1. Create a database and a table in it of your choice and write a program to connect it via JDBC

```
DATABASE : STUDENT
TABLE : STUDENT_DETAIL
```

student_id	student_name	course_id
1	PRIYA	SCI
2	FIZA	COM
3	POOJA	ART
4	GAURI	SCI
5	SURAJ	COM
6	DIVYA	ART
7	FRUTI	SCI
NULL	NULL	NULL

```
import java.sql.*;
public class Jdbc_Class {
   public static void main (String[] args){
       try {
           Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/student", "root",
"admin");
           Statement stmt = conn.createStatement();
           ResultSet result = stmt.executeQuery("select * from
student_detail");

           while (result.next()) {
               System.out.println(result.getInt("student_name")+"
"+result.getString("student_id")+" "+result.getString("course_id"));
           }
       }
       catch (Exception e)
       {
           e.printStackTrace();
       }
   }
}
```

```
 C:\Users\supar\IdeaProjects\JDBC_Project\lib\mysql-connector-java-8.0.27.jar Jdbc
PRIYA 1 SCI
FIZA 2 COM
POOJA 3 ART
GAURI 4 SCI
SURAJ 5 COM
DIVYA 6 ART
FRUTI 7 SCI

Process finished with exit code 0
```

Run TODO Problems Terminal Build

2. Write a java program to fetch the record from the table created and store it in a pojo class object.

Student.java

```
public class Student {
    private int id ;
    private String name;
    private String course_id;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCourse_id() {
        return course_id;
    }

    public void setCourse_id(String course_id) {
        this.course_id = course_id;
    }

    @Override
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", course_id='" + course_id + '\'' +
                '}';
    }
}
```

Jdbc_Class.java

```
import java.sql.*;
import java.util.ArrayList;

public class Jdbc_Class {
    public static void main (String[] args){
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/student", "root",
"admin");
            Statement stmt = conn.createStatement();
            ResultSet result = stmt.executeQuery("select * from student_detail");
            ArrayList<Student> student_list = new ArrayList<>();
            while (result.next()) {
```

```
            Student stud = new Student();
            stud.setId(result.getInt("student_id"));
            stud.setName(result.getString("student_name"));
            stud.setCourse_id(result.getString("course_id"));
            student_list.add(stud);
            System.out.println(result.getString("student_name")+" "+result.getInt("student_id")+"
"+result.getString("course_id"));
        }
        System.out.println(student_list);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
  }
}
```

Output

3. **Write a java program to update the record .**

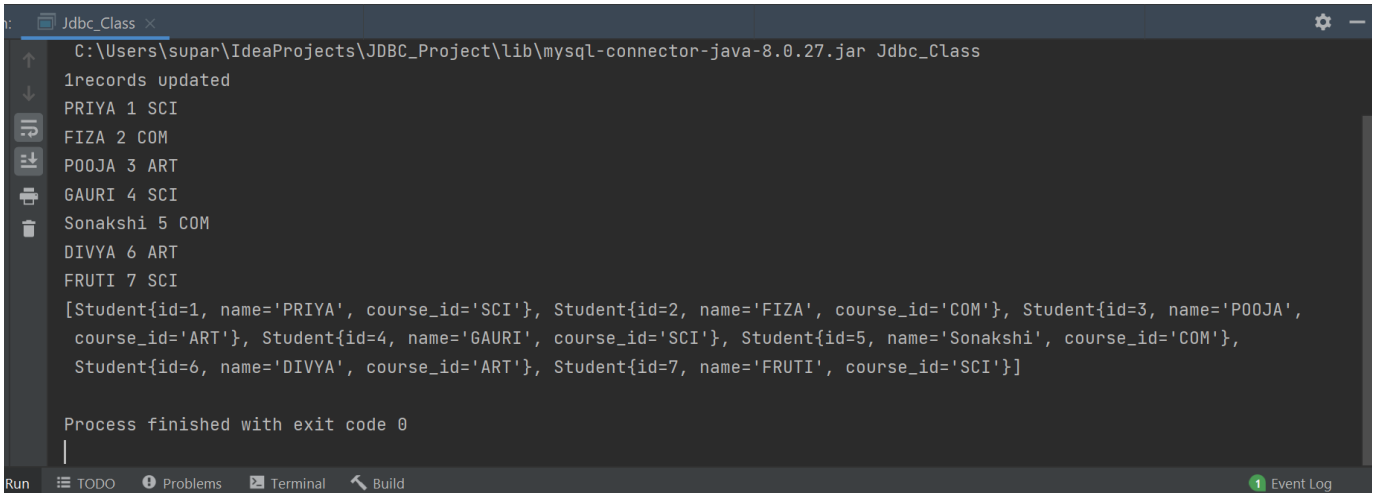
Jdbc_Class.java

```
import java.sql.*;
import java.util.ArrayList;

public class Jdbc_Class {
   public static void main (String[] args){
      try {
         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/student", "root",
"admin");
         PreparedStatement stmt = conn.prepareStatement("update student_detail set student_name =?
where student_id=?");
         stmt.setString(1,"Sonakshi");
         stmt.setInt(2,5);
         int i = stmt.executeUpdate();
         System.out.println(i+"records updated");
         ResultSet result = stmt.executeQuery("select * from student_detail");
         ArrayList<Student> student_list = new ArrayList<>();
```

```
            while (result.next()) {
                Student stud = new Student();
                stud.setId(result.getInt("student_id"));
                stud.setName(result.getString("student_name"));
                stud.setCourse_id(result.getString("course_id"));
                student_list.add(stud);
                System.out.println(result.getString("student_name")+" "+result.getInt("student_id")+"
"+result.getString("course_id"));
            }
            System.out.println(student_list);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Output



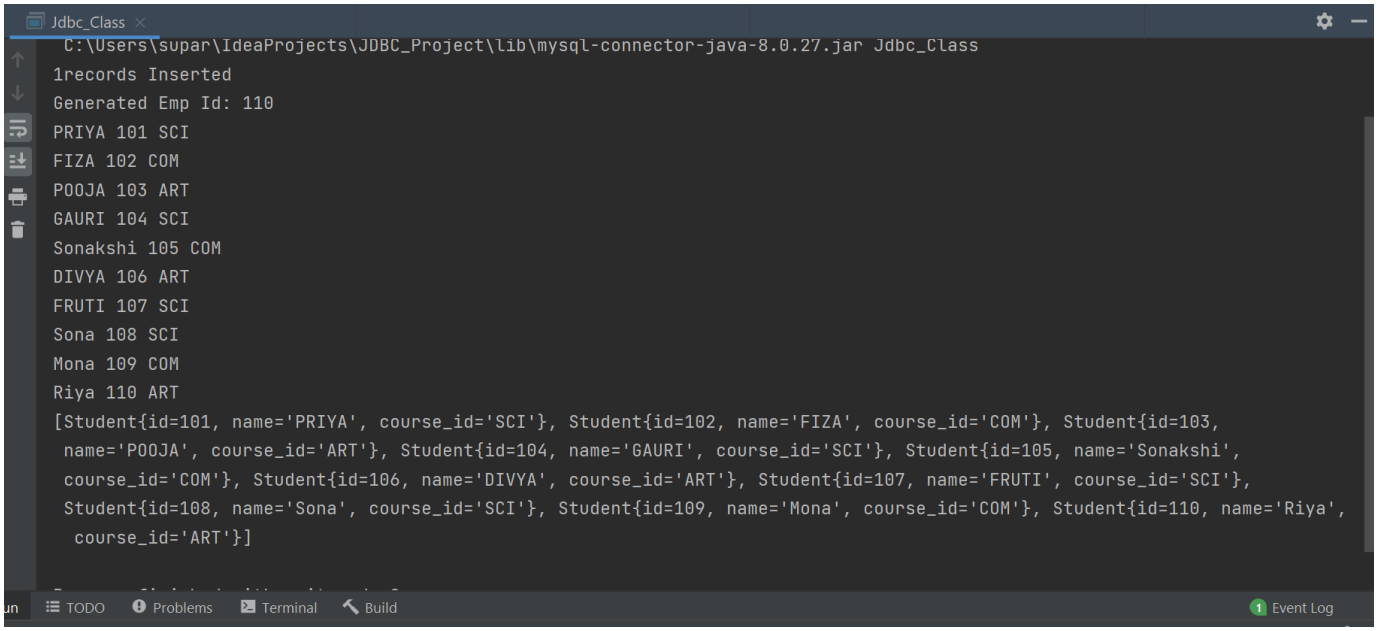
4. **Write a java program to insert a new record and print the auto incremented primary key of newly inserted record**

Jdbc_Class.java

```
public class Jdbc_Class {
    public static void main (String[] args){
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/student", "root",
"admin");
            PreparedStatement stmt = conn.prepareStatement("insert into student_detail
(student_id,student_name,course_id)values(?,?,?)",Statement.RETURN_GENERATED_KEYS);
            stmt.setInt(1,110);
            stmt.setString(2,"Riya");
            stmt.setString(3,"ART");
            int i = stmt.executeUpdate();
            System.out.println(i+" records Inserted");


            ResultSet result = stmt.getGeneratedKeys();
            if(result != null && result.next()){
                System.out.println("Generated Emp Id: "+result.getInt(1));
```

```
        }
        result=stmt.executeQuery("Select * from student_detail");
        ArrayList<Student> student_list = new ArrayList<>();
        while (result.next()) {
            Student stud = new Student();
            stud.setId(result.getInt("student_id"));
            stud.setName(result.getString("student_name"));
            stud.setCourse_id(result.getString("course_id"));
            student_list.add(stud);
            System.out.println(result.getString("student_name")+" "+result.getInt("student_id")+"
"+result.getString("course_id"));
        }
        System.out.println(student_list);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
  }
}
```



5. Demonstrate the example for different kinds of statements in JDBC by writing a java program

- ★ **Statement Interface**

The Statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet i.e. it provides factory method to get the object of ResultSet.

Statement stmt=con.createStatement();

stmt.executeUpdate("insert into student_detail values(111,”Irfan”,”SCI”)");
int result=stmt.executeUpdate("update student_detail set student_name=”Vimal” where student_id=108");
int result=stmt.executeUpdate("delete from student_detail where student_id = 105");
System.out.println(result +" records affected");
con.close();

- ★ **PreparedStatement Interface**

The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

Insert

```
PreparedStatement stmt=con.prepareStatement("insert into student_detail values(?,?,?)");
stmt.setInt(1,113);
stmt.setString(2,"Rani");
stmt.setString(3,”ART”);

int i=stmt.executeUpdate();
System.out.println(i+" records inserted");
```

Update

```
PreparedStatement stmt=con.prepareStatement("update student_detail set student_name=? where
id=?");
stmt.setString(1,"Sonu");
stmt.setInt(2,101);

int i=stmt.executeUpdate();
System.out.println(i+" records updated");
```

Delete

```
PreparedStatement stmt=con.prepareStatement("delete from student_detail where id=?");
stmt.setInt(1,108);

int i=stmt.executeUpdate();
System.out.println(i+" records deleted");
```

Select

```
PreparedStatement stmt=con.prepareStatement("select * from student_detail");
ResultSet rs=stmt.executeQuery();
while(rs.next()){
System.out.println(rs.getInt(1)+" "+rs.getString(2)+” “+rs.getString(3));
}
```

- ★ **CallableStatement :** CallableStatement interface is used to call the stored procedures and functions.

Stored procedure : INSERY

```
USE `student`;
DROP procedure IF EXISTS `INSERY`;

DELIMITER $$
USE `student`$$

CREATE PROCEDURE INSERY (IN student_id int ,  IN student_name VARCHAR(50), IN course_id
VARCHAR(50))

BEGIN
insert into student_detail values(student_id,student_name,course_id);
Select * from student_detail;
END$$

DELIMITER ;
```

Jdbc_Class.java

```
CallableStatement stmt=conn.prepareCall("{call INSERY(?,?,?)}");
        stmt.setInt(1,113);
        stmt.setString(2,"Amit");
```

```
        stmt.setString(3,"SCI");
        stmt.execute();
System.out.println(“Procedure performed successfully”);
```

6. Go through the concept of batch processing and demonstrate with an example by writing a java program

Instead of executing a single query, we can execute a batch (group) of queries. It makes the performance fast.

The java.sql.Statement and java.sql.PreparedStatement interfaces provide methods for batch processing.

Batch Processing using Statement interface

Jdbc_Class.java

```
import java.sql.*;
import java.util.ArrayList;

public class Jdbc_Class {
   public static void main (String[] args){
      try {
         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/student", "root",
"admin");
         Statement stmt = conn.createStatement();
         stmt.addBatch("insert into student_detail values (114,'abhi','SCI')");
         stmt.addBatch("insert into student_detail values (115,'abhinav','COM')");

         stmt.executeBatch();

         System.out.println("Inserted Successfully");
          }
      catch (Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

Batch Processing using PreparedStatement interface

Jdbc_Class.java

```
import java.sql.*;
import java.io.*;
import java.util.ArrayList;

public class Jdbc_Class {
   public static void main (String[] args){
      try {
         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/student", "root",
"admin");

         PreparedStatement ps=conn.prepareStatement("insert into student_detail values(?,?,?)");

         BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
         while(true){

            System.out.println("enter student_id");
            String s1=br.readLine();
            int id=Integer.parseInt(s1);

            System.out.println("enter student_name");
            String name=br.readLine();
```

```
            System.out.println("enter course_id");
            String course=br.readLine();


            ps.setInt(1,id);
            ps.setString(2,name);
            ps.setString(3,course);

            ps.addBatch();
            System.out.println("Want to add more records y/n");
            String ans=br.readLine();
            if(ans.equals("n")){
               break;
            }

        }
        ps.executeBatch();

        System.out.println("record successfully saved");

        conn.close();

}
      catch (Exception e)
      {
         e.printStackTrace();
      }
   }
}
```