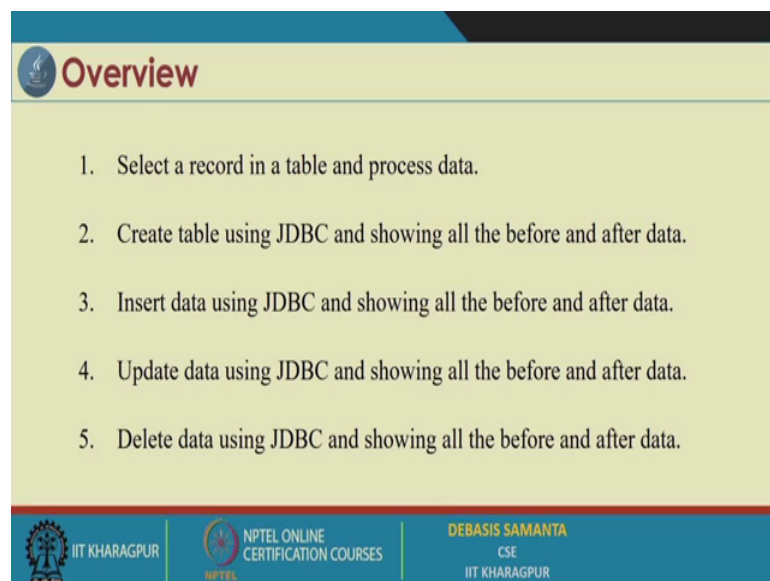


**Programming in Java**  
**Prof. Debasis Samanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 55**  
**Demonstration – XXII**

I hope you are enjoying our Demonstrations. This is our last demo and this demo is related to the lessons that we have learn so far the JDBC concept is concerned.

(Refer Slide Time: 00:31)



**Overview**

1. Select a record in a table and process data.
2. Create table using JDBC and showing all the before and after data.
3. Insert data using JDBC and showing all the before and after data.
4. Update data using JDBC and showing all the before and after data.
5. Delete data using JDBC and showing all the before and after data.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA  
CSE  
IIT KHARAGPUR

Now, to in today's demo we basically going to cover about that not exactly connecting to the SQL server on console rather program to program only. So, that from the java application itself how can issue different SQL commands and the commands can be sent to the server, server will execute the command and then the server will return the results and all the results will be processed from the application itself. So, everything is basically included in the application only we will not go out from the application.

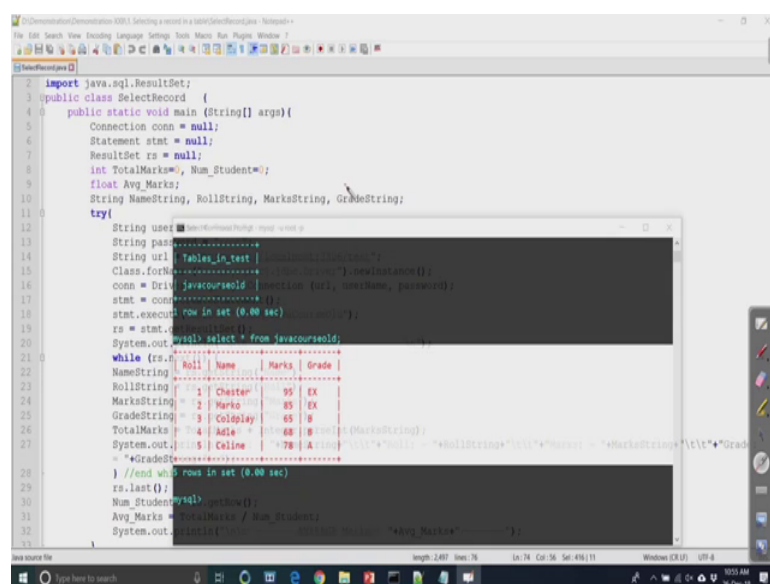
Now, so far this thing is concerned, we will see exactly how the different statements can be executed and then they can be fire to the server. And you know exactly as a process, their first step that you have to consider is that you have to first create three objects. One is the connection objects, the statement object and then result set objects. So, we have already learnt about how the connection objects can be created and then from your application how the connection can be established to a server through JDBC driver.

And then we have also learned about in our last session that how the JDBC driver can be used to have the communication that mean from application to the server some request can be issued. Today we will discuss, now we are going to discuss about the program; the statement exhibition of the different SQL statements and then getting the results and processing the results in the application environment itself.

Now, we are going to discuss about how the execution of a select statement can be done and after the select statement as you know, it will return a set of records, how all those records can be processed from the application itself. For example, display on the screen only. So, far the processing is concerned we will consider simple processing. In the first example we will consider the data will be faced from the server and the data some calculation for example, average calculation of all the records will be calculated like this one. And then we will discuss about how the table can be created and then after the creation of successful table from the program itself, we will be able to confirm our self that the table has been created successfully.

And then we will see exactly how the insert command can be executed and after the completion of the insert command execution we can ensure our self that insert of data is successfully achieved and then finally, with the repeat for the same other commands like update, delete, drop table etcetera.

(Refer Slide Time: 03:23)



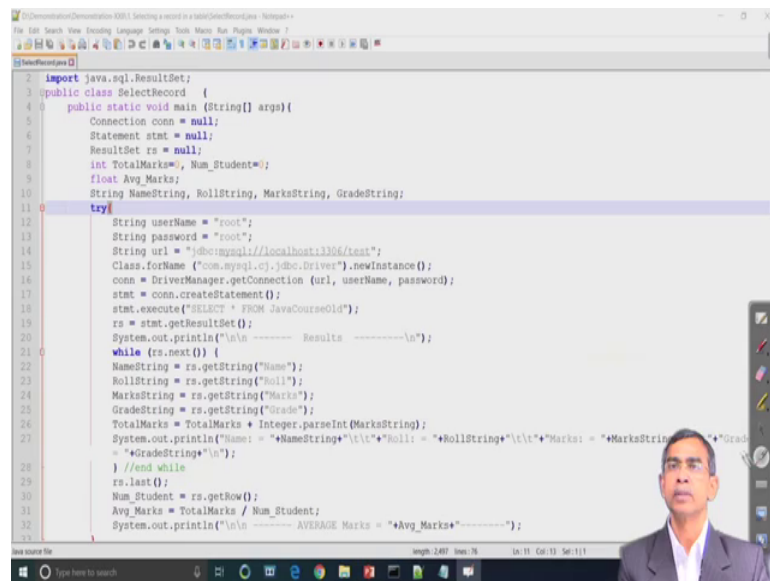
```
import java.sql.ResultSet;
public class SelectRecord {
    public static void main (String[] args){
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        int TotalMarks=0, Num_Student=0;
        float Avg_Marks;
        String NameString, RollString, MarksString, GradeString;
        try{
            String user = "root";
            String pass = "root";
            String url = "jdbc:mysql://localhost:3306/test";
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(url, user, pass);
            stmt = conn.createStatement();
            stmt.executeQuery("select * from javacourse01");
            rs = stmt.executeQuery();
            System.out.println("select * from javacourse01");
            while (rs.next()) {
                NameString = rs.getString("Name");
                RollString = rs.getString("Roll");
                MarksString = rs.getString("Marks");
                GradeString = rs.getString("Grade");
                TotalMarks = TotalMarks + Integer.parseInt(MarksString);
                System.out.println("Roll: " + RollString + " Name: " + NameString + " Marks: " + MarksString + " Grade: " + GradeString);
            }
            //end while
            rs.close();
            Num_Student = rs.getInt("Num_Student");
            Avg_Marks = TotalMarks / Num_Student;
            System.out.println("Avg_Marks: " + Avg_Marks);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

So, let us proceed for the demonstration. The first program that we are going to demo is basically how the select statement can be executed. For these things we assume that there is already a table available in the database. We have already created a table for the purpose of this demo. The Name of the table is javacourseold. Now as a yeah, so, this is the table that it is there already stored in the MySQL server and we just applied operation select statement on this.

For example, select star from javacourseold. So, basically it will execute this statement and then on the successful execution it will produce a result, those result will be faced to the application and from the application, we will be able to process it.

Now, here is the program. Let us see. As you see the program here so, few statements as you know which are obvious like how to establish the connection.

(Refer Slide Time: 04:25)



```
1  import java.sql.ResultSet;
2
3  public class SelectRecord {
4      public static void main (String[] args){
5          Connection conn = null;
6          Statement stmt = null;
7          ResultSet rs = null;
8          int TotalMarks=0, Num_Student=0;
9          float Avg_Marks;
10         String NameString, RollString, MarksString, GradeString;
11     try
12     {
13         String userName = "root";
14         String password = "root";
15         String url = "jdbc:mysql://localhost:3306/test";
16         Class.forName ("com.mysql.cj.jdbc.Driver").newInstance();
17         conn = DriverManager.getConnection (url, userName, password);
18         stmt = conn.createStatement();
19         stmt.execute ("SELECT * FROM javacourseold");
20         rs = stmt.getResultSet();
21         System.out.println("\n\n ----- Results ----- \n\n");
22         while (rs.next() {
23             NameString = rs.getString("Name");
24             RollString = rs.getString("Roll");
25             MarksString = rs.getString("Marks");
26             GradeString = rs.getString("Grade");
27             TotalMarks = TotalMarks + Integer.parseInt(MarksString);
28             System.out.println("Name = "+NameString+"\t\t"+Roll = "+RollString+"\t\t"+Marks = "+MarksString+"\t\t"+Grade
29             = "+GradeString+"\n");
30         } //end while
31         rs.last();
32         Num_Student = rs.getRow();
33         Avg_Marks = TotalMarks / Num_Student;
34         System.out.println("\n\n ----- AVERAGE Marks = "+Avg_Marks+"-----");
35     }
36 }
```

So, as we see this is basically regarding the connecting to the server. So, as we see these are the connection is there and then finally, in this statement we are executing the statement. As you see the statement that we are going to execute select star from javacourseold. So, from this table it will be there. Now if this command is executed then this will basically produce a result. So, getResultSet will receive the result of the statement object and then it will be stored in another object call the rs. So, rs is a result set object is there. So, mainly three objects as I have already told you involved; connection object, statement object and result sets. We you have already familiar in our

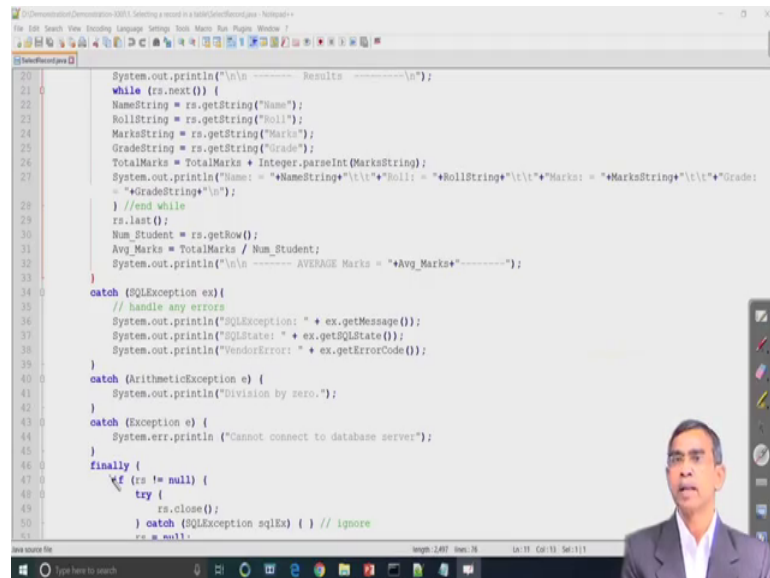
last two demonstration, we have discussed about it. Now we are going to process this rs objects.

Now, you know while rs dot next is basically written a Boolean value. So, it will basically rs objects contents a set of records that will basically receive. So, all the records it is there; that means, all rows we can say. Now rs dot next is basically till the end of the rows it will process. So, initially the rows in number of rows is not empty actually. So, it will the first row. So, it rs dot next is true actually. So, as it is true it will go to this one. So, rs dot getString is basically for the record in the javacourseold; there is a one field call Name then another is called Roll and Marks and Grade. So, for the current record the field value which has the field name Name will be obtained and it basically read as a string. So, java always read initially as a string and then we can convert into the other form that is per as per our requirement. So, is basically read the string for the field Name the value and we store is at a string.

So, similarly NameString, RollString, MarkString, GradeStrings are the four different field values for a record; namely, Name, Roll, Marks and Grade and then we then we convert all the values. For example, we are our objective is that once you read the marks we have to calculate the TotalMarks sum of all the marks for all the records and then finally, divide by the total number of records and then average will be calculated. So, this basically is process is basically TotalMarks initially 0 and then Integer dot parseInt MarkString because it is initially as a string and use this conversation we can convert the integer value and then TotalMarks will be added. So, these basically reach a record each time and then read the I mean at the values marks value to the TotalMarks and finally, the total of all marks will be obtained here and here actually next is that.

So, System dot out dot println Name: NameString Roll RollString and then Marks is basically printed here. So, MarkString is there. And then next is basically rs dot last means is basically when you reach to the last object, so, rs dot last is basically there. And then number of students is basically can be obtained from the rs object itself by calling the method getRow method. So, rs dot getRow. So, it basically says the total number of records that we have read it. So, TotalMarks divided by this total number of records it basically gives you the average calculation.

(Refer Slide Time: 08:03)



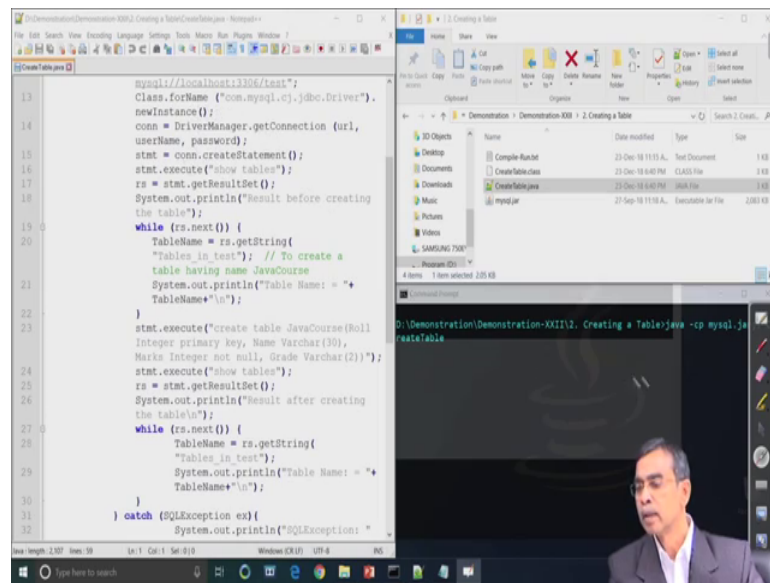
```
20 System.out.println("\n\n ----- Results ----- \n\n");
21 while (rs.next()) {
22     NameString = rs.getString("Name");
23     RollString = rs.getString("Roll");
24     MarksString = rs.getString("Marks");
25     GradeString = rs.getString("Grade");
26     TotalMarks = TotalMarks + Integer.parseInt(MarksString);
27     System.out.println("Name: " + NameString + "\t\t" + Roll: " + RollString + "\t\t" + Marks: " + MarksString + "\t\t" + Grade:
28     + GradeString + "\n");
29 } //end while
30 rs.last();
31 Num_Student = rs.getRow();
32 Avg_Marks = TotalMarks / Num_Student;
33 System.out.println("\n\n ----- AVERAGE Marks = " + Avg_Marks + "-----");
34 }
35 catch (SQLException ex) {
36     // handle any errors
37     System.out.println("SQLException: " + ex.getMessage());
38     System.out.println("SQLState: " + ex.getSQLState());
39     System.out.println("VendorError: " + ex.getErrorCode());
40 }
41 catch (ArithmeticException e) {
42     System.out.println("Division by zero.");
43 }
44 catch (Exception e) {
45     System.err.println("Cannot connect to database server");
46 }
47 finally {
48     if (rs != null) {
49         try {
50             rs.close();
51         } catch (SQLException sqlEx) { } // ignore
52         rs = null;
53     }
```

As we see the average calculation is the average is basically calculated here and then we print this average. And the rest of the thing is for the try, catch, block and everything this is the customary try, catch, block we have to maintain and so that the program can be executed in a reliable and robust manner.

So, these basically tell you that how the result set object can be processed. So, in this session we will mainly emphasize of the processing of the results such that the or alternatively the all result that will obtain after the execution of SQL statement from the application itself ok. So, this is the demonstration, hope you have you are able to understand it.

Our next demonstrations little bit simpler. We will see exactly how the different other SQL type of statement. We will consider only execution of basic SQL statements and this will be enough for the begin for the learning purpose only. So, in this case, we will consider one table already available.

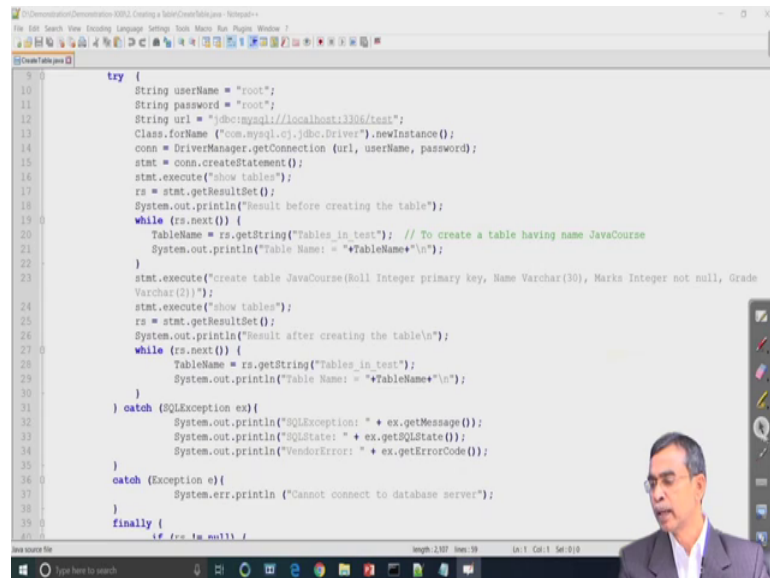
(Refer Slide Time: 08:59)



```
13  @SuppressWarnings("unchecked")
14  Class.forName ("com.mysql.cj.jdbc.Driver").
15  newInstance();
16  conn = DriverManager.getConnection (url,
17  userName, password);
18  stmt = conn.createStatement();
19  stmt.execute ("show tables");
20  rs = stmt.getResultSet();
21  System.out.println ("Result before creating
22  the table");
23  while (rs.next()) {
24      TableName = rs.getString(
25      "Tables_in_test"); // To create a
26      table having name JavaCourse
27      System.out.println ("Table Name = "+
28      TableName+"\n");
29  }
30  stmt.execute ("create table JavaCourse(Roll
31  Integer primary key, Name Varchar(30),
32  Marks Integer not null, Grade Varchar(2))");
33  stmt.execute ("show tables");
34  rs = stmt.getResultSet();
35  System.out.println ("Result after creating
36  the table");
37  while (rs.next()) {
38      TableName = rs.getString(
39      "Tables_in_test");
40      System.out.println ("Table Name = "+
41      TableName+"\n");
42  }
43  } catch (SQLException ex){
44      System.out.println ("SQLException: "
```

Now, we will consider the create table right. So, we are going to consider how a new table can be created and then so, we first see first from the application itself, we will exactly check that what are the different tables are available there in the database server. So, in that case we have to issue the command. So, tables and then once the table is there we can create a new table. So, we just create a new table. The name of the new table maybe say java course and then after in creating a new table we will again see the confirmation about that new table has been created successfully. So, these are the steps that we will follow one by one, but from the java application itself ok. So, here is a program as we see go little bit ok.

(Refer Slide Time: 09:57)



```
try {
    String userName = "root";
    String password = "root";
    String url = "jdbc:mysql://localhost:3306/test";
    Class.forName("com.mysql.cj.jdbc.Driver").newInstance();
    conn = DriverManager.getConnection(url, userName, password);
    stmt = conn.createStatement();
    stmt.execute("show tables");
    rs = stmt.getResultSet();
    System.out.println("Result before creating the table");
    while (rs.next()) {
        TableName = rs.getString("Tables_in_test"); // To create a table having name JavaCourse
        System.out.println("Table Name = "+TableName+"\n");
    }
    stmt.execute("create table JavaCourse(Roll Integer primary key, Name Varchar(30), Marks Integer not null, Grade
    Varchar(2));");
    stmt.execute("show tables");
    rs = stmt.getResultSet();
    System.out.println("Result after creating the table\n");
    while (rs.next()) {
        TableName = rs.getString("Tables_in_test");
        System.out.println("Table Name = "+TableName+"\n");
    }
} catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
} catch (Exception e) {
    System.err.println ("Cannot connect to database server");
}
finally {
    if (rs != null) {

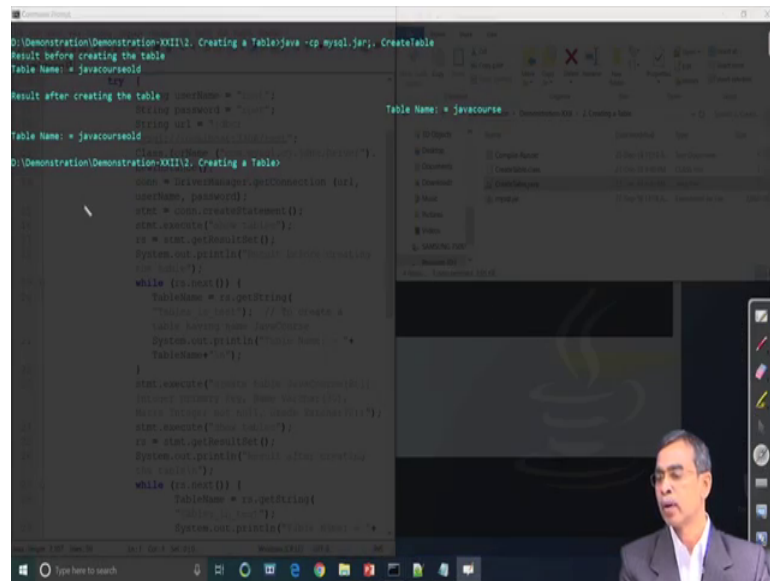
```

So, first few part of the program. So, first few part of the program is obvious is basically connecting to the server using JDBC. So, loading JDBC connecting all these things are there and then next is you see this is a statement that we are going to execute, show table command. Now so, the result set will be obtained. Basically show table will see you all the tables those are there in the current active database in this case may be test database. So, rs will basically get the result sets includes all the tables those are there in the database right now.

Now, the prompt is that results before creating the table and here is basically processing the rs objects is basically doing nothing is basically getting each TableName from the result sets and printing it on the screen that is all. So, it will basically do these things and then after the next thing we execute another statement create table JavaCourse.

And as you know this is the full syntax about how to create a table new table, in fact, and add into the database. So, is a statement execute and after this create a new table is created. The show table, it is basically send the previous code right after getting all the tables right; it will basically display all the tables. So, this is the part that ok, it will show table before the creating and then in create a new table and after the creation it will show again all the tables. Now, we will see the results that can be obtained from the java application. It is a as you see here from the java application we are able to see it.

(Refer Slide Time: 11:43)



```
D:\Demonstration\Demonstration-XXXX\1. Creating a Table>java -cp mysql.jar, CreateTable
Result before creating the table
Table Name = javacourseold

Result after creating the table
Table Name = javacourseold

D:\Demonstration\Demonstration-XXXX\1. Creating a Table>
class = DriverManager.getConnection(url,
username, password);
stmt = stmt.createStatement();
stmt.executeUpdate("create table
if not exists javacourseold
(course varchar(255),
password varchar(255))");
System.out.println("Table Name = " +
tableName);
while (rs.next()) {
tableName = rs.getString(
"table_name"); // to create a
table having name javacourseold
System.out.println("Table Name = " +
tableName);
}
stmt.executeUpdate("insert into javacourseold
(course, password)
values ('javacourseold',
'javacourseold')");
rs = stmt.executeQuery("select * from
javacourseold");
System.out.println("Table Name = " +
tableName);
while (rs.next()) {
tableName = rs.getString(
"table_name");
System.out.println("Table Name = " +
tableName);
}
```

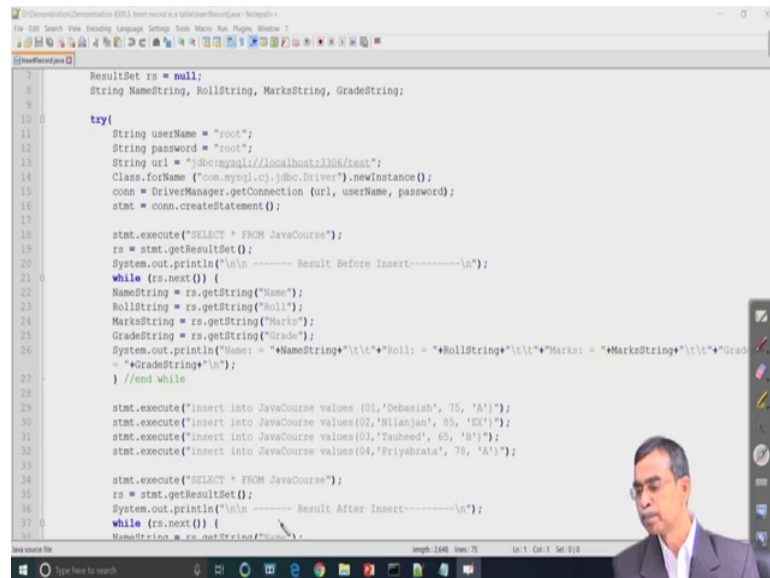
Now, if we run this program, the execution will flow look like this. So, this is the program and then ok. So, we just Table Name is a javacourse.

So these basically shows initially only one table is present there called the javacourseold. Now results after creating the table as you see after creating the table. So, we run this code again. After creating the table, it basically shows that javacourseold is there and then javacourse table is there. So, two tables are here; javacourseold the previous and then new one, it is there and then is basically is that basically this is the output actually as we see here. So, the statements have been executed. After the execution we can see the results and that results also through the JDBC itself.

Now, let us come to the next one another example where we are we can see exactly how we can insert a record into an already existing table. We will consider inserting record into the table say javacourseold. First we show before execution what is the content in the table and then we will again issue the insert statement and then after the insert statement the current data that is there.



(Refer Slide Time: 13:09)

A screenshot of a Java IDE window titled 'NewJDBC.java'. The code defines a ResultSet and several String variables for database connection. It includes a try block with database connection logic, a SELECT query execution, and a while loop to iterate through the result set and print record details. The IDE interface includes a menu bar, toolbar, and a small inset video of a man in the bottom right corner.

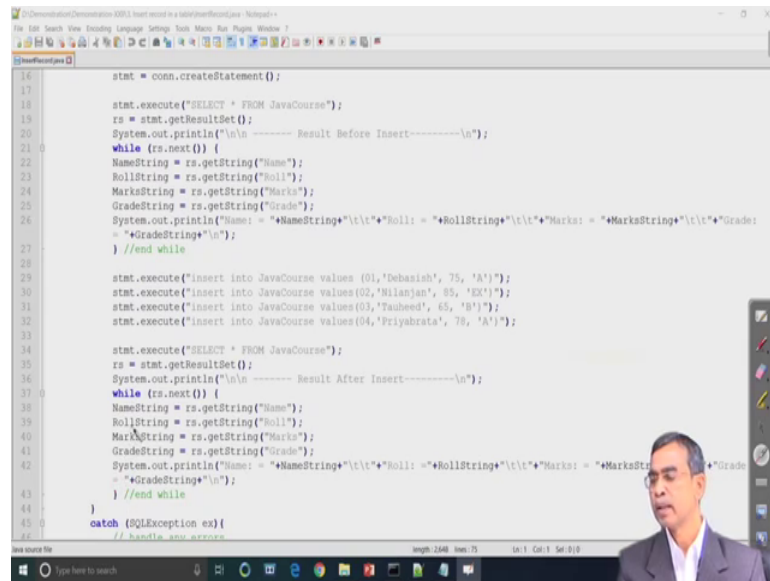
```
7 ResultSet rs = null;
8 String NameString, RollString, MarksString, GradeString;
9
10 try
11 {
12     String userName = "root";
13     String password = "root";
14     String url = "jdbc:mysql://localhost:3306/test";
15     Class.forName ("com.mysql.jdbc.Driver").newInstance();
16     conn = DriverManager.getConnection (url, userName, password);
17     stmt = conn.createStatement();
18     stmt.execute("SELECT * FROM JavaCourse");
19     rs = stmt.getResultSet();
20     System.out.println("\n\n ----- Result Before Insert-----\n\n");
21     while (rs.next()) {
22         NameString = rs.getString("Name");
23         RollString = rs.getString("Roll");
24         MarksString = rs.getString("Marks");
25         GradeString = rs.getString("Grade");
26         System.out.println("Name = "+NameString+"\t\t*Roll = "+RollString+"\t\t*Marks = "+MarksString+"\t\t*Grade = "+GradeString+"\n");
27     } //end while
28
29     stmt.execute("insert into JavaCourse values (01,'Debanish', 75, 'A')");
30     stmt.execute("insert into JavaCourse values (02,'Nilanjan', 85, 'B')");
31     stmt.execute("insert into JavaCourse values (03,'Tauheed', 65, 'B')");
32     stmt.execute("insert into JavaCourse values (04,'Priyabrata', 70, 'A')");
33
34     stmt.execute("SELECT * FROM JavaCourse");
35     rs = stmt.getResultSet();
36     System.out.println("\n\n ----- Result After Insert-----\n\n");
37     while (rs.next()) {
38         NameString = rs.getString("Name");
```

Now here is the course. Here is the obvious course for creating a connection and then loading the JDBC driver and then, here is basically we creating the statement objects and this is the first statement that we are going to create is the statement star from java course. That means, before inserting what is the content it is there, the result sets and this result set will basically display whatever the records that it returns to us and we will see exactly before create before inserting a data, how what are the different records are there.

And then next we fire few statements regarding inserting one record at a time. So, here is inserting first record then second record third record and fourth record. So, if the table is empty then it will show only these four records. If table contents already data so, then all these four newly inserted record will be added after the existing data and then this statement execute select star from JavaCourse.

Basically, same is basically select all the records and get it the result set will receive it and then finally, the result set will here is a while loop the result set will be process.

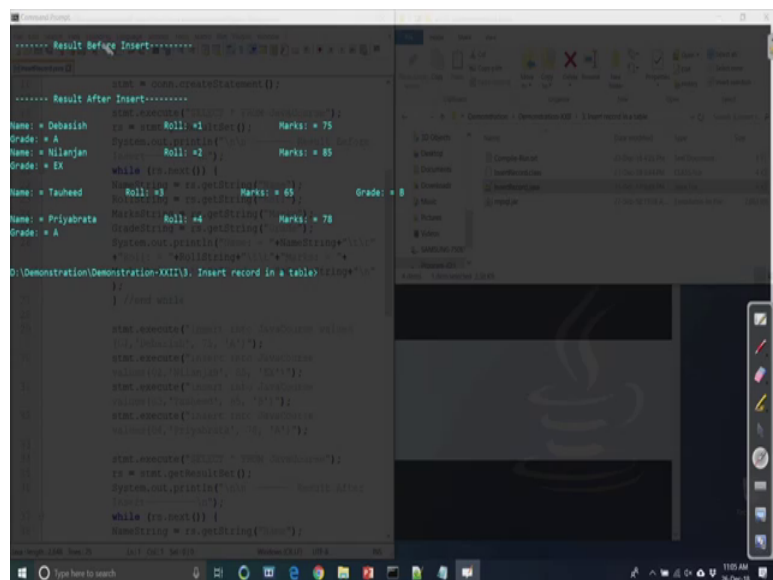
(Refer Slide Time: 14:25)



```
16 stat = conn.createStatement();
17
18 stat.execute("SELECT * FROM JavaCourse");
19 rs = stat.getResultSet();
20 System.out.println("\n\n ----- Result Before Insert-----\n\n");
21 while (rs.next()) {
22     NameString = rs.getString("Name");
23     RollString = rs.getString("Roll");
24     MarksString = rs.getString("Marks");
25     GradeString = rs.getString("Grade");
26     System.out.println("Name: = "+NameString+"\t\t"+Roll: = "+RollString+"\t\t"+Marks: = "+MarksString+"\t\t"+Grade:
27     = "+GradeString+"\n");
28 } //end while
29
30 stat.execute("insert into JavaCourse values (01,'Debasish', 75, 'A')");
31 stat.execute("insert into JavaCourse values (02,'Nilanjan', 85, 'EX')");
32 stat.execute("insert into JavaCourse values (03,'Tauheed', 65, 'B')");
33 stat.execute("insert into JavaCourse values (04,'Priyabrata', 78, 'A')");
34
35 stat.execute("SELECT * FROM JavaCourse");
36 rs = stat.getResultSet();
37 System.out.println("\n\n ----- Result After Insert-----\n\n");
38 while (rs.next()) {
39     NameString = rs.getString("Name");
40     RollString = rs.getString("Roll");
41     MarksString = rs.getString("Marks");
42     GradeString = rs.getString("Grade");
43     System.out.println("Name: = "+NameString+"\t\t"+Roll: = "+RollString+"\t\t"+Marks: = "+MarksString+"\t\t"+Grade:
44     = "+GradeString+"\n");
45 } //end while
46
47 catch (SQLException ex){
48     // handle any errors
49 }
```

And then result set will be processed and it will display all the records that we have receive it. So, this is the program. Now, let us come to the execution of this program, so that you can understand how it is working. So, insert record here.

(Refer Slide Time: 14:41)



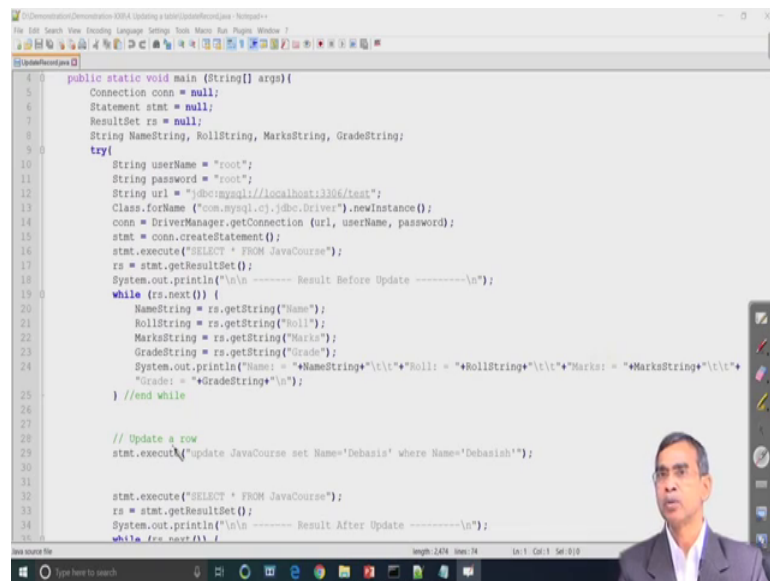
```
----- Result Before Insert-----
----- Result After Insert-----
Name: = Debasish
Grade: = A
Roll: = 01
Marks: = 75
Name: = Nilanjan
Grade: = EX
Roll: = 02
Marks: = 85
Name: = Tauheed
Grade: = B
Roll: = 03
Marks: = 65
Name: = Priyabrata
Grade: = A
Roll: = 04
Marks: = 78
```

As you see here Results Before Insert. So, these basically the table does not contain any record. So, it basically empty. Then Result After Insert, as we see this is the record first record we have in inserted this is the second record third record and fourth record. So, the results have been received and then they have been displayed on the screen; that

means, we can confirm that the results are now available at the application in itself. So, this is the example that is a insert, opposite to the insert our next example is update example.

So, we will just going to update a particular record. As it is always there we will execute the command that what are the records are there and then we will update a record and then after the updation we will see whether updation is reflected or not.

(Refer Slide Time: 15:31)

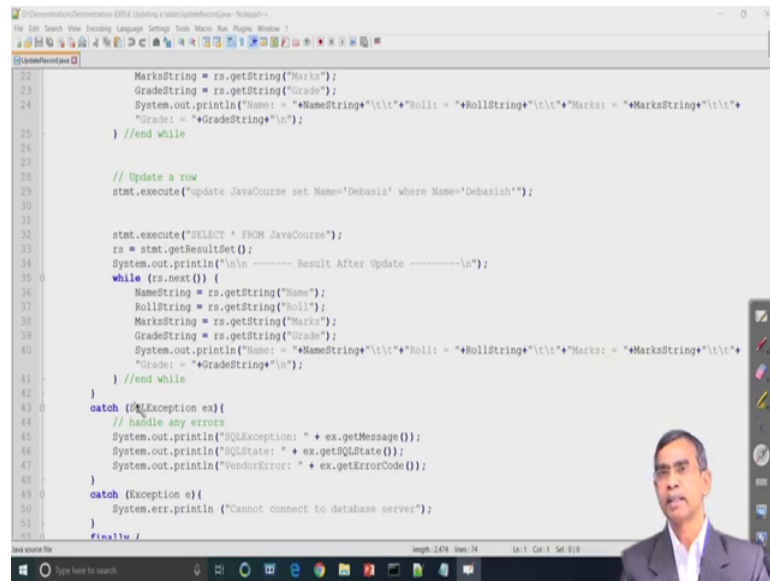


```
4 public static void main (String[] args){
5     Connection conn = null;
6     Statement stmt = null;
7     ResultSet rs = null;
8     String NameString, RollString, MarksString, GradeString;
9     try{
10        String userName = "root";
11        String password = "root";
12        String url = "jdbc:mysql://localhost:3306/test";
13        Class.forName ("com.mysql.cj.jdbc.Driver").newInstance();
14        conn = DriverManager.getConnection (url, userName, password);
15        stmt = conn.createStatement();
16        stmt.execute("SELECT * FROM JavaCourse");
17        rs = stmt.getResultSet();
18        System.out.println("\n ----- Result Before Update ----- \n");
19        while (rs.next() ) {
20            NameString = rs.getString("Name");
21            RollString = rs.getString("Roll");
22            MarksString = rs.getString("Marks");
23            GradeString = rs.getString("Grade");
24            System.out.println("Name: "+NameString+"\t\t"+Roll: "+RollString+"\t\t"+Marks: "+MarksString+"\t\t"+
25                "Grade: "+GradeString+"\n");
26        } //end while
27
28        // Update a row
29        stmt.executeUpdate("update JavaCourse set Name='Debasis' where Name='Debasis'");
30
31        stmt.execute("SELECT * FROM JavaCourse");
32        rs = stmt.getResultSet();
33        System.out.println("\n ----- Result After Update ----- \n");
34        while (rs.next() ) {
```

So, we will receive the results after the updation and we will display the updated record on the screen. So, this is the program flow and here is basically creating the connection and then executing the statement that what are the records are there. So, the before updation and then it basically show all the reports and here is the command by which we execute the update. Here we see Update JavaCourse set Name Debasis where Name is Debasis. So, if there is any record whose value is basically Debasis as a Name field and it will basically change into this form. So, this is the next and this is the previous one.

So, it is the updation is there. Only one field value of a record will be updated here. If there are many records containing the same thing, all records in fact, will be updated by this commend because it will updated it will scan entire data base tables and then finally, whichever the condition is match it will basically applied to this one, bind to that, records and then do it.

(Refer Slide Time: 16:45)

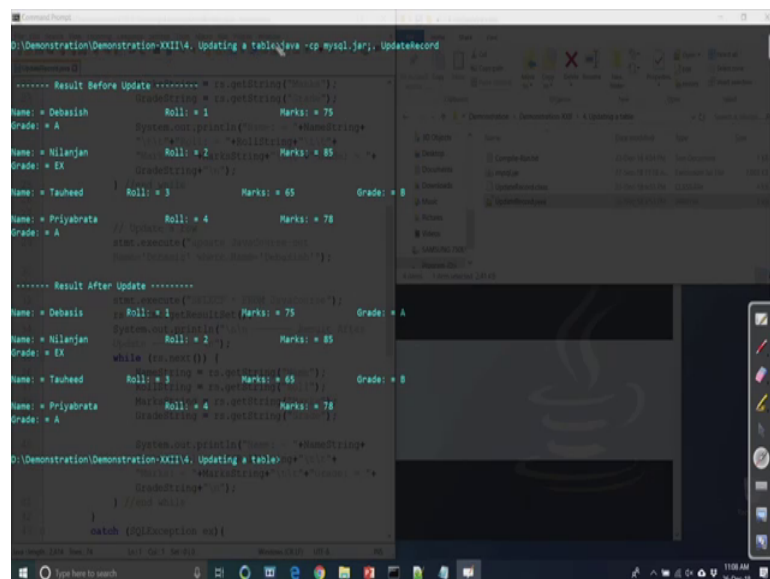


```
22 MarksString = rs.getString("Marks");
23 GradeString = rs.getString("Grade");
24 System.out.println("Name: " + NameString + "\t\t" + Roll: = " + RollString + "\t\t" + Marks: = " + MarksString + "\t\t" +
25 "Grade: = " + GradeString + "\n");
26 } //end while
27
28 // Update a row
29 stat.execute("update JavaCourse set Name='Debasish' where Name='Debasish'");
30
31
32 stat.execute("SELECT * FROM JavaCourse");
33 rs = stat.getResultSet();
34 System.out.println("\n\n ----- Result After Update ----- \n\n");
35 while (rs.next()) {
36 NameString = rs.getString("Name");
37 RollString = rs.getString("Roll");
38 MarksString = rs.getString("Marks");
39 GradeString = rs.getString("Grade");
40 System.out.println("Name: " + NameString + "\t\t" + Roll: = " + RollString + "\t\t" + Marks: = " + MarksString + "\t\t" +
41 "Grade: = " + GradeString + "\n");
42 } //end while
43
44 catch (SQLException ex){
45 // handle any errors
46 System.out.println("SQLException: " + ex.getMessage());
47 System.out.println("SQLState: " + ex.getSQLState());
48 System.out.println("VendorError: " + ex.getErrorCode());
49 }
50 catch (Exception e){
51 System.err.println ("Cannot connect to database server");
52 }
53 finally {
```

And then find and our next step is basically we are going to we are going to again execute that select command as we see here select command Select star from JavaCourse. It basically after the updation, it will receive the results and then all the results are displayed on the screen and then these are the necessary try, catch formalities.

Now, let us have the execution of this program, it is called the update record. This is the name of the program that we have given. We are running this program update record right.

(Refer Slide Time: 17:15)



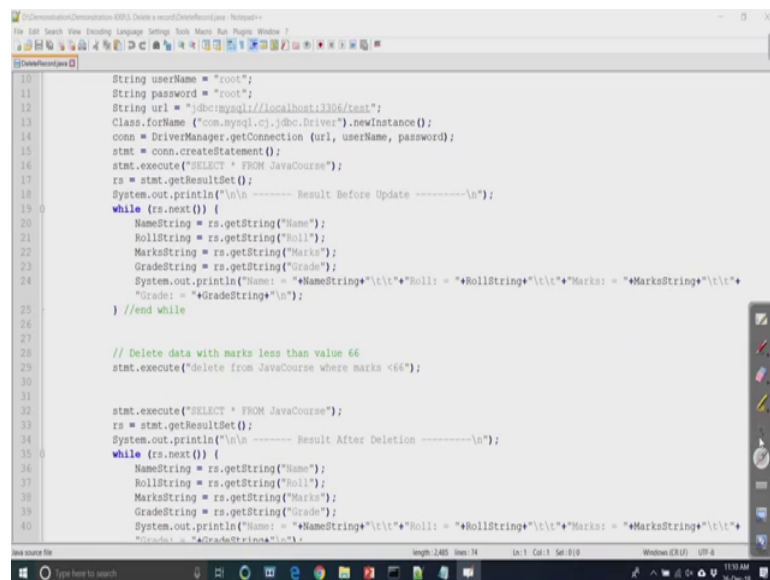
```
..... Result Before Update .....
Name: = Debasish Roll: = 1 Marks: = 75 Grade: = A
Name: = Nilanjan Roll: = 2 Marks: = 85 Grade: = EX
Name: = Tauheed Roll: = 3 Marks: = 65 Grade: = B
Name: = Priyabrata Roll: = 4 Marks: = 78 Grade: = A

..... Result After Update .....
Name: = Debasish Roll: = 1 Marks: = 75 Grade: = A
Name: = Nilanjan Roll: = 2 Marks: = 85 Grade: = EX
Name: = Tauheed Roll: = 3 Marks: = 65 Grade: = B
Name: = Priyabrata Roll: = 4 Marks: = 78 Grade: = A
```

And this will show the records before the update. Now here is the execution the total output of the execution. Here you see yeah. So, here as we see start from here, so, this is the output started from the screen here after executing the update record. So, Results Before Update as we see Debasis, Nilanjan, Tauheed, Priyobrate; four records are there.

Now, we have executed update command. Then results after update command as you see initially, Debasis has been updated Debasis and it is only applicable to this record. So, it is only one updation is there and then the result is there. So, now, we can understand that. So, everything is possible from your program itself right and then all the SQL statement either execution then after getting result after the result is available processing all those results from the one point of node only.

(Refer Slide Time: 18:17)



```
10 String userName = "root";
11 String password = "root";
12 String url = "jdbc:mysql://localhost:3306/test";
13 Class.forName("com.mysql.jdbc.Driver").newInstance();
14 conn = DriverManager.getConnection(url, userName, password);
15 stmt = conn.createStatement();
16 stmt.execute("SELECT * FROM JavaCourse");
17 rs = stmt.getResultSet();
18 System.out.println("\n\n ----- Result Before Update ----- \n\n");
19 while (rs.next()) {
20     NameString = rs.getString("Name");
21     RollString = rs.getString("Roll");
22     MarksString = rs.getString("Marks");
23     GradeString = rs.getString("Grade");
24     System.out.println("Name = " + NameString + "\t\t" + "Roll = " + RollString + "\t\t" + "Marks = " + MarksString + "\t\t" +
25         "Grade = " + GradeString + "\n");
26 } //end while
27
28 // Delete data with marks less than value 66
29 stmt.execute("Delete from JavaCourse where marks <66");
30
31
32 stmt.execute("SELECT * FROM JavaCourse");
33 rs = stmt.getResultSet();
34 System.out.println("\n\n ----- Result After Deletion ----- \n\n");
35 while (rs.next()) {
36     NameString = rs.getString("Name");
37     RollString = rs.getString("Roll");
38     MarksString = rs.getString("Marks");
39     GradeString = rs.getString("Grade");
40     System.out.println("Name = " + NameString + "\t\t" + "Roll = " + RollString + "\t\t" + "Marks = " + MarksString + "\t\t" +
41         "Grade = " + GradeString + "\n");
42 }
```

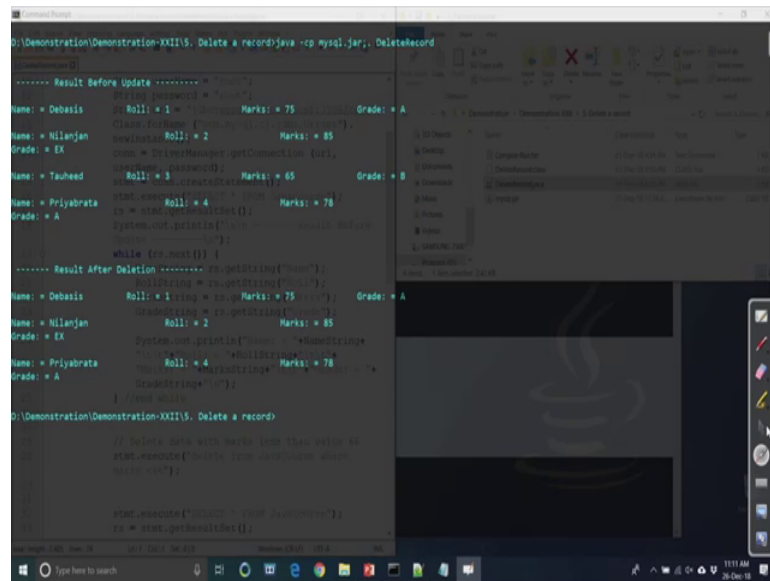
Now, our next example; this is the example ok, so, for the delete is common. So, that whether delete command can be also successfully executed or not. So, first in the same manner we will first see all the tables; accountants, first records and then we will issue a delete command and after the delete command we will exactly see the results.

So, here is the program flow, creating the connection. Then we execute the first statement; Select star from JavaCourse is basically receive all the records and then display all the results on the screen after the execution of the statement. Then we execute the delete command; delete from java course where marks less than 66. If there are any record records or there is any there is a record whose value of the field marks is less than

66 will be deleted from the record and after the execution of this statement we will be able to again see the records after the deletion.

So, we have to issue another command Select star from JavaCourse, it will get all the results and then there result will be printed on the screen. The result will be printed on the screen and then rest of the program is basically try catch. Now, let us run this program, this program is a delete record.

(Refer Slide Time: 19:41)

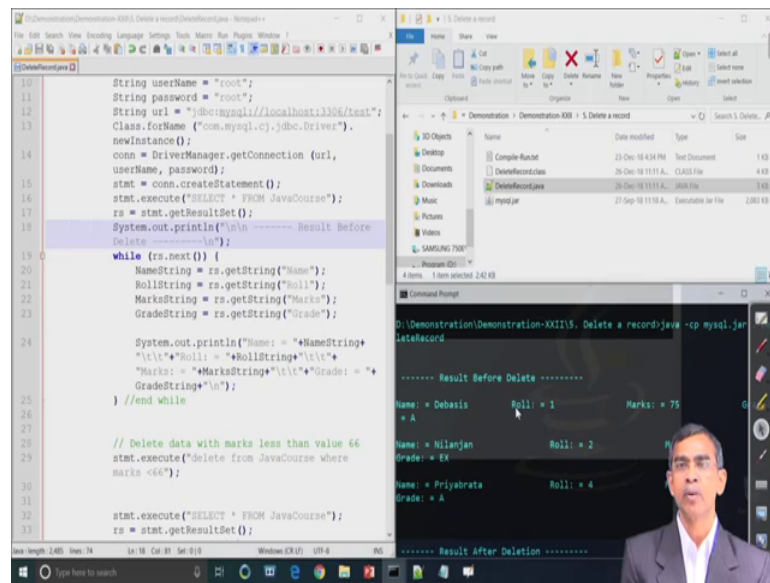


```
D:\Demonstration\Demonstration-XXII\5. Delete a record>java -cp mysql.jar;. DeleteRecord
..... Result Before Update .....
Name: = Debasis      Roll: = 1      Marks: = 75      Grade: = A
Name: = Nilanjan    Roll: = 2      Marks: = 85      Grade: = EX
Name: = Tauheed     Roll: = 3      Marks: = 65      Grade: = B
Name: = Priyabrata  Roll: = 4      Marks: = 78      Grade: = A
..... Result After Deletion .....
Name: = Debasis      Roll: = 1      Marks: = 75      Grade: = A
Name: = Nilanjan    Roll: = 2      Marks: = 85      Grade: = EX
Name: = Priyabrata  Roll: = 4      Marks: = 78      Grade: = A
D:\Demonstration\Demonstration-XXII\5. Delete a record>
```

And this is the execution results written results. As we see delete record is executed here. So, Results Before Delete ok; result before delete any way. So, there is a mistakes there in the program itself is updated. Anyway, so, you read add delete and then these are the four records as we see. Results after deletion as we see here actually the marks is 75, 85, 65, 78.

So, if the execution is correct then the third records should be deleted from the table. As you see the first record is there, second record is there, fourth record is there and third record is no more. So, it has been deleted. So, we have learned about how the result set object is there.

(Refer Slide Time: 20:25)



```
String userName = "root";
String password = "root";
String url = "jdbc:mysql://localhost:3306/test";
Class.forName("com.mysql.cj.jdbc.Driver").
newInstance();
conn = DriverManager.getConnection(url,
userName, password);
stat = conn.createStatement();
stat.execute("SELECT * FROM JavaCourse");
rs = stat.getResultSet();
System.out.println("\n\n ----- Result Before
Delete -----");
while (rs.next()) {
    NameString = rs.getString("Name");
    RollString = rs.getString("Roll");
    MarksString = rs.getString("Marks");
    GradeString = rs.getString("Grade");

    System.out.println("Name: "+NameString+
"\t\t\t"+RollString+"\t\t\t"+
"\t\t\t"+MarksString+"\t\t\t"+GradeString+"\n");
} //end while

// Delete data with marks less than value 66
stat.execute("delete from JavaCourse where
marks <66");

stat.execute("SELECT * FROM JavaCourse");
rs = stat.getResultSet();
```

----- Result Before Delete -----

Name: = Debasish	Roll: = 1	Marks: = 75	Grade: = A
Name: = Nilanjan	Roll: = 2	Marks: = 65	Grade: = EX
Name: = Priyabrata	Roll: = 4	Marks: = 85	Grade: = A

----- Result After Deletion -----

Now, so far the JDBC programming is concerned there are few steps that we have learnt about and all the steps that we have done it and then also the sufficient demonstration we have given starting from how the MySQL server can be installed.

And then how the MySQL server can be used using the console based mySQL server accessing. Then connection to the server using the JDBC and how the JDBC driver also can be downloaded and it can be archived and then the command can be executed. And then we can after the execution of command how the results can be obtained or receives into the program and the program can process these things. So, this is the demo about the our last third say part of the demo, so for the JDBC is concerned and this is the end of our demonstration session as well as all the demonstration.

In this course we have covered around 20 22 demonstration and then other around 28 lecture slice in other. So, basically almost 50 percent demonstration followed by the 50 percent theoretical discussion, so that all the theoretically leaned concept can be complimented through the demonstration.

I hope you have enjoyed the course and the demonstration. In our next section, we will discuss about some java projects that projects can be handled by you as a single team member or as a jointly with many team people involved. Now we will discuss the projects and then we will basically see exactly how far you have leant and then whether we can solve any real life application sort of things in this ok.

Thank you very much.