Experiment No.___ *Date___/___/2022*

TITLE OF EXPERIMENT: - A JAVA Program to insert and retrieve the data from the database using JDBC.

DIVISION:_________________________ BRANCH: __________________

BATCH:____________________________ ROLL NO.: _________________

PERFORMED ON DATE: ____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 4

Aim: Write a program to insert and retrieve the data from the database using JDBC.

Software:

1.	Command prompt
2.	JDK 8
3.	Eclipse neon3
4.	Mysql installer 8.0.28
5.	Mysql connector jar file

Theory:

A) Java Database Connectivity with MySQL

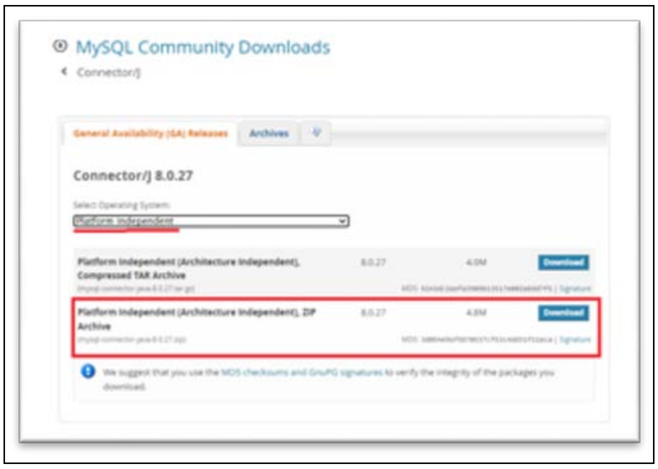
In Java, we can connect to our database (MySQL) with JDBC(Java Database Connectivity) through the Java code. JDBC is one of the standard APIs for database connectivity, using it we can easily run our query, statement, and also fetch data from the database.

Prerequisite to understand Java Database Connectivity with MySQL:-

1. You have MySQL on your System.
2. You have JDK on your System.
3. To set up the connectivity user should have MySQL Connector to the Java (JAR file), the 'JAR' file must be in classpath while compiling and running the code of JDBC.

Steps to download MySQL Connector:

- Search for MySQL community downloads.
- Then, go to the **Connector/J**.
- Then, select the Operating System **platform-independent**.
- Then, download the zip file **Platform Independent (Architecture Independent), ZIP Archive**.


MySQL Community Downloads

Connector/J

General Availability (GA) Releases | Archives

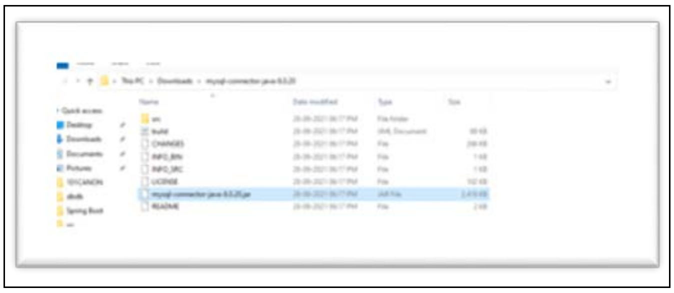
Connector/J 8.0.27

Select Operating System:
Platform Independent

Platform Independent (Architecture Independent), Compressed TAR Archive [illegible]	8.0.27	4.0M	Download
Platform Independent (Architecture Independent), ZIP Archive [illegible]	8.0.27	4.8M	Download

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

- Then, extract the zip file.
- Get the **mysql-connector-java-8.0.20.jar** file from the folder.


This PC > Downloads > mysql-connector-java-8.0.20

Name	Date modified	Type	Size
src	[illegible]	File folder	
build	[illegible]	XML Document	[illegible]
CHANGES	[illegible]	File	[illegible]
INFO_BIN	[illegible]	File	1 KB
INFO_SRC	[illegible]	File	1 KB
LICENSE	[illegible]	File	[illegible]
mysql-connector-java-8.0.20.jar	[illegible]	JAR File	[illegible]
README	[illegible]	File	2 KB

<u>Setting up Database Connectivity with MySQL using JDBC code</u>

Users have to follow the following steps:-

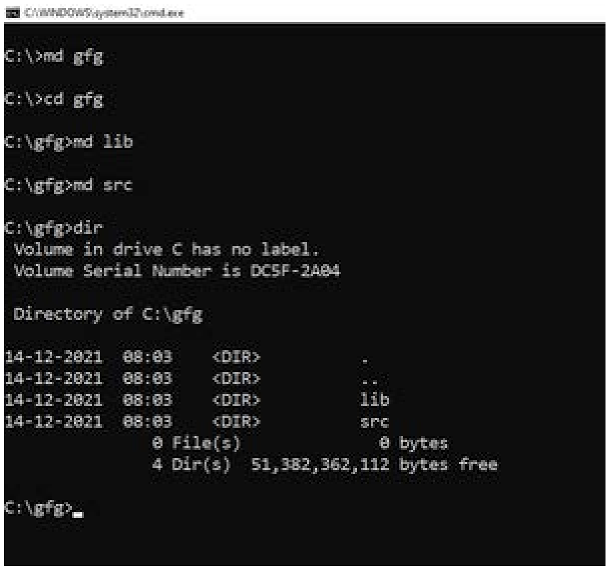
1. Users have to create a database in MySQL (for example let the name of the database be 'mydb').
2. create a table in that database.
Example:
create table designation

```
(
    code int primary key auto_increment,
    title char(35) not null unique
);
```

this is MySQL code for creating a table.

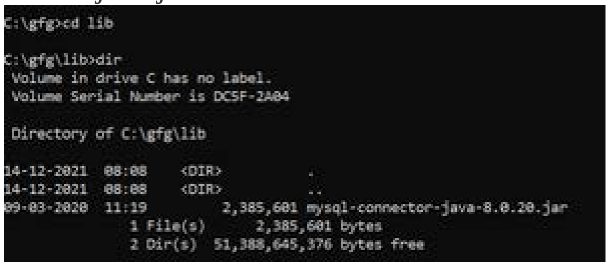
3. Now, we want to access the data of this table using Java database connectivity.

- create a directory in your main drive (named gfg).
- now, inside gfg created two more directories one named as 'src' and the other 'lib'.



```
C:\WINDOWS\system32\cmd.exe

C:\>md gfg

C:\>cd gfg

C:\gfg>md lib

C:\gfg>md src

C:\gfg>dir
 Volume in drive C has no label.
 Volume Serial Number is DC5F-2A04

 Directory of C:\gfg

14-12-2021  08:03    <DIR>          .
14-12-2021  08:03    <DIR>          ..
14-12-2021  08:03    <DIR>          lib
14-12-2021  08:03    <DIR>          src
               0 File(s)              0 bytes
               4 Dir(s)  51,382,362,112 bytes free

C:\gfg>_
```

- Put the MySQL connector java jar file in the lib folder.



```
C:\gfg>cd lib

C:\gfg\lib>dir
 Volume in drive C has no label.
 Volume Serial Number is DC5F-2A04

 Directory of C:\gfg\lib

14-12-2021  08:08    <DIR>          .
14-12-2021  08:08    <DIR>          ..
09-03-2020  11:19         2,385,601 mysql-connector-java-8.0.20.jar
               1 File(s)      2,385,601 bytes
               2 Dir(s)  51,388,645,376 bytes free
```

4. We will write connectivity code in the src folder, to write connectivity code user must know the following information:

- **Driver class**:- The driver class for connectivity of MySQL database *"com.mysql.cj.jdbc.Driver", a*fter the driver has been registered, we can obtain a Connection instance that is connected to a particular database by calling *DriverManager.getConnection():,* in this method, we need to pass URL for connection and name and password of the database.
- ***URL for Connection***:- The connection URL for the mysql database is jdbc:mysql://localhost:3306/mydb ('mydb' is the name of database).

Specify to the DriverManager which JDBC drivers to try to make Connections use below line.

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

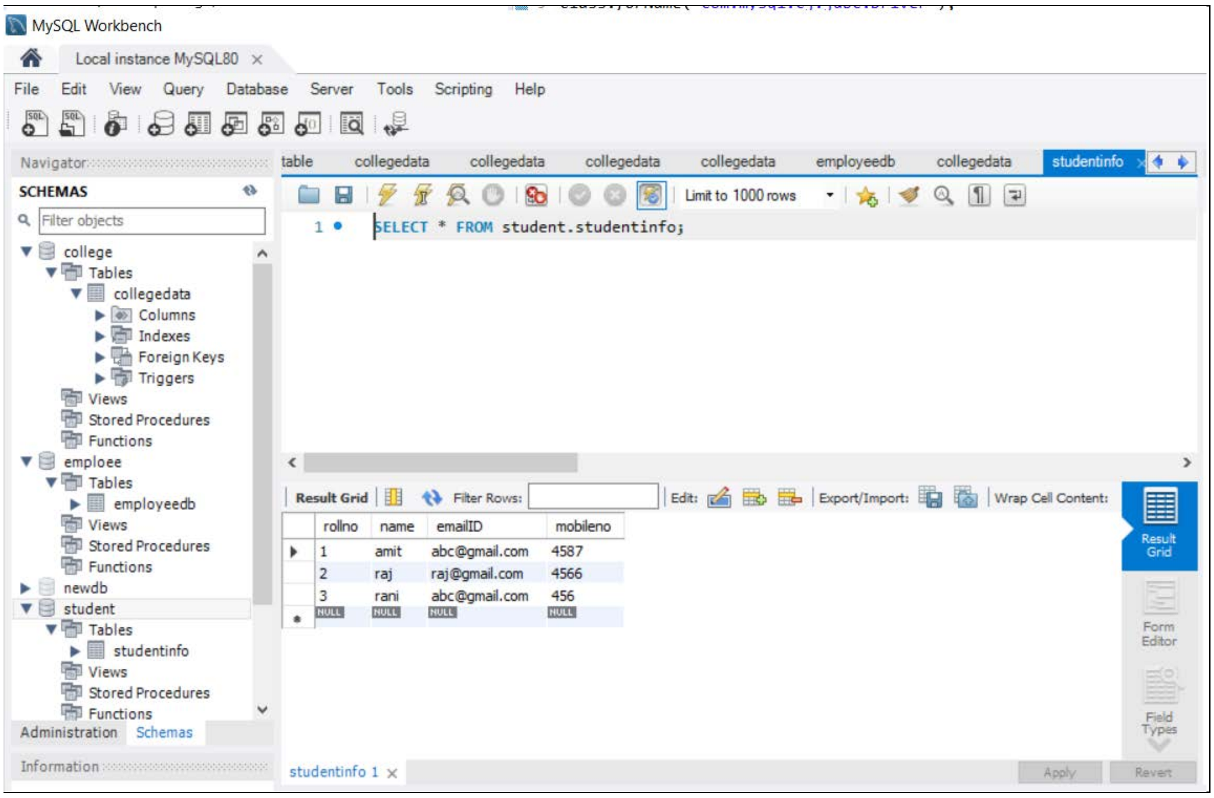
To get connection object use below line :-
Connection connection=DriverManager.getConnection("URL in string","username","password");
To get more clarification follow the connectivity code below.

5. In this src code, we will set up the connection and get all the data from the table. we have created the '*databaseClass.java*' file in the *src* folder.

```
//Java Database Connectivity with MySQL

import java.sql.*;
class databaseClass{
public static void main(String args[]){
try{
//step1 load the driver class
Class.forName("com.mysql.cj.jdbc.Driver");

//step2 create  the connection object  (establish the connection between java
application with oracle database
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/student","root","1234");
//here sonoo is database name, root is username and password
      //step3 create the statement object
Statement stmt=con.createStatement();
      //step4 write and execute query
//String query ="create table studentdata(rollno int(5) primary key, emailid
varchar(10), dept varchar(20))";
//stmt.executeUpdate(query);
//System.out.println("Student data created sussessfuly");


ResultSet rs=stmt.executeQuery("select * from studentinfo");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

//step5 close the connection object
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

Output :

```
1  amit  abc@gmail.com
2  raj  raj@gmail.com
3  rani  abc@gmail.com
```



```
SELECT * FROM student.studentinfo;
```

rollno	name	emailID	mobileno
1	amit	abc@gmail.com	4587
2	raj	raj@gmail.com	4566
3	rani	abc@gmail.com	456
NULL	NULL	NULL	NULL

Note:-

- To run the above code first create a table in your MySQL database and add some data manually.
- To compile the above code use "*javac -classpath ..\lib\mysql-connector-java-8.0.28.jar;. databaseClass.java*".
- To run the above code use "*java -classpath ..\lib\mysql-connector-java-8.0.28.jar;. databaseClass*".

B) To insert and retrieve the data from the database using JDBC

It can be of two types namely **structural and non-structural database**. The structural database is the one which can be stored in row and columns. A nonstructural database can not be stored in form of rows and columns. Most of the real-world data is non-structural like photos, videos, social media. As they are not having pre-defined data-types, so they are not present in the database. Hence, the structural database is stored in data warehouses and unstructured in data lakes. Java Database Connectivity is basically a standard API(application interface) between the java programming language and various databases like Oracle, SQL, PostgreSQL, MongoDB, etc. It connects the front end (for interacting with the users) with the backend for storing data
JDBC consists of 7 elements that are known as connection steps. They are listed below:

JDBC connection steps	Sep/Connection Number
1	Import the package
2	Load and Register the drivers
3	Establish the connection
4	Create the statement
5	Execute the statement
6	Process Result
7	Close/terminate

The syntax for importing package to deal with JDBC operations:

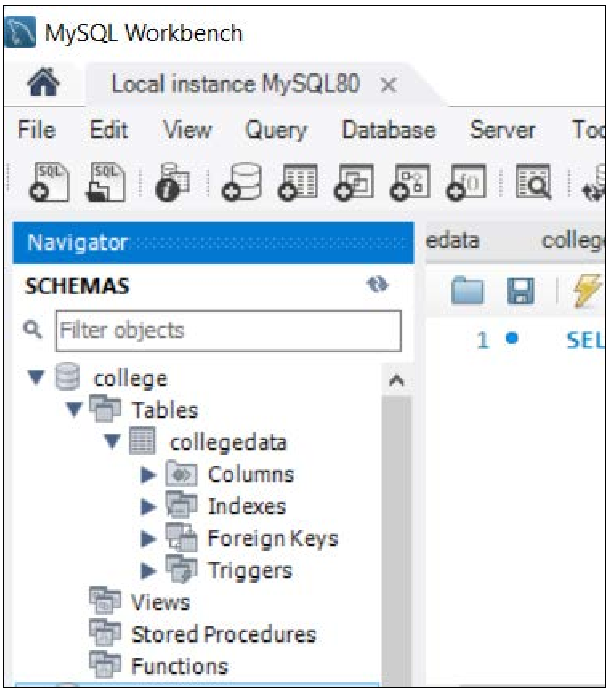
import java.sql.* ;

The syntax for registering drivers after loading the driver class:

forName(com.mysql.jdbc.xyz) ;

Steps: Below are 3 steps listed covering all 7 components and applying the same:
1. The first database will be created from which the data is supposed to be fetched. If the data is structured SQL can be used to fetch the data. If the data is unstructured MongoDB to fetch data from da lakes. Here SQL database is created and so do all further executions.
2. This step is optional as here need is simply too old and register drivers. In some IDE drivers are inbuilt for example NetBeans but they are missing in eclipse. Here the installation of drivers is needed. Jar files are needed to be downloaded hardly taking any space in memory.
3. Retrieve Contents of a Table Using JDBC connection

Step 1: Creating a database using MYSQL Workbench: The user can create a database using 'MYSQL Workbench' and create some tables in it and fill data inside it. Considering a random example of college database management systems. Now applying SQL commands over it naming this database as 'college'. Now suppose the user starts inserting tables inside it be named as "*collegedata*".



Step 2: Create Connection: Considering IDE as Eclipse for clear understanding because now work let in this package else if using any other IDEs install drivers first. Once cleared with

1. Create a new package
2. Open a new java file
3. Write the below code for JDBC
4. Save the filename with ***connection.java.***

Here considering IDE as Eclipse for clear' sake of understanding. Below is the illustration for the same:

```java
// Java Program to Insert Details in a Table using JDBC
// Connections class

// Importing all SQL classes
// Showing linking of created database

import java.sql.*;

public class connection {

    // object of Connection class
    // initially assigned NULL
    Connection con = null;

    public static Connection connectDB()

    {

        try {

            // Step 2 is involved among 7 in Connection
            // class i.e Load and register drivers

            // 2(a) Loading drivers using forName() method
            // name of database here is mysql
            Class.forName("com.mysql.cj.jdbc.Driver");

            // 2(b) Registering drivers using DriverManager
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/college",
                "root", "1234");
            // For DB here (custom sets)
            // root is the username, and
            // 1234 is the password

            // returning the object of Connection class
            // to be used in main class (Example2)
            return con;
        }

        // Catch block to handle the exceptions
        catch (SQLException | ClassNotFoundException e) {

            // Print the exceptions
            System.out.println(e);

            return null;
        }
    }
}
```

Step 3: Retrieve Contents of a Table Using JDBC connection: Suppose *"collegedata"* table has columns namely "name", "short", "email", "address", "phno" and the user wants to see the contents of "collegedata" table. It involves a series of steps give below with declaration and syntax for interpretation

3.1: Initialize a string with the SQL query as follows
String sql="select * from collegedata ";

3.2: Initialize the below objects of Connection class, PreparedStatement class, and ResultSet class(needed for JDBC) and connect with the database as follows:
Connection con=null;

PreparedStatement p=null;

ResultSet rs=null;

con=connection.connectDB();

3.3: Now, add the SQL query of 3.1 inside prepareStatement and execute it as follows
p =con.prepareStatement(sql);

rs =p.executeQuery();

3.4 Run a loop till rs.next() is not equal to NULL , fetch values from the table based on the data types, for example we use *getInt()* for integer datatype values and getString() for string datatype values.
3.5 Open a new java file (here, its GFG.java) inside the same package and type the full code (shown below) for the retrieval of contents of table "collegedata".
3.6 Both the files viz '*GFG.java*' and '*connection.java*' should be inside the same package, else the program won't give the desired output!
Implementation: Below is the java example illustrating the same:

// Java Program retrieving contents of

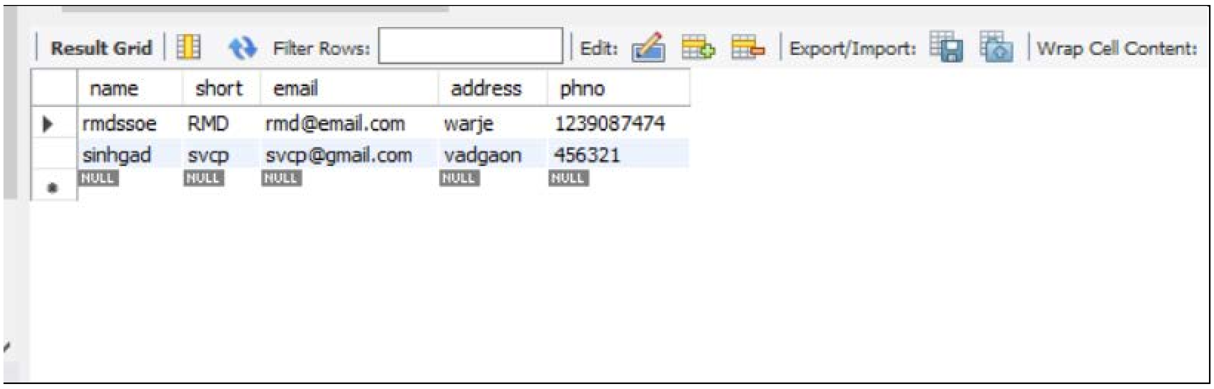
// Table Using JDBC connection

// Java code producing output which is based

// on values stored inside the "collegedata" table in DB

```
// Java Program to Insert Details in a Table using JDBC and retrieve/ display data
// Main class

// Step 1: Importing DB classes
// DB is SQL here
import java.sql.*;

// Main/App class of above Connection class
public class GFG {

    // MAin driver method
    public static void main(String[] args)
    {
        // Step 2: Showing above Connection class i.e
        // loading and registering drivers

        // Initially assigning NULL parameters
        // to object of Connection class
        Connection con = null;
        PreparedStatement ps = null;

        // Step 3: Establish the connection
        con = connection.connectDB();

        // Try block to check if exception/s occurs
        try {

            // Step 4: Create a statement
            String sql = "insert into  collegedata 
values('rmdssoe','RMD','rmd@email.com','warje','1239087474')";

            // Step 5: Execute the query
            ps = con.prepareStatement(sql);

            // Step 6: Process the results
            ps.execute();
        }

        // Optional but recommended
        // Step 7: Close the connection

        // Catch block to handle the exception/s
        catch (Exception e) {

            // Print the exception
            System.out.println(e);
        }
    }
}
```

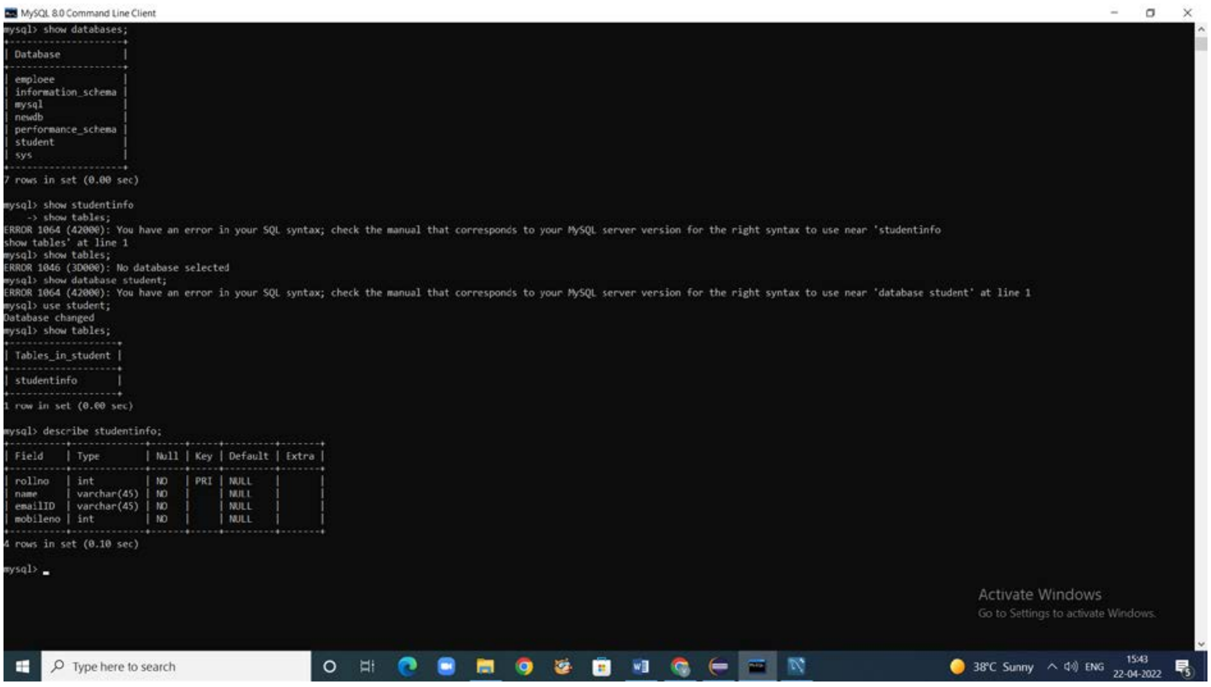
Output: The above output is completely based on the values stored inside the "collegedata" table as shown in the above code



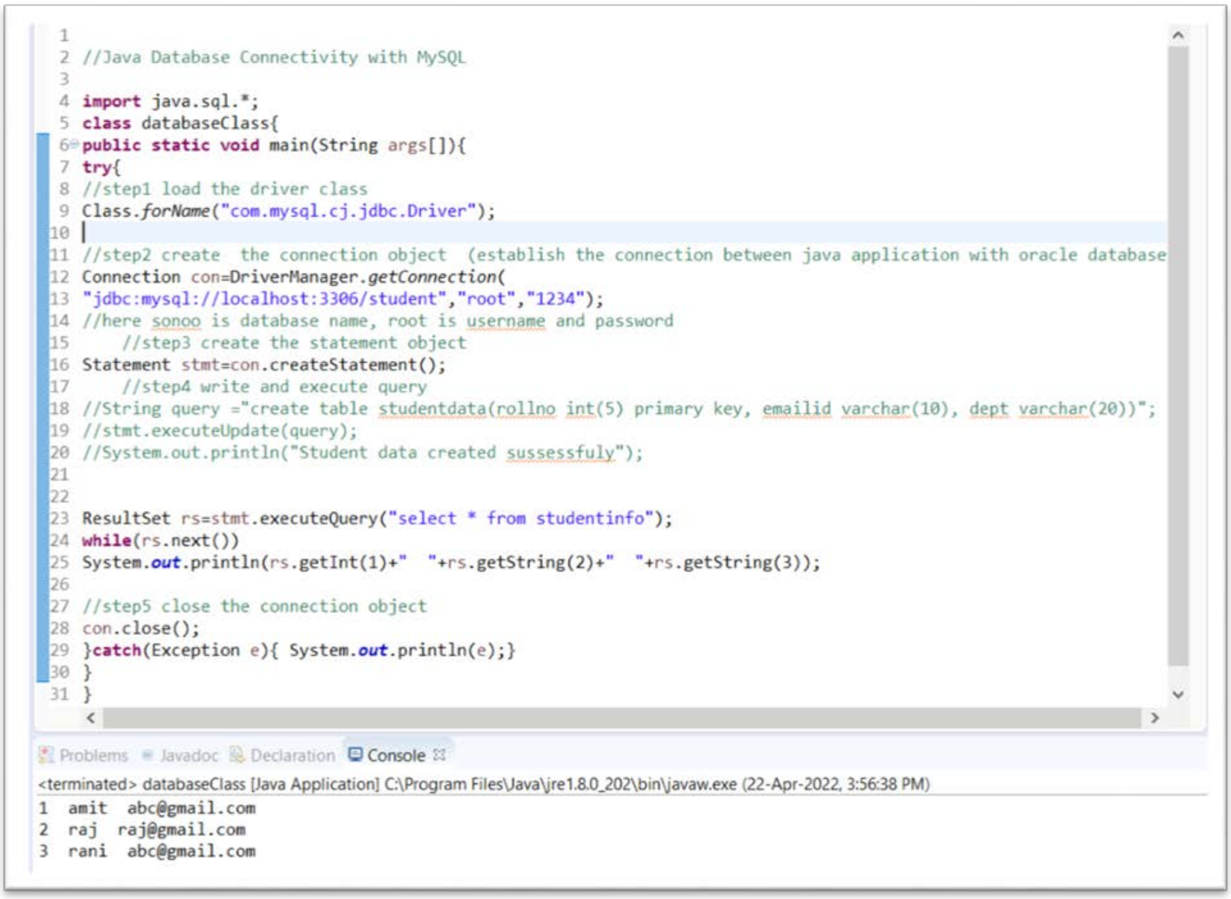
	name	short	email	address	phno
▶	rmdssoe	RMD	rmd@email.com	warje	1239087474
	sinhgad	svcp	svcp@gmail.com	vadgaon	456321
*	NULL	NULL	NULL	NULL	NULL

Screenshot's:

A) Database created in myssql



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| emploee            |
| information_schema |
| mysql              |
| newdb              |
| performance_schema |
| student            |
| sys                |
+--------------------+
7 rows in set (0.00 sec)

mysql> show studentinfo
    -> show tables;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'studentinfo
show tables' at line 1
mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> show database student;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'database student' at line 1
mysql> use student;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_student |
+-------------------+
| studentinfo       |
+-------------------+
1 row in set (0.00 sec)

mysql> describe studentinfo;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| rollno   | int         | NO   | PRI | NULL    |       |
| name     | varchar(45) | NO   |     | NULL    |       |
| emailID  | varchar(45) | NO   |     | NULL    |       |
| mobileno | int         | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.10 sec)

mysql> _
```

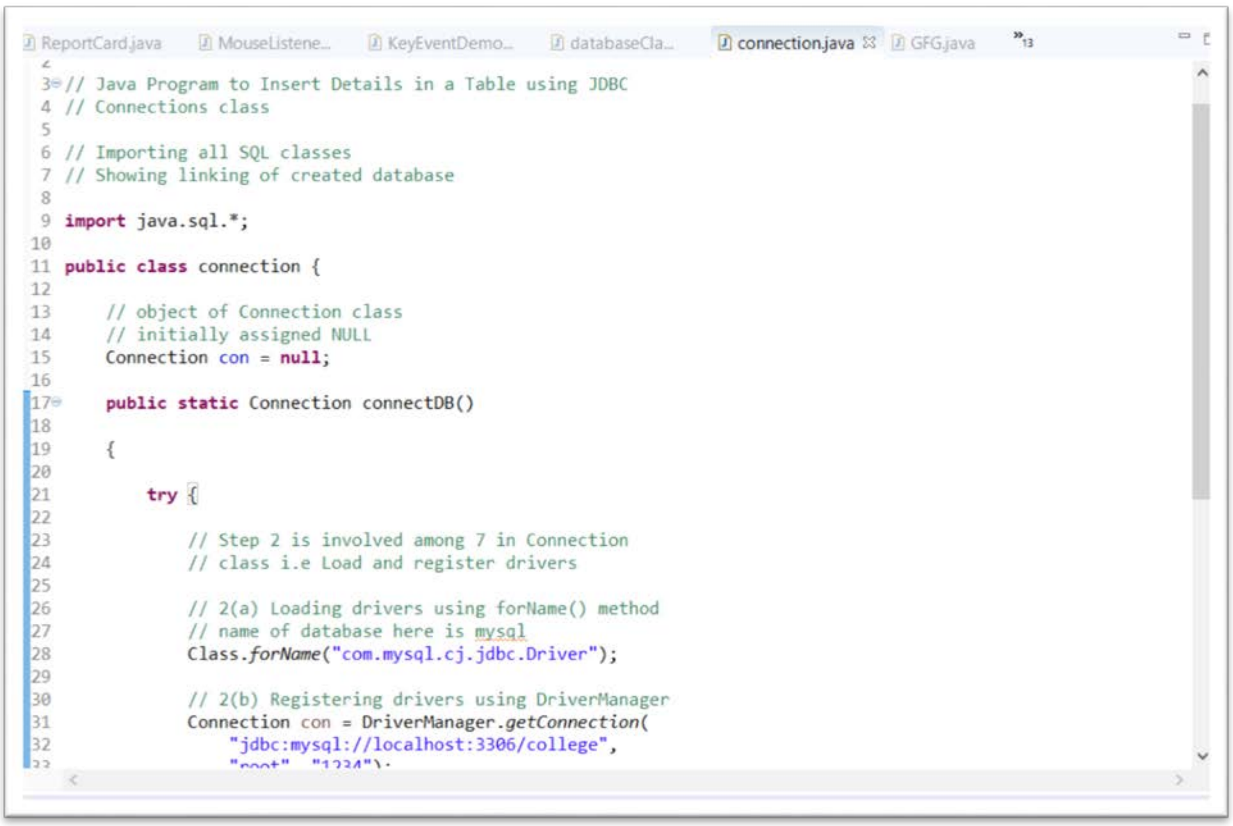


```java

//Java Database Connectivity with MySQL

import java.sql.*;
class databaseClass{
public static void main(String args[]){
try{
//step1 load the driver class
Class.forName("com.mysql.cj.jdbc.Driver");

//step2 create  the connection object  (establish the connection between java application with oracle database
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/student","root","1234");
//here sonoo is database name, root is username and password
    //step3 create the statement object
Statement stmt=con.createStatement();
    //step4 write and execute query
//String query ="create table studentdata(rollno int(5) primary key, emailid varchar(10), dept varchar(20))";
//stmt.executeUpdate(query);
//System.out.println("Student data created sussessfuly");


ResultSet rs=stmt.executeQuery("select * from studentinfo");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

//step5 close the connection object
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

```
<terminated> databaseClass [Java Application] C:\Program Files\Java\jre1.8.0_202\bin\javaw.exe (22-Apr-2022, 3:56:38 PM)
1  amit  abc@gmail.com
2  raj  raj@gmail.com
3  rani  abc@gmail.com
```

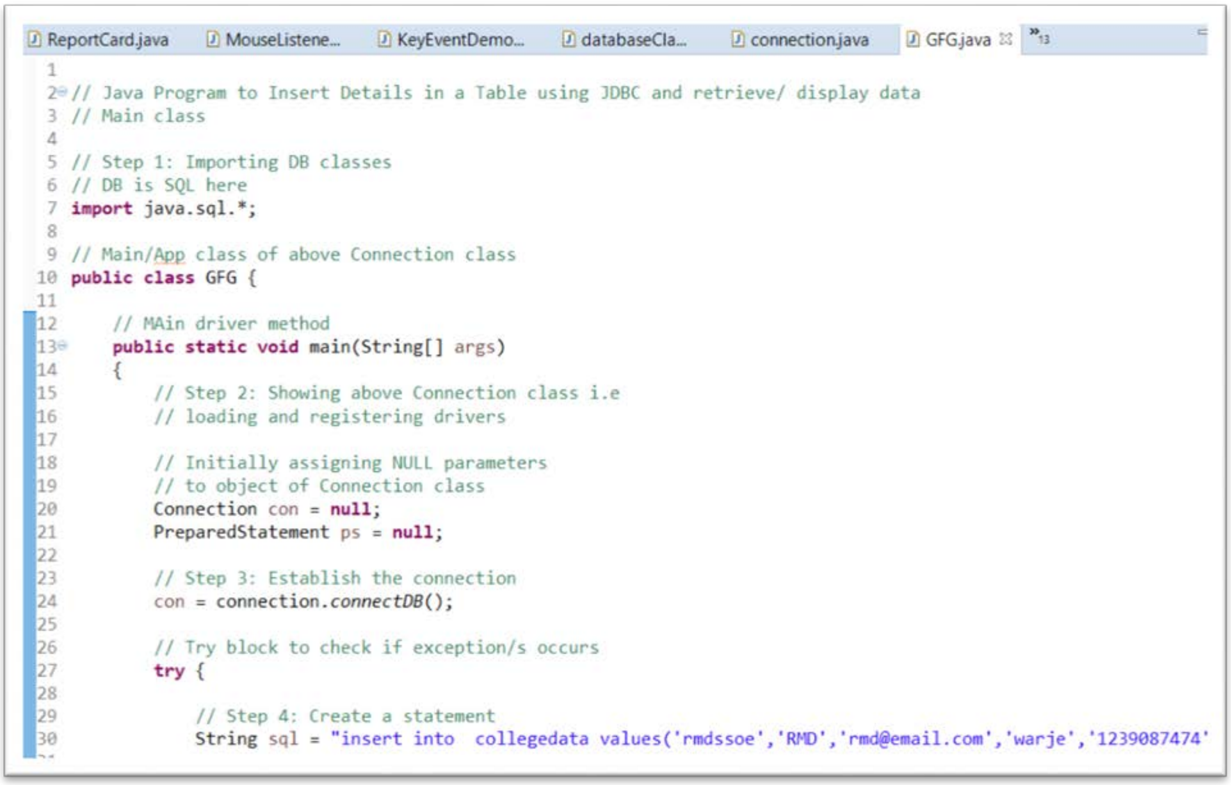
B) To insert and retrieve the data from the database using JDBC

a) connection. Java file



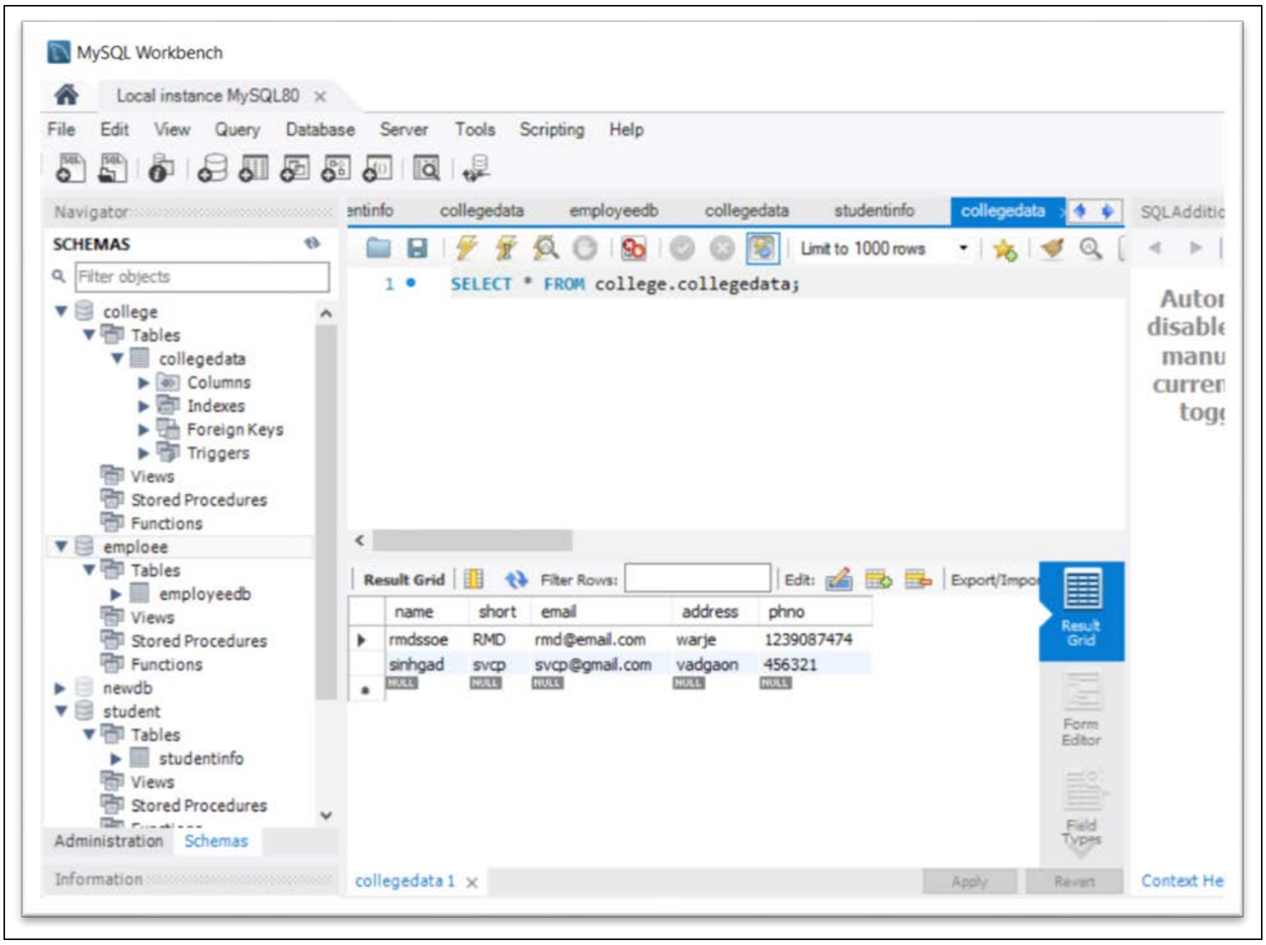
```
// Java Program to Insert Details in a Table using JDBC
// Connections class

// Importing all SQL classes
// Showing linking of created database

import java.sql.*;

public class connection {

    // object of Connection class
    // initially assigned NULL
    Connection con = null;

    public static Connection connectDB()

    {

        try {

            // Step 2 is involved among 7 in Connection
            // class i.e Load and register drivers

            // 2(a) Loading drivers using forName() method
            // name of database here is mysql
            Class.forName("com.mysql.cj.jdbc.Driver");

            // 2(b) Registering drivers using DriverManager
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/college",
```

b)GFG.java file



```

// Java Program to Insert Details in a Table using JDBC and retrieve/ display data
// Main class

// Step 1: Importing DB classes
// DB is SQL here
import java.sql.*;

// Main/App class of above Connection class
public class GFG {

    // MAin driver method
    public static void main(String[] args)
    {
        // Step 2: Showing above Connection class i.e
        // loading and registering drivers

        // Initially assigning NULL parameters
        // to object of Connection class
        Connection con = null;
        PreparedStatement ps = null;

        // Step 3: Establish the connection
        con = connection.connectDB();

        // Try block to check if exception/s occurs
        try {

            // Step 4: Create a statement
            String sql = "insert into  collegedata values('rmdssoe','RMD','rmd@email.com','warje','1239087474'
```

c) output



```
SELECT * FROM college.collegedata;
```

	name	short	email	address	phno
▶	rmdssoe	RMD	rmd@email.com	warje	1239087474
	sinhgad	svcp	svcp@gmail.com	vadgaon	456321
*	NULL	NULL	NULL	NULL	NULL

Conclusion: