StudentAdmissionDAO.java

```
package com.cts.unoadm.dao;


import java.util.ArrayList;

import java.util.List;

import java.util.Date;

import java.sql.Connection;

import java.sql.SQLException;

import java.sql.PreparedStatement;

import java.sql.ResultSet;


import com.cts.unoadm.exception.StudentAdmissionException;

import com.cts.unoadm.vo.StudentAdmission;

import com.cts.unoadm.util.ApplicationUtil;

import com.cts.unoadm.util.DBConnectionManager;



public class StudentAdmissionDAO {


		@SuppressWarnings("finally")

		public boolean addStudentAdmissionDetails(List<StudentAdmission>
stdAdmissions) throws StudentAdmissionException {

		boolean recordsAdded = false;

		//code here
```

```
try(Connection con = DBConnectionManager.getInstance().getConnection()) {
    for(StudentAdmission stdAdmObj:stdAdmissions) {
        String sql = "INSERT INTO students
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?);";
        PreparedStatement prepState = con.prepareStatement(sql);
        prepState.setString(1, stdAdmObj.getAdmissionId());
        prepState.setString(2, stdAdmObj.getStudentCode());
        prepState.setDate(3,
ApplicationUtil.convertUtilToSqlDate(stdAdmObj.getDateOfCounseling()));
        prepState.setString(4, stdAdmObj.getDepartmentName());
        prepState.setDate(5,
ApplicationUtil.convertUtilToSqlDate(stdAdmObj.getDateOfAdmission()));
        prepState.setString(6, stdAdmObj.getPreferCollegeHostel());
        prepState.setString(7, stdAdmObj.getFirstGraduate());
        prepState.setString(8, stdAdmObj.getManagerApproval());
        prepState.setFloat(9, (float)stdAdmObj.getAdmissionFee());
        prepState.setFloat(10, (float)stdAdmObj.getTuitionFee());
        prepState.setDouble(11, (double)stdAdmObj.getHostelFee());
        prepState.setFloat(12, (float)stdAdmObj.getTotalCollegeFee());
        prepState.setString(13, stdAdmObj.getFinalStatusOfAdmission());
        prepState.execute();
    }

    recordsAdded= true;
```

```
		} catch(Exception e) {
			System.out.println(e.getMessage());

			throw new StudentAdmissionException(e.getMessage(), e.getCause());
		} finally {

			return recordsAdded;
		}
	}


	@SuppressWarnings("finally")
	public List<StudentAdmission> getAllStudentAdmissionDetails() throws
StudentAdmissionException {

		List<StudentAdmission> stdAdmissions = new ArrayList<StudentAdmission>();

		//code here
		try(Connection con = DBConnectionManager.getInstance().getConnection()) {
			String sql = "SELECT * FROM students";
			PreparedStatement prepState = con.prepareStatement(sql);
			ResultSet resSet = prepState.executeQuery();
			while(resSet.next()) {
				String admissionId = resSet.getString(1);
```

```
                String studentCode = resSet.getString(2);
                Date dateOfCounseling =
ApplicationUtil.convertStringToDate(resSet.getString(3));
                String departmentName = resSet.getString(4);
                Date dateOfAdmission =
ApplicationUtil.convertStringToDate(resSet.getString(5));
                String preferCollegeHostel = resSet.getString(6);
                String firstGraduate = resSet.getString(7);
                String managerApproval = resSet.getString(8);
                double admissionFee = resSet.getDouble(9);
                double tuitionFee = resSet.getDouble(10);
                double hostelFee = resSet.getDouble(11);
                double totalCollegeFee = resSet.getDouble(12);
                String finalStatusOfAdmission = resSet.getString(13);
                StudentAdmission stdAdmObj = new StudentAdmission(
                                admissionId,
                                studentCode,
                                dateOfCounseling,
                                departmentName,
                                dateOfAdmission,
                                preferCollegeHostel,
                                firstGraduate,
                                managerApproval,
                                admissionFee,
                                tuitionFee,
```

```
                                                hostelFee,
                                                totalCollegeFee,
                                                finalStatusOfAdmission
                                        );
                                stdAdmissions.add(stdAdmObj);
                        }
                        resSet.close();
                } catch(SQLException e) {
                        throw new StudentAdmissionException(e.getMessage(), e.getCause());
                } finally {
                        return stdAdmissions;
                }

        }
}
```

StudentAdmissionException.java

```
package com.cts.unoadm.exception;

public class StudentAdmissionException extends Exception {

      private static final long serialVersionUID = -
1105431869622052445L;

      /**
       * @param message
       * @param cause
       */
      public StudentAdmissionException(String message, Throwable cause)
{
```

```
            super(message, cause);
        }
}
```

MainApp.java

```
package com.cts.unoadm.main;

//import java.io.BufferedReader;
//import java.io.File;
//import java.io.FileReader;
//import java.io.BufferedReader;
//import java.io.File;
//import java.io.FileNotFoundException;
//import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

import com.cts.unoadm.skeletonvalidator.SkeletonValidator;
import com.cts.unoadm.service.StudentAdmissionService;
import com.cts.unoadm.exception.StudentAdmissionException;

public class MainApp {
```

```
	public static void main(String[] args) throws IOException {
		//Don't delete this code
		//Skeletonvalidaton starts
		new SkeletonValidator();
		//Skeletonvalidation ends

		//Write your code here..
		@SuppressWarnings("resource")
		Scanner sc = new Scanner(System.in);
		StudentAdmissionService stdAdmService = new
StudentAdmissionService();
		try {
//			File file = new File("inputFeed.txt");
//
//			BufferedReader br = new BufferedReader(new
FileReader(file));
//
//			String st;
//			while ((st = br.readLine()) != null)
//			  System.out.println(st);
//			br.close();
```

```
        if(stdAdmService.addStudentAdmissionDetails("inputFeed.txt"))
                        {
                                System.out.println("Data has been inserted into
database");
                        } else {
                                System.out.println("Database insertion failed");


                        }
                        System.out.print("Enter a admission Id to search status - ");
                        String admissionId = sc.nextLine();
                        if(stdAdmService.searchStudentAdmission(admissionId)) {
                                System.out.println("Student admission found");
                        } else {
                                System.out.println("Can't be found on database");
                        }
                } catch(StudentAdmissionException e) {
                        System.out.println(e.getMessage());
                }
        }
```

```
}
```

StudentAdmissionService.java

```
package com.cts.unoadm.service;

import java.util.ArrayList;
import java.util.List;
import java.util.Date;

import com.cts.unoadm.exception.StudentAdmissionException;
import com.cts.unoadm.vo.StudentAdmission;
import com.cts.unoadm.util.ApplicationUtil;
import com.cts.unoadm.dao.StudentAdmissionDAO;


public class StudentAdmissionService {

	/**
	 * @return List<StudentAdmission>
	 */
```

```
	public static List<StudentAdmission>
buildStudentAdmissionsList(List<String> studentAdmissionRecords) {

		List<StudentAdmission> studentAdmissionList = new
ArrayList<StudentAdmission>();


		//Code here

		for(String line:studentAdmissionRecords) {

			String[] words = line.split(",");

			String admissionId = words[0].trim();

			String studentCode = words[1].trim();

			Date dateOfCounseling =
ApplicationUtil.convertStringToDate(words[2].trim());

			String departmentName = words[3].trim();

			Date dateOfAdmission =
ApplicationUtil.convertStringToDate(words[4].trim());

			String preferCollegeHostel = words[5].trim();

			String firstGraduate = words[6].trim();

			String managerApproval = words[7].trim();

			double[] fees = calculateTotalCollegeFee(preferCollegeHostel,
firstGraduate, departmentName);

			double admissionFee = fees[0];

			double tuitionFee = fees[1];

			double hostelFee = fees[2];
```

```
                double totalCollegeFee = fees[3];


                String finalStatusOfAdmission = "AdmissionSuccessfull";
                StudentAdmission stdObj = new StudentAdmission(
                                admissionId,
                                studentCode,
                                dateOfCounseling,
                                departmentName,
                                dateOfAdmission,
                                preferCollegeHostel,
                                firstGraduate,
                                managerApproval,
                                admissionFee,
                                tuitionFee,
                                hostelFee,
                                totalCollegeFee,
                                finalStatusOfAdmission
                        );
                studentAdmissionList.add(stdObj);
        }
        return studentAdmissionList;
}
```

```
	public boolean addStudentAdmissionDetails(String inputFeed) throws
StudentAdmissionException {


		//Code here

		List<String> parsedRecords = ApplicationUtil.readFile(inputFeed);

		List<StudentAdmission> studentAdmissionRecords =
StudentAdmissionService.buildStudentAdmissionsList(parsedRecords);

		StudentAdmissionDAO s=new StudentAdmissionDAO();


		return s.addStudentAdmissionDetails(studentAdmissionRecords);


	}



	//	return false;

	//}


	public static double[] calculateTotalCollegeFee(String preferCollegeHostel,
String firstGraduate, String departmentName) {

		double[] studentAdmissionCosts = new double[4];
```

```
        //Code here..
        studentAdmissionCosts[0] = 30000d;
        studentAdmissionCosts[1] = 0d;
        studentAdmissionCosts[2] = 0d;
        studentAdmissionCosts[3] = 0d;
        if(departmentName.equalsIgnoreCase("EEE") ||
departmentName.equalsIgnoreCase("CSE") ||
departmentName.equalsIgnoreCase("IT")) {
                studentAdmissionCosts[1] = 45000d;
        } else if(departmentName.equalsIgnoreCase("ECE") ||
departmentName.equalsIgnoreCase("CIVIL")) {
                studentAdmissionCosts[1] = 50000d;
        } else if(departmentName.equalsIgnoreCase("MECH")) {
                studentAdmissionCosts[1] = 55000d;
        }

        if(preferCollegeHostel.equalsIgnoreCase("YES")) {
                studentAdmissionCosts[2] = 75000d;
        }

        studentAdmissionCosts[3] = studentAdmissionCosts[0] +
studentAdmissionCosts[1] + studentAdmissionCosts[2];
```

```
		if(firstGraduate.equalsIgnoreCase("YES")) {
			studentAdmissionCosts[3] -= 20000d;
		}

		return studentAdmissionCosts;
	}

	public boolean searchStudentAdmission(String admissionId) throws StudentAdmissionException {
		boolean status = false;

		//Code here..
		List<StudentAdmission> fetchedAdmissions = new StudentAdmissionDAO().getAllStudentAdmissionDetails();

		for(StudentAdmission stdAdm:fetchedAdmissions) {
			if(stdAdm.getAdmissionId().equalsIgnoreCase(admissionId)) {
				status = true;
				System.out.println(stdAdm.toString());
			}
		}
```

```
            return status;
        }
}
```

SkeletonValidator.java

```
package com.cts.unoadm.skeletonvalidator;

//import java.lang.reflect.Array;
import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author t-aarti3
 *       This class is used to verify if the Code Skeleton is intact and not
 *       modified by participants thereby ensuring smooth auto evaluation
 * */

public class SkeletonValidator {
        public SkeletonValidator() {
                validateClassName("com.cts.unoadm.util.DBConnectionManager");
                validateClassName("com.cts.unoadm.util.ApplicationUtil");
```

```
validateClassName("com.cts.unoadm.service.StudentAdmissionService");
validateClassName("com.cts.unoadm.dao.StudentAdmissionDAO");
validateClassName("com.cts.unoadm.vo.StudentAdmission");

validateClassName("com.cts.unoadm.exception.StudentAdmissionException");

validateMethodSignature(
"addStudentAdmissionDetails:boolean,getAllStudentAdmissionDetails:List",
"com.cts.unoadm.dao.StudentAdmissionDAO");
validateMethodSignature(
"buildStudentAdmissionsList:List,addStudentAdmissionDetails:boolean,calculateTotalCollegeFee:double[],searchStudentAdmission:boolean",
"com.cts.unoadm.service.StudentAdmissionService");
validateMethodSignature(
"readFile:List,convertUtilToSqlDate:Date,convertStringToDate:Date,checkIfValidAdmission:boolean",
"com.cts.unoadm.util.ApplicationUtil");
validateMethodSignature(
```

```
"getConnection:Connection,getInstance:DBConnectionManager",
                        "com.cts.unoadm.util.DBConnectionManager");




}
private static final Logger LOG = Logger.getLogger("SkeletonValidator");
protected final boolean validateClassName(String className) {

        boolean iscorrect = false;
        try {
                Class.forName(className);
                iscorrect = true;
                LOG.info("Class Name " + className + " is correct");

        } catch (ClassNotFoundException e) {
                LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
                                + "and class name as provided in the skeleton");

        } catch (Exception e) {
```

```
                LOG.log(Level.SEVERE,
                                "There is an error in validating the " + "Class
Name. Please manually verify that the "
                                        + "Class name is same as skeleton
before uploading");
            }
            return iscorrect;
        }

        protected final void validateMethodSignature(String methodWithExcptn,
String className) {
            Class<?> cls = null;
            try {

                String[] actualmethods = methodWithExcptn.split(",");
                boolean errorFlag = false;
                String[] methodSignature;
                String methodName = null;
                String returnType = null;

                for (String singleMethod : actualmethods) {
                    boolean foundMethod = false;
                    methodSignature = singleMethod.split(":");
```

```
                methodName = methodSignature[0];
                returnType = methodSignature[1];
                cls = Class.forName(className);
                Method[] methods = cls.getMethods();
                for (Method findMethod : methods) {
                        if (methodName.equals(findMethod.getName())) {
                                foundMethod = true;
                                if
(!(findMethod.getReturnType().getSimpleName().equals(returnType))) {
                                        errorFlag = true;
                                        LOG.log(Level.SEVERE, " You have
changed the " + "return type in '" + methodName
                                                        + "' method. Please stick
to the " + "skeleton provided");

                                } else {
                                        LOG.info("Method signature of " +
methodName + " is valid");
                                }

                        }
                }
```

```
                        if (!foundMethod) {
                                errorFlag = true;
                                LOG.log(Level.SEVERE, " Unable to find the given
public method " + methodName
                                                + ". Do not change the " + "given
public method name. " + "Verify it with the skeleton");
                        }

                }
                if (!errorFlag) {
                        LOG.info("Method signature is valid");
                }

        } catch (Exception e) {
                LOG.log(Level.SEVERE,
                                " There is an error in validating the " + "method
structure. Please manually verify that the "
                                                + "Method signature is same as the
skeleton before uploading");
        }
    }

}
```

ApplocationUtil.java

```
package com.cts.unoadm.util;



import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.stream.Stream;
import java.util.concurrent.TimeUnit;



import com.cts.unoadm.exception.StudentAdmissionException;

public class ApplicationUtil {

	/**
	 * @param fileName
```

```
	 * @return List<String>
	 * @throws StudentAdmissionException
	 */
	public static List<String> readFile(String fileName) throws StudentAdmissionException {
		List<String> studentAdmissionList = new ArrayList<String>();
		 //Code here..
			try(Stream<String> lines = Files.lines(Paths.get(fileName))) {
			lines.forEach((line)->{
				String[] words = line.split(",");
				if(ApplicationUtil.checkIfValidAdmission(

	ApplicationUtil.convertStringToDate(words[2].trim()),

	ApplicationUtil.convertStringToDate(words[4].trim()),
								words[7].trim()
						)
					){
					studentAdmissionList.add(line.trim());
				}
			});
		} catch(IOException e) {
```

```
                    throw new StudentAdmissionException(e.getMessage(),
e.getCause());
                }


                return studentAdmissionList;
        }







        /**
         * @param util
         *          Date
         * @return sql Date
         */
        public static java.sql.Date convertUtilToSqlDate(java.util.Date uDate) {




                //Code here..


                return new java.sql.Date(uDate.getTime());
```

```
        //return sDate;
}


/**
 * @param inDate
 * @return Date
 */
public static Date convertStringToDate(String inDate) {


        //Code here..
        SimpleDateFormat sDf = new SimpleDateFormat("yyyy-MM-dd");
        Date date = null;
        try {
                date = sDf.parse(inDate);
        } catch(ParseException e) {
                e.printStackTrace();
        }
        return date;//TODO change this return value
}
```

```
        //return new Date();//TODO change this return value
    //}

    public static boolean checkIfValidAdmission(Date dtOfCounseling, Date
dtOfAdmission, String manager) {
        boolean admissionValidity = false;

        //Code here..
        long counselingMillis = dtOfCounseling.getTime();
        long admissionMillis = dtOfAdmission.getTime();

        long days = TimeUnit.DAYS.convert(Math.abs(admissionMillis -
counselingMillis), TimeUnit.MILLISECONDS);

        if(days <= 10 && manager.equalsIgnoreCase("Approved")) {
            admissionValidity = true;
        }

        return admissionValidity;
    }
}
```

DBConnectionManager.java

```
/**
 * Don't change this code
 */
package com.cts.unoadm.util;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;



import com.cts.unoadm.exception.StudentAdmissionException;


public class DBConnectionManager {

	public static final String PROPERTY_FILE = "database.properties";
	public static final String DRIVER = "drivername";
	public static final String URL = "url";
	public static final String USER_NAME = "username";
	public static final String PASSWORD = "password";

	private static Connection connection = null;
	private static Properties props = null;

	/**
	 * @throws StudentAdmissionException
	 */
	private DBConnectionManager() throws StudentAdmissionException {
		loadProperties();
		try {
			Class.forName(props.getProperty(DRIVER));
		DBConnectionManager.connection =
DriverManager.getConnection(props.getProperty(URL),
props.getProperty(USER_NAME),
				props.getProperty(PASSWORD));
		//Class.forName(com.mysql.cj.jdbc.Driver);
```

```
            //DBConnectionManager.connection =
DriverManager.getConnection(URL,USER_NAME,PASSWORD);

            } catch (ClassNotFoundException ex) {

                  throw new StudentAdmissionException("Could not find
Driver class ", ex.getCause());
            } catch (SQLException e) {
                  throw new StudentAdmissionException("Database
Connection Creation Failed", e.getCause());
            }
      }

      /**
       * @return Connection
       */
      public Connection getConnection() {
            return connection;
      }

      /**
       * @return DBConnectionManager
       * @throws StudentAdmissionException
       */
      public static DBConnectionManager getInstance() throws
StudentAdmissionException {

            // Code here
            return new DBConnectionManager();

            //return null;
      }

      /**
       * @throws StudentAdmissionException
       */
      private void loadProperties() throws StudentAdmissionException {
            FileInputStream inputStream = null;
            try {
                  inputStream = new FileInputStream(PROPERTY_FILE);
                  props = new Properties();
                  props.load(inputStream);
            } catch (FileNotFoundException e) {

                  throw new StudentAdmissionException("Database Property
File Not Found", e.getCause());
```

```
            } catch (IOException e) {
                  throw new StudentAdmissionException("Exception during
property file I/O", e.getCause());
            } finally {
                  if (inputStream != null) {
                        try {
                              inputStream.close();
                        } catch (IOException e) {
                              throw new
StudentAdmissionException("Exception during property file I/O",
e.getCause());
                        }
                  }
            }
      }
}
```

StudentAdmission.java

```
/*
 * Don't change this code
 */
package com.cts.unoadm.vo;

import java.util.Date;

public class StudentAdmission {
      String admissionId;
      String studentCode;
      Date dateOfCounseling;
      String departmentName;
      Date dateOfAdmission;
      String preferCollegeHostel;
      String firstGraduate;
      String managerApproval;
      double admissionFee;
      double tuitionFee;
      double hostelFee;
      double totalCollegeFee;
      String finalStatusOfAdmission;

      public StudentAdmission() {
            super();
      }
```

```
	public StudentAdmission(String admissionId, String studentCode,
Date dateOfCounseling, String departmentName,
			Date dateOfAdmission, String preferCollegeHostel,
String firstGraduate, String managerApproval,
			double admissionFee, double tuitionFee, double
hostelFee, double totalCollegeFee,
			String finalStatusOfAdmission) {
		super();
		this.admissionId = admissionId;
		this.studentCode = studentCode;
		this.dateOfCounseling = dateOfCounseling;
		this.departmentName = departmentName;
		this.dateOfAdmission = dateOfAdmission;
		this.preferCollegeHostel = preferCollegeHostel;
		this.firstGraduate = firstGraduate;
		this.managerApproval = managerApproval;
		this.admissionFee = admissionFee;
		this.tuitionFee = tuitionFee;
		this.hostelFee = hostelFee;
		this.totalCollegeFee = totalCollegeFee;
		this.finalStatusOfAdmission = finalStatusOfAdmission;
	}

	public String getAdmissionId() {
		return admissionId;
	}

	public void setAdmissionId(String admissionId) {
		this.admissionId = admissionId;
	}

	public String getStudentCode() {
		return studentCode;
	}

	public void setStudentCode(String studentCode) {
		this.studentCode = studentCode;
	}

	public Date getDateOfCounseling() {
		return dateOfCounseling;
	}

	public void setDateOfCounseling(Date dateOfCounseling) {
		this.dateOfCounseling = dateOfCounseling;
```

```
}

public String getDepartmentName() {
	return departmentName;
}

public void setDepartmentName(String departmentName) {
	this.departmentName = departmentName;
}

public Date getDateOfAdmission() {
	return dateOfAdmission;
}

public void setDateOfAdmission(Date dateOfAdmission) {
	this.dateOfAdmission = dateOfAdmission;
}

public String getPreferCollegeHostel() {
	return preferCollegeHostel;
}

public void setPreferCollegeHostel(String preferCollegeHostel) {
	this.preferCollegeHostel = preferCollegeHostel;
}

public String getFirstGraduate() {
	return firstGraduate;
}

public void setFirstGraduate(String firstGraduate) {
	this.firstGraduate = firstGraduate;
}

public String getManagerApproval() {
	return managerApproval;
}

public void setManagerApproval(String managerApproval) {
	this.managerApproval = managerApproval;
}

public double getAdmissionFee() {
	return admissionFee;
}
```

```
	public void setAdmissionFee(double admissionFee) {
		this.admissionFee = admissionFee;
	}

	public double getTuitionFee() {
		return tuitionFee;
	}

	public void setTuitionFee(double tuitionFee) {
		this.tuitionFee = tuitionFee;
	}

	public double getHostelFee() {
		return hostelFee;
	}

	public void setHostelFee(double hostelFee) {
		this.hostelFee = hostelFee;
	}

	public double getTotalCollegeFee() {
		return totalCollegeFee;
	}

	public void setTotalCollegeFee(double totalCollegeFee) {
		this.totalCollegeFee = totalCollegeFee;
	}

	public String getFinalStatusOfAdmission() {
		return finalStatusOfAdmission;
	}

	public void setFinalStatusOfAdmission(String
finalStatusOfAdmission) {
		this.finalStatusOfAdmission = finalStatusOfAdmission;
	}

	@Override
	public String toString() {
		return "Student Admission Details: [admissionId=" +
admissionId + ", studentCode=" + studentCode + ", dateOfCounseling="
				+ dateOfCounseling + ", departmentName=" +
departmentName + ", dateOfAdmission=" + dateOfAdmission + ",
preferCollegeHostel="
				+ preferCollegeHostel + ", firstGraduate=" +
firstGraduate + ", managerApproval=" + managerApproval
```

```
                    + ", admissionFee=" + admissionFee + ",
tuitionFee=" + tuitionFee + ", hostelFee=" + hostelFee + ",
totalCollegeFee=" + totalCollegeFee
                    + ", finalStatusOfAdmission=" +
finalStatusOfAdmission + "]";
      }

}
```

Database.properties

#IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE

#ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY

#YOU CAN CHANGE THE VALUE OF THE PROPERTY

#LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD using this properties file only.

#Do not hard code the values

drivername=com.mysql.cj.jdbc.Driver

url=jdbc:mysql://localhost:3306/uno_admission

username=root

password=

inputFeed.txt

A001,S001,2020-01-15,EEE,2020-01-25,YES,YES,Approved

A002,S002,2020-02-04,MECH,2020-02-12,N0,YES,Approved

A003,S003,2020-04-21,CSE,2020-05-27,YES,NO,Approved

A004,S004,2020-07-16,IT,2020-07-24,NO,NO,Approved

A005,S005,2020-08-10,ECE,2020-08-11,YES,YES,Approved

A006,S006,2020-09-01,EEE,2020-09-10,YES,NO,Pending

A007,S007,2020-10-19,CIVIL,2020-10-28,N0,YES,Approved

MySQL Queries ::

```
--
-- Table structure for table `students`
--

DROP TABLE IF EXISTS students;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE students (
  admission_id varchar(4) NOT NULL,
  student_code varchar(4) NOT NULL,
  date_of_counseling date DEFAULT NULL,
  department_name varchar(15) NOT NULL,
  date_of_admission date DEFAULT NULL,
  prefer_college_hostel varchar(20) NOT NULL,
  first_graduate varchar(20) NOT NULL,
  manager_approval varchar(15) NOT NULL,
```

```
  admission_fee float(11,2) NOT NULL,
  tution_fee float(11,2) NOT NULL,
  hostel_fee float(11,2) NOT NULL,
  total_college_fee float(11,2) DEFAULT NULL,
  final_status_of_admission varchar(25) NOT NULL,
  PRIMARY KEY (admission_id),
  UNIQUE KEY student_code (student_code)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
/*!40101 SET character_set_client = @saved_cs_client */;

select * from students;
```