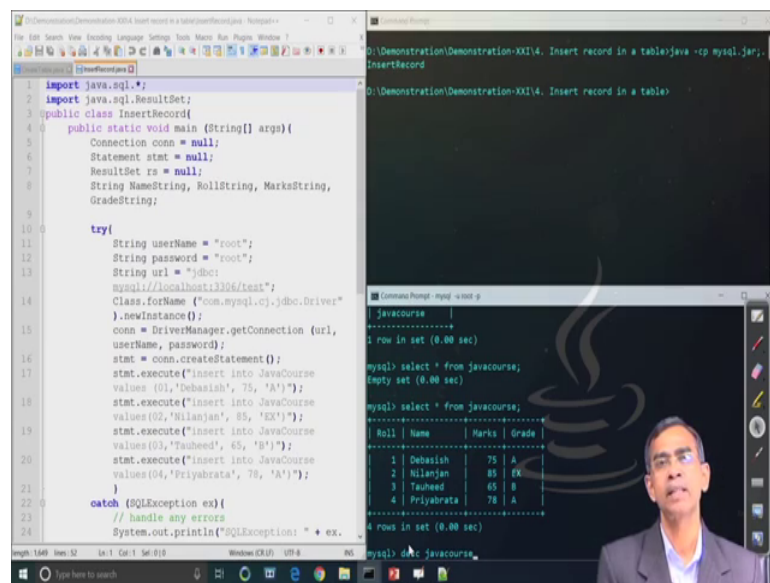
Then we establish the connection using this command Class dot forName com mysql dot cj dot jdbc dot Driver then dot newInstance. This is basically the syntax for which the connection can be made. When the connection is there is successful then we have to create the connection object through the DriverManager getConnection giving the argument; url, userName and password which is basically assigned here. And then so, this basically completes the connection and here is basically we are creating a statement object.

So, statement for conn dot createStatement object is created and then this statement object is initialized with one statement the create table; JavaCourse Roll Integer primary key, Name Varchar 30, Marks Integer not null, Grade Varchar 2. So, it is basically this is the way the usual way, that a table can be created with four fields namely roll number name marks and grade like.

And so, now so, this basically completes the firing the execution and then if there is any error or whatever it is there it will obviously, catch by the try catch block. So, this basically is the program which basically create a connection objects connect to the server through JDBC driver and finally, execute a statement to create a new table and the command is that java minus cp mysql dot jar and then create table this is the name of the class that we have created here.

So, this is basically the program and then in the server we are coming to the console to the server and here show table use test show table. So, we see the new table java course has been created under the test database. So, this is the first step or creating the table here. Now let us to execute some others statement from this table. First is that we have to insert few records into the table.

(Refer Slide Time: 11:53)



The screenshot shows a Java IDE on the left and a terminal on the right. The IDE contains the following Java code:

```
1 import java.sql.*;
2 import java.sql.ResultSet;
3 public class InsertRecord{
4     public static void main (String[] args){
5         Connection conn = null;
6         Statement stmt = null;
7         ResultSet rs = null;
8         String NameString, RollString, MarksString,
9         GradeString;
10
11     try{
12         String userName = "root";
13         String password = "root";
14         String url = "jdbc:
15         mysql://localhost:3306/test";
16         Class.forName ("com.mysql.cj.jdbc.Driver"
17         ).newInstance();
18         conn = DriverManager.getConnection (url,
19         userName, password);
20         stmt = conn.createStatement();
21         stmt.executeUpdate ("insert into javacourse
22         values (01,'Debasish', 75, 'A')");
23         stmt.executeUpdate ("insert into javacourse
24         values (02,'Nilanjan', 85, 'EX')");
25         stmt.executeUpdate ("insert into javacourse
26         values (03,'Taubeed', 65, 'B')");
27         stmt.executeUpdate ("insert into javacourse
28         values (04,'Priyabrata', 78, 'A')");
29     }
30     catch (SQLException ex){
31         // handle any errors
32         System.out.println ("SQLException: " + ex.
33         .getMessage());
34     }
35 }
```

The terminal on the right shows the execution of the program and the output of SQL queries:

```
O:\Demonstration\Demonstration> java -cp mysql.jar;
InsertRecord
O:\Demonstration\Demonstration> mysql -u root -p
mysql> create table javacourse;
mysql> select * from javacourse;
+-----+
1 row in set (0.00 sec)
mysql> select * from javacourse;
Empty set (0.00 sec)
mysql> select * from javacourse;
+----+-----+-----+-----+
Roll | Name | Marks | Grade
+----+-----+-----+-----+
1 | Debasish | 75 | A
2 | Nilanjan | 85 | EX
3 | Taubeed | 65 | B
4 | Priyabrata | 78 | A
+----+-----+-----+-----+
4 rows in set (0.00 sec)
mysql> create javacourse;
```

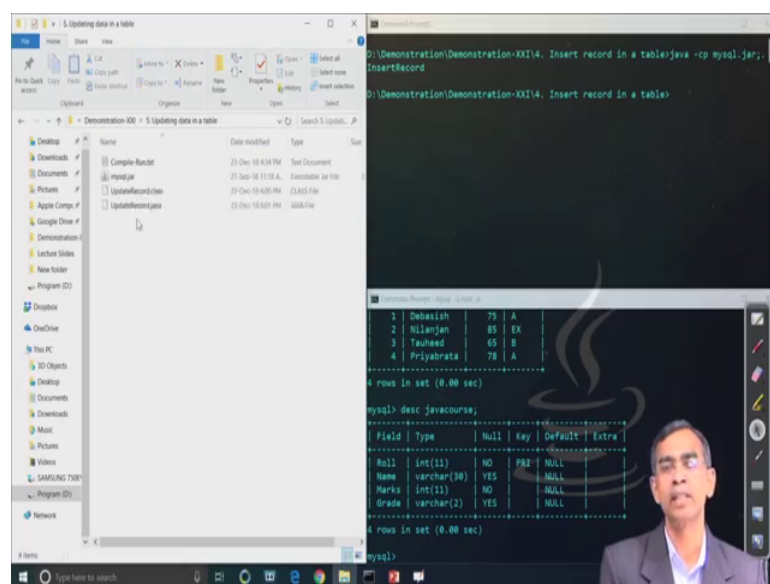
So, we will see exactly the table that you have just now created from our Java application how the same can be used to insert some data, we will create many records as many we wish. Here we will consider four records to be inserted with one SQL statement. Now this is a program as you see this is a program. You see here first of all these are the. So, this is the conn is the Connection object, the stmt is the Statement object and then here is the basically userName and password and url these are the three parameters that required to install the DriverManager.

So, we use these object and then finally, through DriverManager we connect it. So, the conn object is now ready; that means, its connection is established and then we just create a statement object and then statement is basically created with execution for request with four insert records one by one. Here you see this is the first insert then semicolon, then second statement dot execute second insert and another statement dot execute the third insert and then the fourth insert for this fourth execute statement.

So, this way we have issued the four a statement and then we call the executive method for this statement object to execute these are the four statement. This basically the request will pass through JDBC to the server, and server will execute it and then finally, we will see whether server has done the things correctly according to the request that is there in the application.

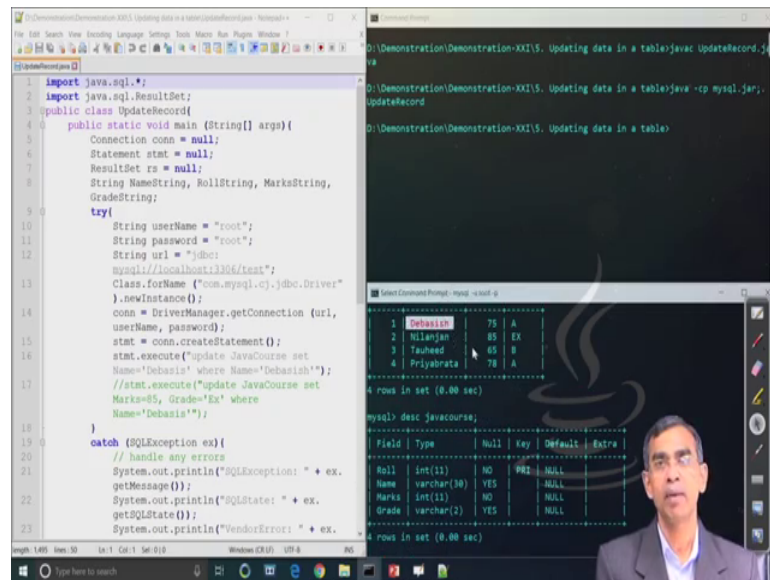
So, here again we are running the program, this is the name of the classic insert records. So, right in ok. So, the common is java minus cp mysql dot jar that you have to give it colon then you have to give the InsertRecord. This is the syntax that you have to follow if you want to communicate it. So, there is no error key message that means it has been executed. So, let us come back to the table here again. So, we can see select star from java course, it is already there. Here also if we write say desc table java course also it will give the same things as usual because the table has been successfully created with the structure that we have given it there.

(Refer Slide Time: 14:21)



So, it is successfully created. Now so, table is created four records has been have been inserted into it. Now we will see exactly the Update command how this can be executed from the Java application through JDBC driver to MySQL.

(Refer Slide Time: 14:43)



```
1 import java.sql.*;
2 import java.sql.ResultSet;
3 public class UpdateRecord{
4     public static void main (String[] args){
5         Connection conn = null;
6         Statement stat = null;
7         ResultSet rs = null;
8         String NameString, RollString, MarksString,
9         GradeString;
10        try{
11            String userName = "root";
12            String password = "root";
13            String url = "jdbc:
14            mysql://localhost:3306/test";
15            Class.forName ("com.mysql.cj.jdbc.Driver"
16            ).newInstance();
17            conn = DriverManager.getConnection (url,
18            userName, password);
19            stat = conn.createStatement();
20            stat.executeUpdate("update JavaCourse set
21            Name='Debasish' where Name='Debasish'");
22            //stat.executeUpdate("update JavaCourse set
23            Marks=85, Grade='Ex' where
24            Name='Debasish");
25        }
26        catch (SQLException ex){
27            // handle any errors
28            System.out.println("SQLException: " + ex.
29            getMessage());
30            System.out.println("SQLState: " + ex.
31            getSQLState());
32            System.out.println("VendorError: " + ex.
33            getErrorCode());
34        }
35    }
36 }
```

```
mysql> select * from javacourse;
+----+-----+-----+-----+
| Id | Name   | Marks | Grade |
+----+-----+-----+-----+
| 1  | Debasish | 75    | A     |
| 2  | Nilanjana | 85    | Ex    |
| 3  | Trained | 80    | B     |
| 4  | Priyabrata | 78    | A     |
+----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> desc javacourse;
+----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+----+-----+-----+-----+
| Roll | int(11) | NO | PRI | NULL | |
| Name | varchar(30) | YES | | NULL | |
| Marks | int(11) | NO | | NULL | |
| Grade | varchar(2) | YES | | NULL | |
+----+-----+-----+-----+
4 rows in set (0.00 sec)
```

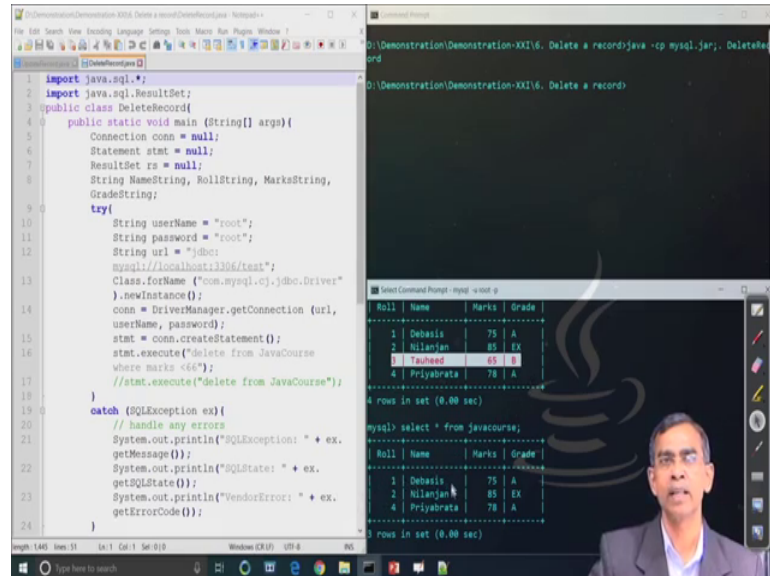
Again let us look at the code here ok. This code is as usual this is the Connection, Statement objects and these are userName, password and then this is the url and we create the Class dot forName passing the userName, password and then url.

And then finally, we create the conn object; that means, connection object in calling the DriverManager dot getConnection method with the url, userName and password and then this is the statement object is now ready to fire. Now here the statement you see we use which command, SQL command to include it here update JavaCourse set. Name equals to this one where Name equals to this one. So, here is basically and you see in the first table that we have discussed there is a one entry basically the first entry for which the name is Debasish d e b a s i s h. So, it is basically this command is requested to change this entry as a d e b a s i s. So, now, we are this program is now ready and then let us compile this program and then ok.

So, this program yes. So, update record this is the name of the file yes. Now the record has been updated and let us come back to the command prompt. So, command prompt again just see exactly select star from you can see the updation is successful. So, this way the command relative update can be fired from the Java application and then we will see

exactly delete command, we will just going to delete one record from the table is java course here and here.

(Refer Slide Time: 16:37)

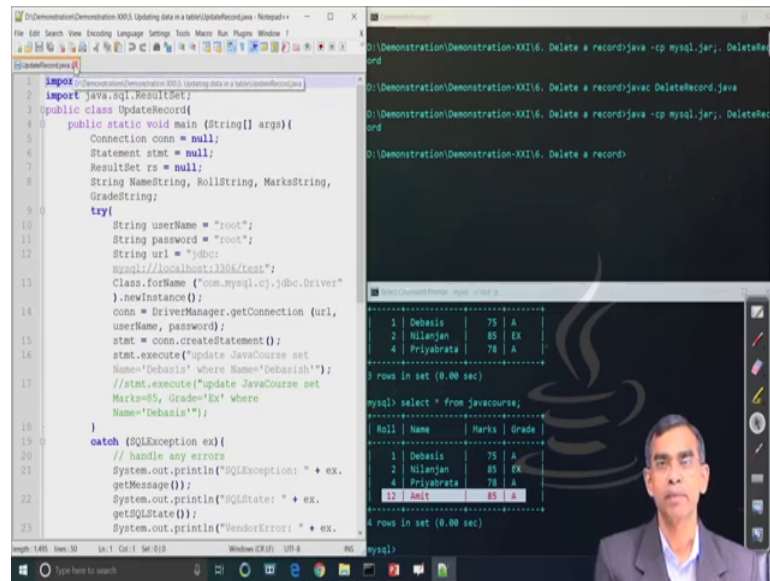


So, it is basically this is the program as we see. So, this is the usual the things it is there in we have already familiar to this thing in the last examples and this is also clear only I just changed that we have done here. So, we have just create delete from JavaCourse where marks is less than 66.

So, in the record there is some marks if it is less than 66 like the which 65 it will be deleted. Now so, let us. So, this is the only statement that we are going to do it. So, delete record. So, it is there now as you see there. Now here I am just going to change one little bit small change here. So, as you have already seen that. So, more than one SQL statement can be executed. So, those actually this is basically statement dot execute method is there.

So, I am just changing you just add one more statement dot execute instead of delete keep it delete here, then insert right insert right see roll number 7 Amit marks is no not here, next just copy statement dot execute again right. So, these are delete we are giving going to execute. So, insert right.

(Refer Slide Time: 18:11)



```
1 import java.sql.*;
2 import java.sql.ResultSet;
3 public class UpdateRecord{
4     public static void main (String[] args){
5         Connection conn = null;
6         Statement stat = null;
7         ResultSet rs = null;
8         String NameString, RollString, MarksString,
9         GradeString;
10        try{
11            String userName = "root";
12            String password = "root";
13            String url = "jdbc:
14            mysql://localhost:3306/test";
15            Class.forName ("com.mysql.cj.jdbc.Driver"
16            ).newInstance();
17            conn = DriverManager.getConnection (url,
18            userName, password);
19            stat = conn.createStatement();
20            stat.executeUpdate("update javacourse set
21            Name='Debasis' where Name='Debasis'");
22            //stat.executeUpdate("update JavaCourse set
23            Marks=85, Grade='Ex' where
24            Name='Debasis'");
25        }
26        catch (SQLException ex){
27            // handle any errors
28            System.out.println("SQLException: " + ex.
29            getMessage());
30            System.out.println("SQLState: " + ex.
31            getSQLState());
32            System.out.println("VendorError: " + ex.
33            getErrorCode());
34        }
35    }
36 }
```

```
D:\Demonstration\Demonstration-XXI\6. Delete a record>java -cp mysql.jar;. DeleteRecord
D:\Demonstration\Demonstration-XXI\6. Delete a record>java -cp mysql.jar;. DeleteRecord
D:\Demonstration\Demonstration-XXI\6. Delete a record>java -cp mysql.jar;. DeleteRecord
D:\Demonstration\Demonstration-XXI\6. Delete a record>

mysql> select * from javacourse;
+----+-----+-----+-----+
| Roll | Name   | Marks | Grade |
+----+-----+-----+-----+
| 1    | Debasis | 75    | A     |
| 2    | Nilanjan | 85    | EX    |
| 4    | Priyabrata | 78    | A     |
| 11   | amit   | 85    | EX    |
+----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Insert into right javacourse then 0 name roll value yes. So, fine, so, Amit. So, values it should be values inserting to java course values v a l u e s values 12 Amit, 85, a this one right. Now we are inserting one more record now I am just right go to here. So, we have to save this program we have changed it. Now it will execute two things; delete as well as insert also. So, yes now here you can see right. So, the new entry has been made here as you see. So, as many as SQL statement can be executed from one program, as you have seen you have tested it to two; so, it is not necessary limited to two more than two statement also can be executed from the same program itself.

So, depending on your requirement whatever the SQL statements are there, that we will be able to insert it.





So, this means that it will delete all the records from the JavaCourse. So, it is the command it is there, we are changing like delete from now delete from JavaCourse as this is the statement that we have executed from this one. So, it will basically delete all the records. So, now, record is empty.

So, there is no records. If you again insert as you see another record will be inserted or whatever it is there, and if it is select star also it will not return anything because it is it does not have any record there.

(Refer Slide Time: 21:17)

```
1 import java.sql.*;
2 import java.sql.ResultSet;
3 public class DropTable{
4     public static void main (String[] args){
5         Connection conn = null;
6         Statement stmt = null;
7         ResultSet rs = null;
8         String NameString, RollString, MarksString,
9         GradeString;
10        try{
11            String userName = "root";
12            String password = "root";
13            String url = "jdbc:
14            mysql://localhost:3306/test";
15            Class.forName ("com.mysql.cj.jdbc.Driver"
16            ).newInstance();
17            conn = DriverManager.getConnection (url,
18            userName, password);
19            stmt = conn.createStatement();
20            stmt.execute ("drop table javacourse;");
21        }
22        catch (SQLException ex){
23            // handle any errors
24            System.out.println("SQLException: " + ex.
25            getMessage());
26            System.out.println("SQLState: " + ex.
27            getSQLState());
28            System.out.println("VendorError: " + ex.
29            getErrorCode());
30        }
31        catch (Exception e){
32            System.err.println ("cannot connect to
```

```
mysql: select * from javacourse;
4 rows in set (0.00 sec)
mysql: show tables;
Empty set (0.00 sec)
mysql: select * from javacourse;
ERROR 1146 (42502): Table 'test.javacourse' doesn't exist
mysql: show tables;
Empty set (0.00 sec)
mysql:
```

So, finally, we can delete the entire table also drop table and then. So, this is the program here drop table here in this program. So, a statement can be executed and then if we show tables there after the drop table program is executed successfully.

So, drop table will be permanently delete the entire table actually and you see there is no table here. So, the selects and show tables also if you see, so, it does not have any tables in this way. Now so, what we have learnt here is that, how we can establish a connection and then after the connection is successful then we will be able to execute the SQL statement from Java application.

So, in this demonstration although we have use the localhost, but the url as you know the url specification that we have discussed during our networking discussion, the complete specification url using the protocol and then resource name and everything if we mention

properly including port number etcetera, then it will basically redirect your aggregation to that particular remote server whether it is connected through the network or it is the localhost absolutely this is the no issue you can use the same way actually.

So, in this case all though remote connection is not possible to establish at the moment although you could do that, but anyway you have done it, but the procedure is same only the url needs to be changed. And you should have the privilege to access your SQL server using your username and password that also should be ready with you, then only you will be able to establish any connection and thereby execute any SQL statement.

So, this is for the today's plan and in our next session we will discussed about how all the results those basically after executing a particular SQL statement, which will return the SQL server will be handled by our Java application itself. So, that will be discussed in our next session.

Thank you very much.