Fatima Jinnah Women University

Department Of Software Engineering

PROJECT

Course Title

Database(Lab)

Submitted To

Engr.Shoaib

Submitted By

Mariam Fatima (2021-BSE-020)

Date of Submission

June 22, 2023

Contents

PROJECT SCOPE

The scope of project (IIT Admission System) involves developing a system that manages the admission process for students. The system will handle the following functionalities:

STUDENT REGISTRATION

Allow prospective students to create an account by providing their personal details such as name, email address, and date of birth. Assign a unique student ID to each registered student.

COURSE SELECTION

Display the available database courses, including their field of study, to the students. Allow students to select the courses they wish to apply for admission.

APPLICATION SUBMITTION

Enable students to submit their applications, providing information such as their desired field of study, year of admission, and expected year of graduation. Capture the application date and assign a unique admission ID to each application.

ADMISSION PROCESSING

Provide administrative staff with an interface to review and process the submitted applications. Allow administrators to evaluate each application based on the student's desired field of study and admission details. Update the status of each application (accepted, rejected, pending) based on the evaluation.

SEMESTER MANAGEMENT

Maintain a record of different semesters offered by the university, including their semester ID and current semester indicator. Store information related to fees, discounts, and payment details for each semester.

ENROLLEMENT AND REGISTRATION

Once an application is accepted, facilitate the enrollment process for the student into the desired database course. Assign the student to the appropriate semester based on their selected year of admission and graduation. Record the student's course registration details, including the semester, course name, and other relevant information.

FEE COLLECTION AND MANAGEMENT

Calculate the fees for each student based on the selected semester, course, and any applicable discounts. Provide a mechanism for students to view and pay their fees. Keep track of fee payment status and update the system accordingly.

REPORTING AND ANALYTICS

Generate reports on the admission process, including the number of applications received, admission statistics, and enrollment trends. Provide analytics on fee collection, discounts offered.

SECURITY AND AUTHENTICATION

Implement secure user authentication to protect student and administrative data. Ensure appropriate access controls to safeguard sensitive information.

USER FRIENDLY INTERFACE

Design an intuitive and user-friendly interface for both students and administrative staff. Provide easy navigation, search functionality, and clear progress indicators to enhance user experience.

GOALS AND OBJECTIVES

The goal of the IIT Admission System project is to develop a comprehensive and efficient system that facilitates the admission process. The project's primary objectives include:

STREAMLINE THE ADMISSION PROCESS

The system aims to automate and streamline the admission process for the database course, reducing manual effort, paperwork, and potential errors.

IMPROVE EFFICIENCY AND ACCURACY

The project aims to enhance the efficiency and accuracy of the admission process. By automating various tasks such as application management, evaluation, and fee calculation, the system reduces the time and effort required by administrative staff, allowing them to focus on higher-value activities.

ENHANCE USER EXPERIENCE

The project strives to provide a user-friendly interface for both students and administrative staff.

ENSURE DATA SEURITY AND PRIVACY

The system emphasizes the security and privacy of student and administrative data. By implementing robust authentication mechanisms, access controls, and data encryption, the project aims to protect sensitive information from unauthorized access or breaches.

ENABLE EFFECTIVE DECISION-MAKING

The project intends to provide administrators with the necessary tools and insights to make informed decisions regarding admission. By generating reports, analytics, and admission statistics, the system enables administrators to analyze trends, evaluate the effectiveness of the admission process, and make data-driven decisions.

BASICS FEATURES

Some basic features for this system (II Admission system) are:

- Student Admission
- Course Selection
- Semester Management
- Fee Calculation and Management

INTRODUCTION

The II University Admission System is a comprehensive platform designed to streamline the admission process, manage semesters, and maintain student records effectively. This system caters to the needs of a university by providing a centralized and automated solution for handling admissions, semesters, and student information. With its robust features and user-friendly interface, the II University Admission System aims to enhance efficiency, transparency, and convenience for both university administrators and students.

FIRST ENTITY

The primary entity of the system is the Admission module, which encompasses all aspects related to the admission process. It includes essential attributes such as admission ID, field of study, year of admission, and year of graduation. These attributes enable the system to uniquely identify each admission and associate it with specific academic details, ensuring accurate record-keeping and easy retrieval of information.

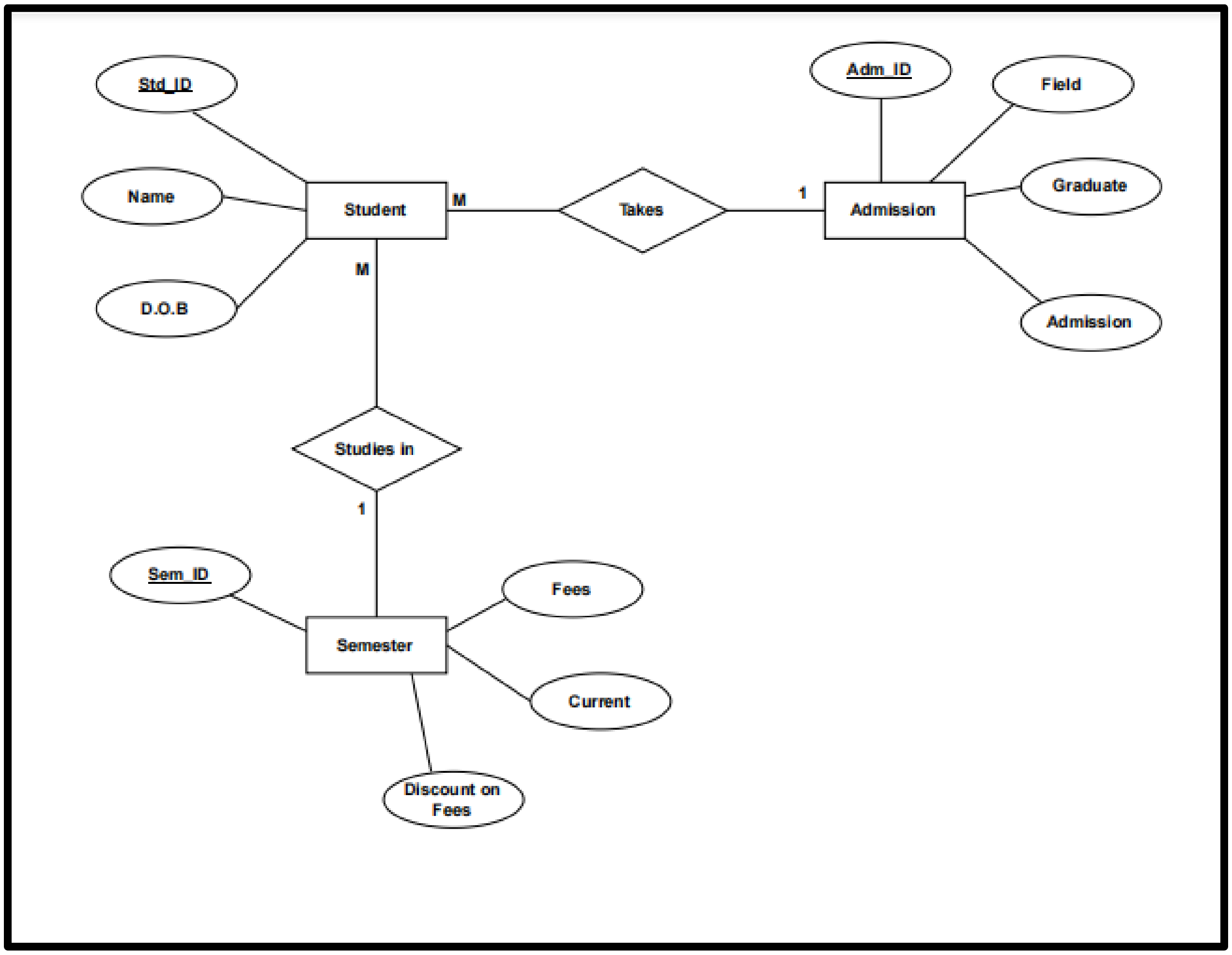
SECOND ENTITY

The second entity within the system is the Semester module, which focuses on managing the various semesters throughout a student's academic journey. Each semester is assigned a unique semester ID and includes attributes like current discount and fees. This allows for efficient tracking of financial details and ensures that the system can calculate accurate fees based on the student's current semester.

THIRD ENTITY

The third entity in the system is the Student module, which stores comprehensive information about each enrolled student. It includes attributes such as student ID, student name, and date of birth. These attributes enable the system to maintain personalized student profiles and facilitate easy identification of students across different modules and functionalities. The II University Admission System integrates these three entities to create a cohesive and holistic solution for university management. It provides administrators with the ability to track and manage admissions, monitor student progress through different semesters, and maintain accurate student records throughout their academic journey. Additionally, the system offers students a user-friendly portal to access their admission and semester-related information, making it easier for them to stay updated with their academic details.

ER-DIAGRAM

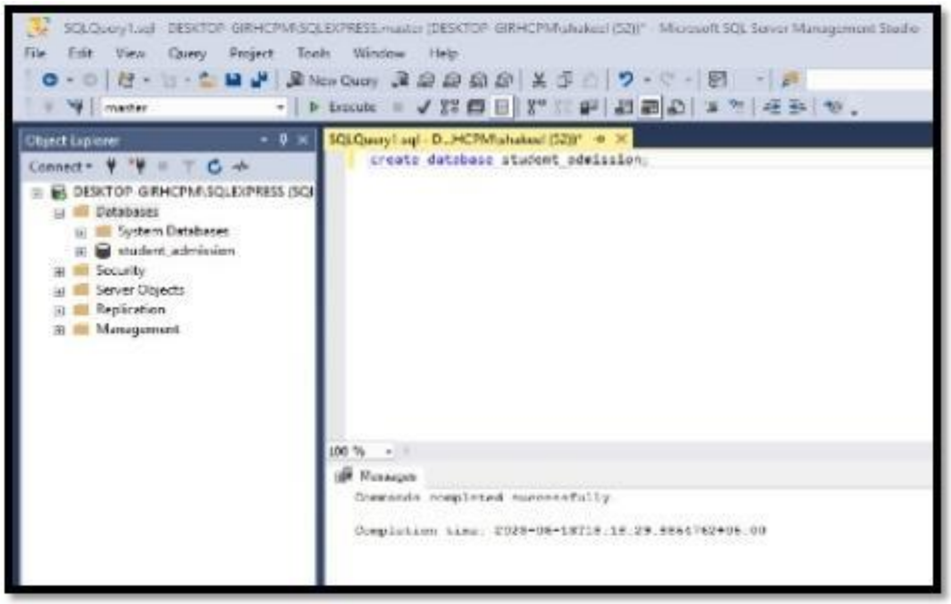

Std_ID
Name
D.O.B
Student
M
Takes
1
Admission
Adm_ID
Field
Graduate
Admission
M
Studies in
1
Sem_ID
Semester
Fees
Current
Discount on Fees

CODE AND QUERIES

CREATION OF DATABASE

For the creation of our database in SQL Server Management Studio, we will need to write queries for the creation of the database and tables. The tables will be created keeping the ER diagram in mind. The data will be inserted from a dataset, downloaded online related to our database. After insertion, we will normalize our data for a good an efficient database, which will lead to an updated ER diagram. We will make needed changed in our database with the help of the updated ER diagram. Different interfaces will be created for different users who will be using the system.

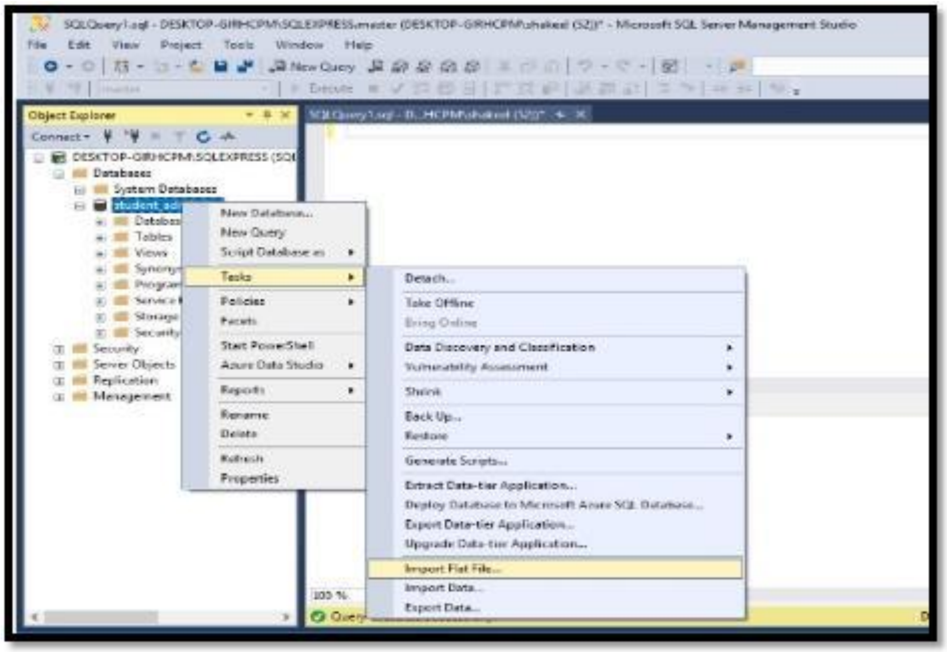
CREATE DATABASE

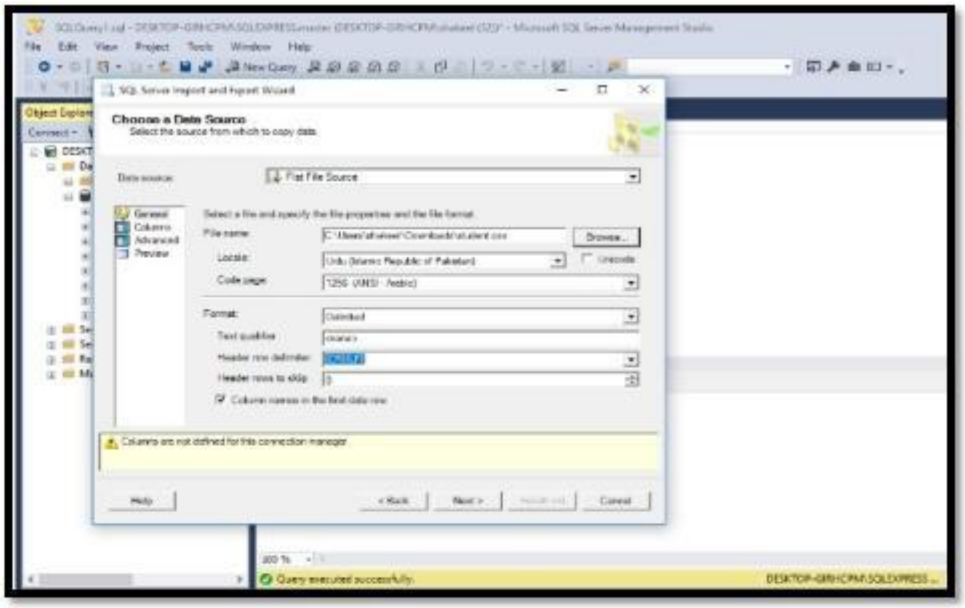


```
create database student_admission;
```

```
Commands completed successfully.

Completion time: 2020-08-18T18:18:29.8861762+05:00
```

IMPORT STUDENT DATA




SQL Server Import and Export Wizard
Choose a Data Source
Select the source from which to copy data.
General
Columns
Advanced
Preview
Browse...
Unicode
Help
< Back
Next >
Finish >>|
Cancel
Query executed successfully.
DESKTOP-GIRHCPM\SQLEXPRESS ...

Data source: Flat File Source

Select a file and specify the file properties and the file format.

File name: C:\Users\shakeel\Downloads\student.csv

Locale: Urdu (Islamic Republic of Pakistan)

Code page: 1256 (ANSI - Arabic)

Format: Delimited

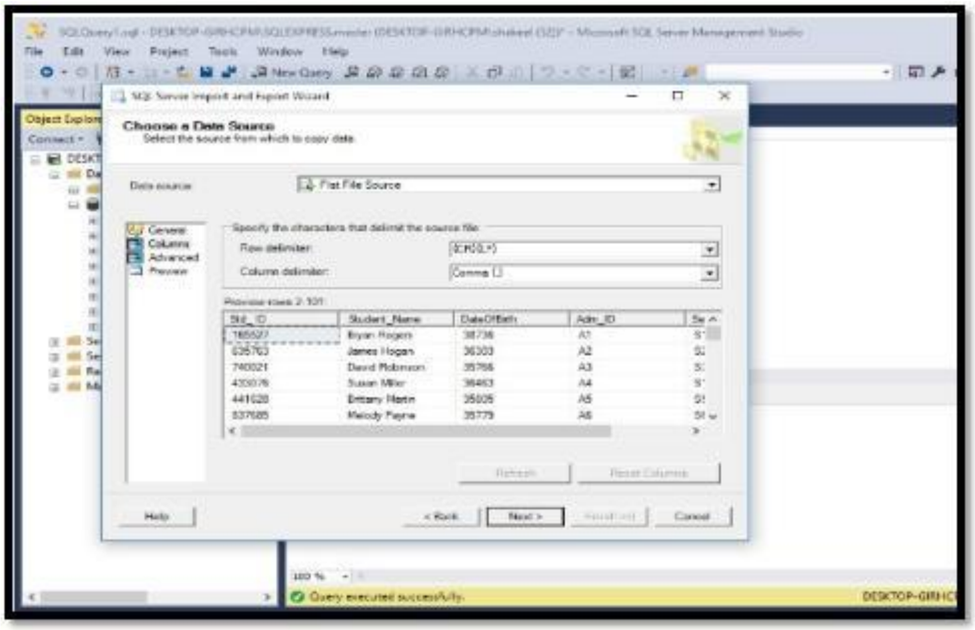
Text qualifier: <none>

Header row delimiter: {CR}{LF}

Header rows to skip: 0

☑ Column names in the first data row

Columns are not defined for this connection manager.



Choose a Data Source
Select the source from which to copy data.

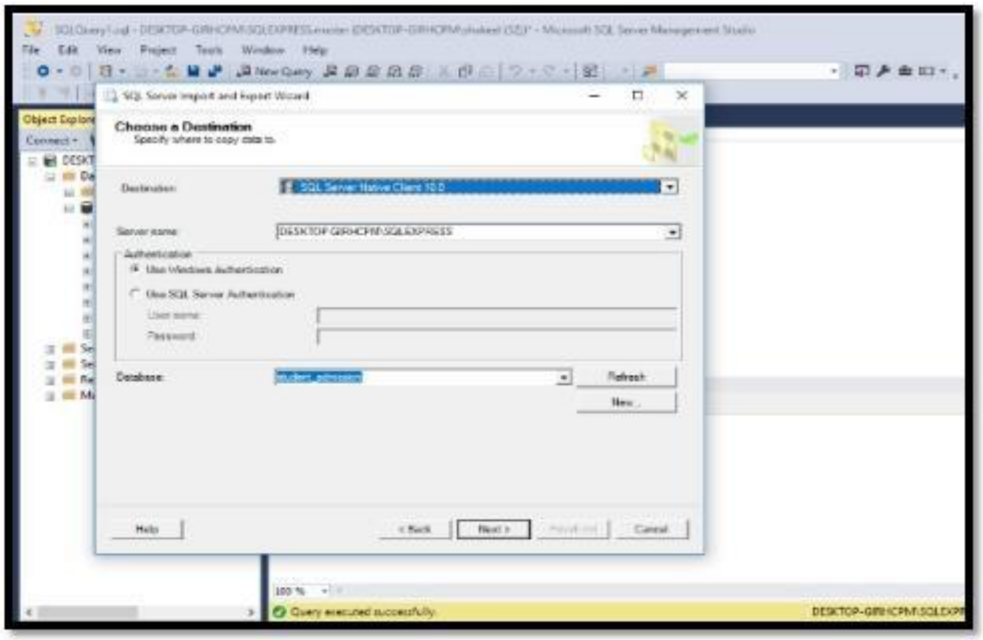
Data source: Flat File Source

Specify the characters that delimit the source file:
Row delimiter: {CR}{LF}
Column delimiter: Comma {,}

Preview rows 2-101:

Std_ID	Student_Name	DateOfBirth	Adm_ID
165527	Bryan Rogers	38736	A1
635763	James Hogan	36303	A2
740021	David Robinson	35766	A3
420076	Susan Miller	38463	A4
441028	Brittany Martin	35005	A5
837685	Melody Payne	35779	A6

Query executed successfully.



Choose a Destination
Specify where to copy data to.

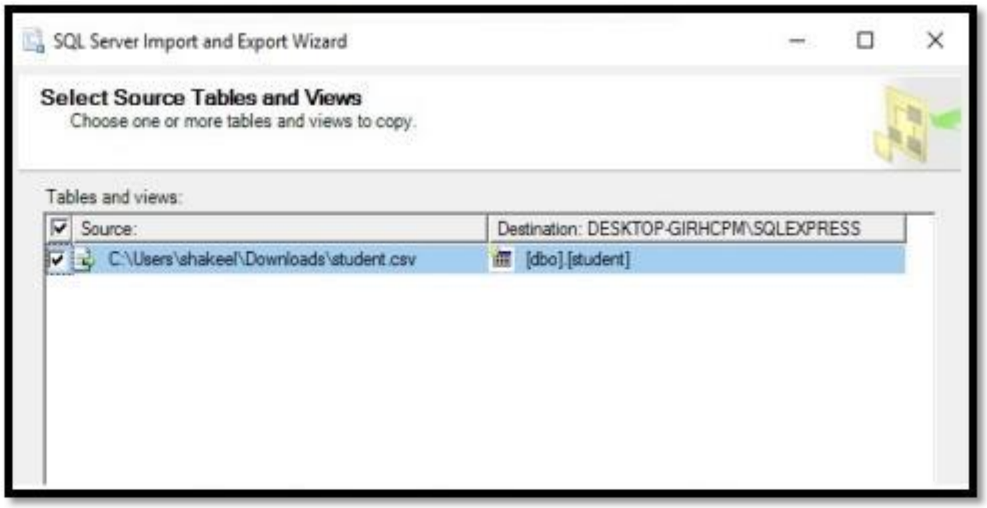
Destination: SQL Server Native Client 10.0

Server name: DESKTOP-GIRHCPM\SQLEXPRESS

Authentication
(●) Use Windows Authentication
( ) Use SQL Server Authentication
User name:
Password:

Database: student_admission

Query executed successfully.



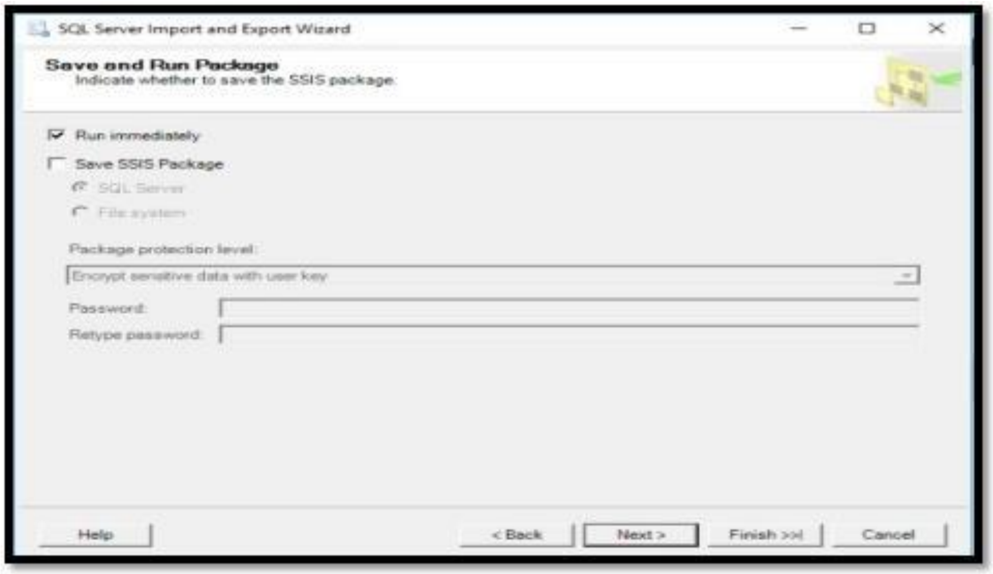
SQL Server Import and Export Wizard

Select Source Tables and Views

Choose one or more tables and views to copy.

Tables and views:

✓	Source:	Destination: DESKTOP-GIRHCPM\SQLEXPRESS
✓	C:\Users\shakeel\Downloads\student.csv	[dbo].[student]



SQL Server Import and Export Wizard

Save and Run Package

Indicate whether to save the SSIS package.

[✓] Run immediately

[ ] Save SSIS Package

(●) SQL Server

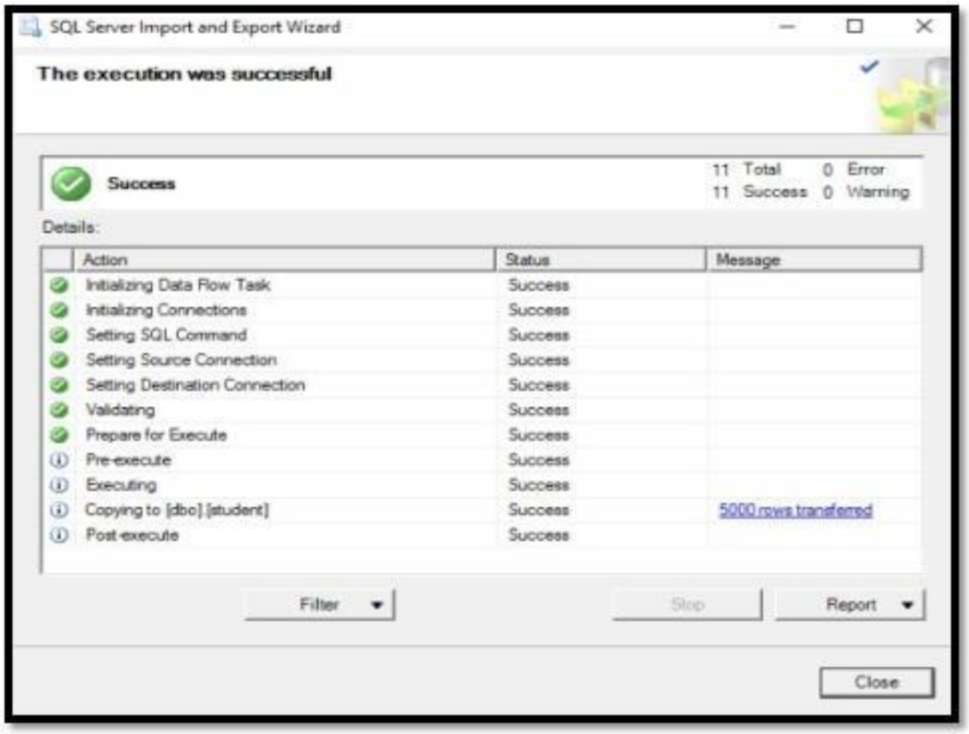
( ) File system

Package protection level:

Encrypt sensitive data with user key

Password:

Retype password:



The execution was successful

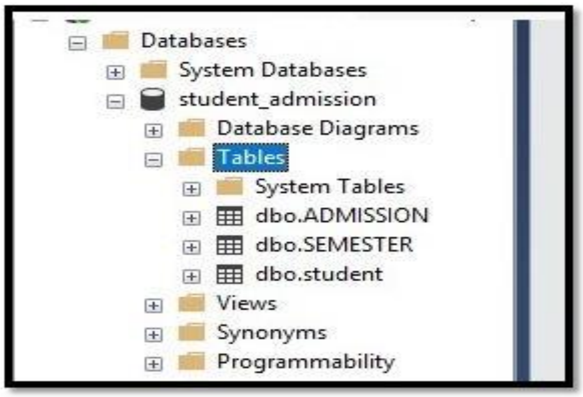
Success

11 Total 0 Error
11 Success 0 Warning

Details:

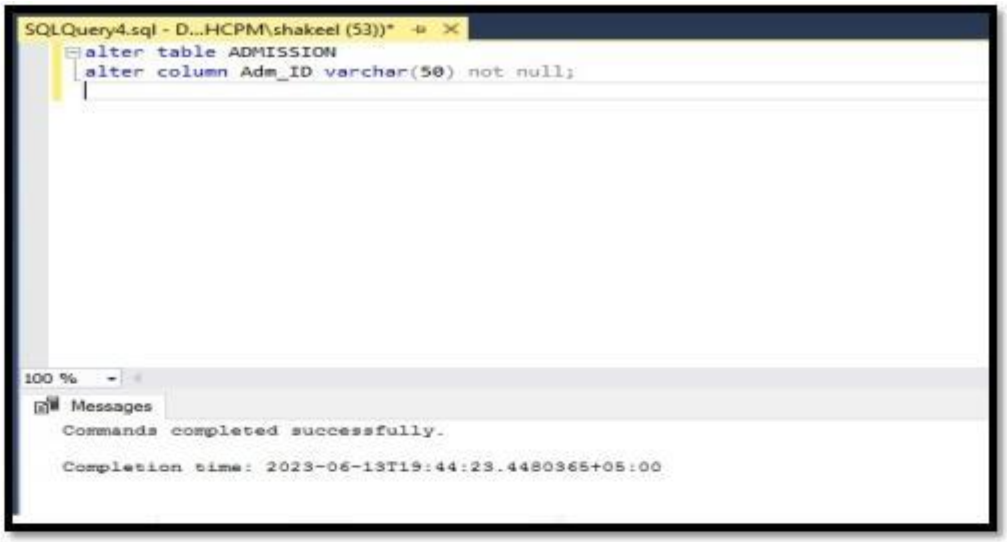
Action	Status	Message
Initializing Data Flow Task	Success	
Initializing Connections	Success	
Setting SQL Command	Success	
Setting Source Connection	Success	
Setting Destination Connection	Success	
Validating	Success	
Prepare for Execute	Success	
Pre-execute	Success	
Executing	Success	
Copying to [dbo].[student]	Success	5000 rows transferred
Post-execute	Success	

Similarly we import ADMISSION and Semester Table.



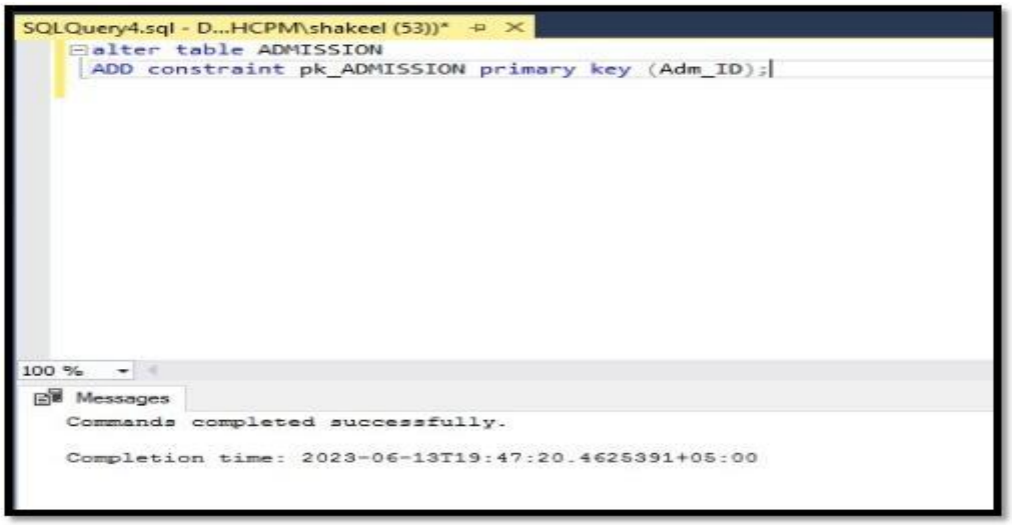
ADD CONSTRAINT PRIMARY KEY AND SET COLUMN TO NULL

ADMISSION TABLE



```
alter table ADMISSION
alter column Adm_ID varchar(50) not null;
```

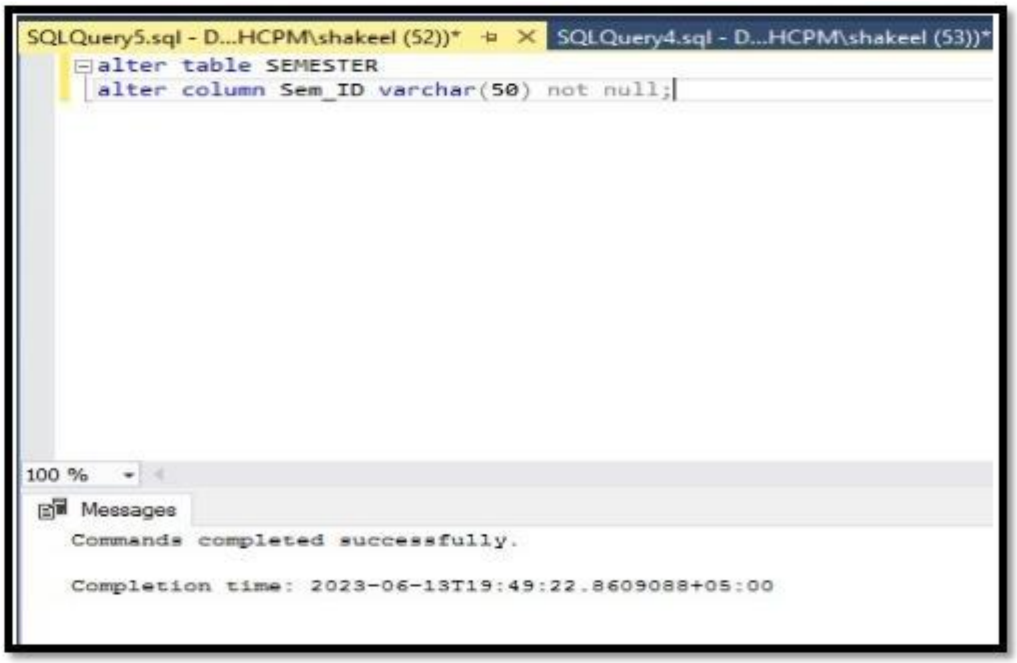
```
Commands completed successfully.

Completion time: 2023-06-13T19:44:23.4480365+05:00
```



```
alter table ADMISSION
ADD constraint pk_ADMISSION primary key (Adm_ID);
```

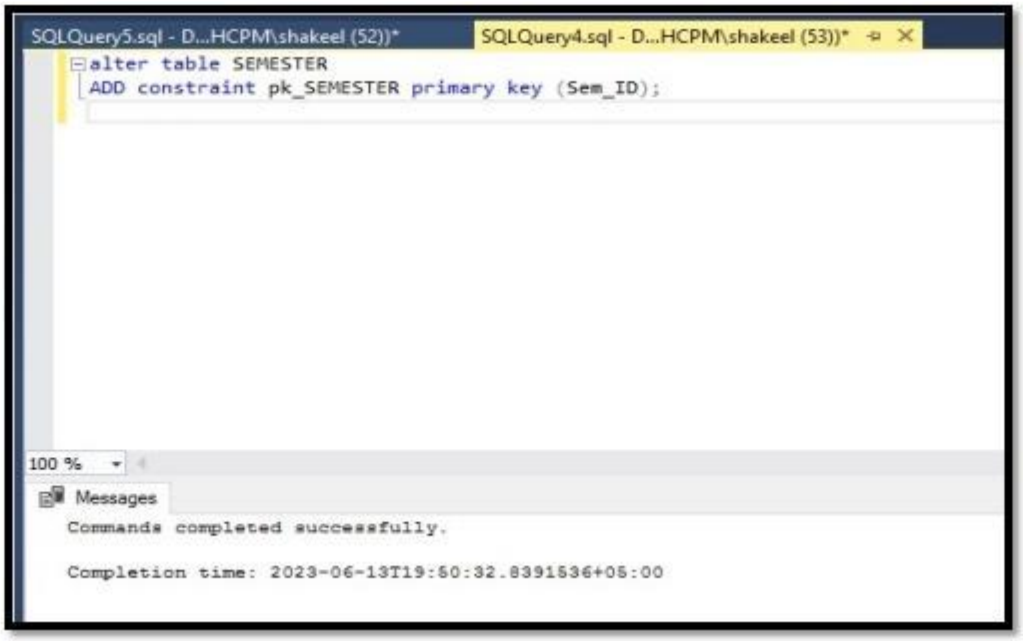
```
Commands completed successfully.

Completion time: 2023-06-13T19:47:20.4625391+05:00
```

SEMESTER TABLE


SQLQuery5.sql - D...HCPM\shakeel (52))*
SQLQuery4.sql - D...HCPM\shakeel (53))*
100 %
Messages

```
alter table SEMESTER
alter column Sem_ID varchar(50) not null;
```

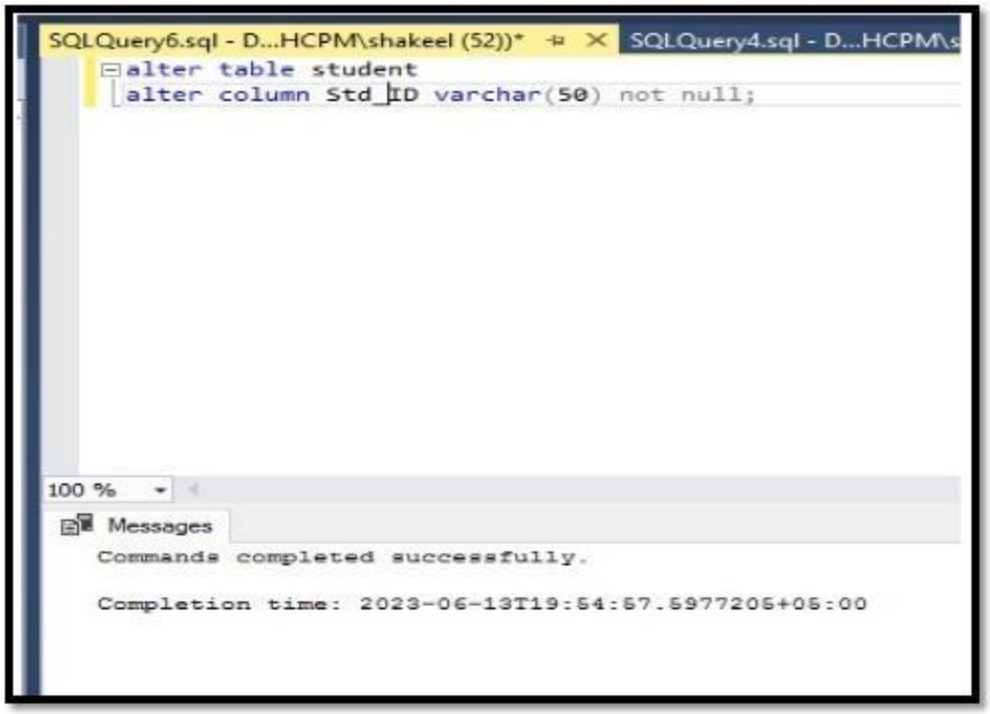
```
Commands completed successfully.

Completion time: 2023-06-13T19:49:22.8609088+05:00
```


SQLQuery5.sql - D...HCPM\shakeel (52))*
SQLQuery4.sql - D...HCPM\shakeel (53))*
100 %
Messages

```
alter table SEMESTER
ADD constraint pk_SEMESTER primary key (Sem_ID);
```

```
Commands completed successfully.

Completion time: 2023-06-13T19:50:32.8391536+05:00
```

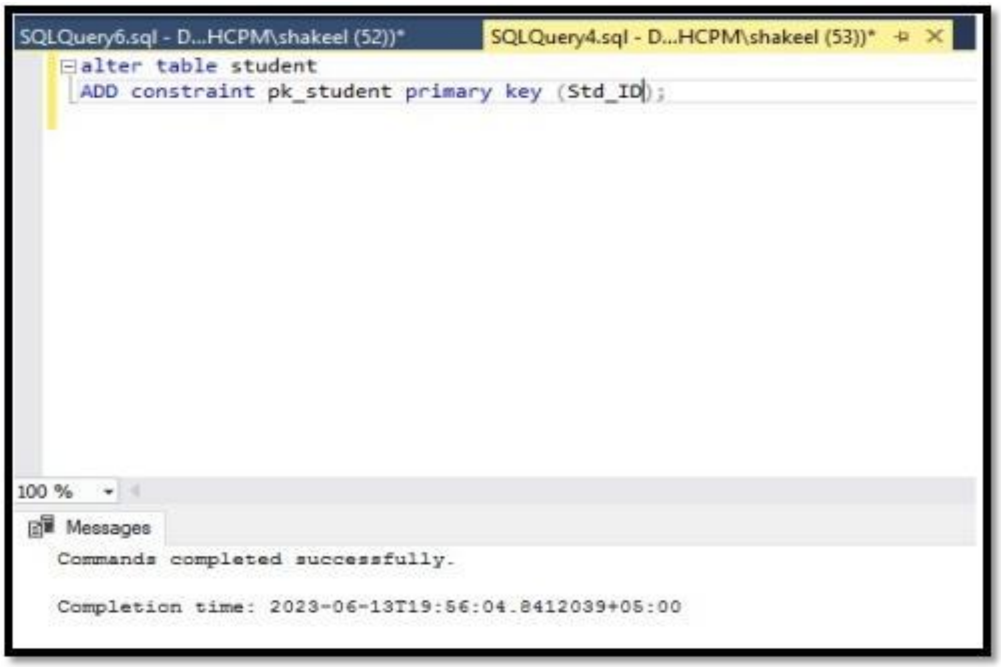
STUDENT TABLE



SQLQuery6.sql - D...HCPM\shakeel (52))*
SQLQuery4.sql - D...HCPM\s
100 %
Messages

```
alter table student
alter column Std_ID varchar(50) not null;
```

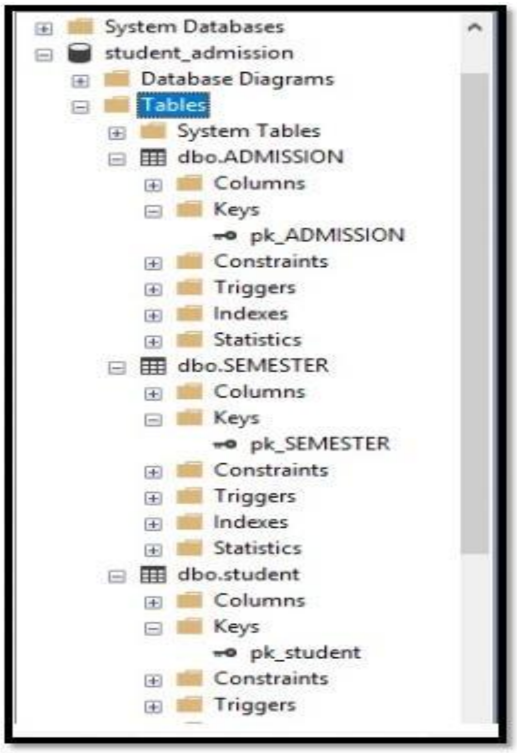
```
Commands completed successfully.

Completion time: 2023-06-13T19:54:57.5977205+05:00
```



SQLQuery6.sql - D...HCPM\shakeel (52))*
SQLQuery4.sql - D...HCPM\shakeel (53))*
100 %
Messages

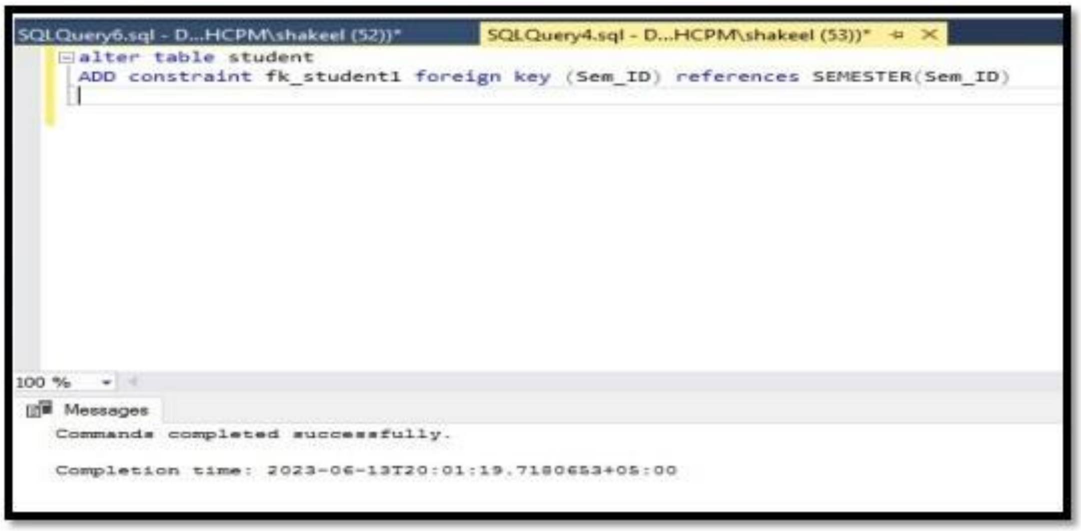
```
alter table student
ADD constraint pk_student primary key (Std_ID);
```

```
Commands completed successfully.

Completion time: 2023-06-13T19:56:04.8412039+05:00
```


System Databases
student_admission
Database Diagrams
Tables
System Tables
dbo.ADMISSION
Columns
Keys
pk_ADMISSION
Constraints
Triggers
Indexes
Statistics
dbo.SEMESTER
Columns
Keys
pk_SEMESTER
Constraints
Triggers
Indexes
Statistics
dbo.student
Columns
Keys
pk_student
Constraints
Triggers

ADD CONSTRAINT FOREIGN KEY AND SET RELATIONSHIPS

SEMESTER TABLE



SQLQuery6.sql - D...HCPM\shakeel (52))* SQLQuery4.sql - D...HCPM\shakeel (53))*

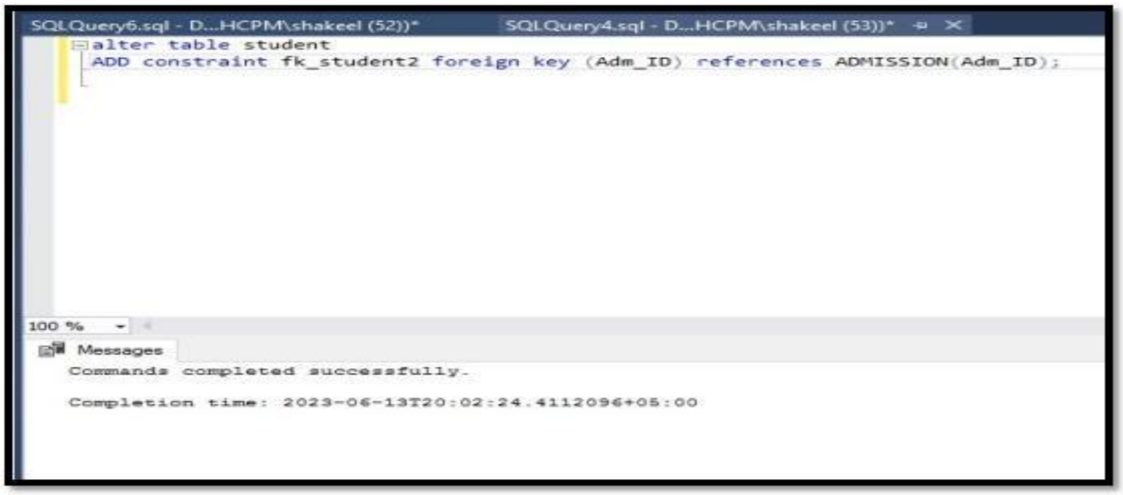
```
alter table student
ADD constraint fk_student1 foreign key (Sem_ID) references SEMESTER(Sem_ID)
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-06-13T20:01:19.7180653+05:00
```

ADMISSION TABLE



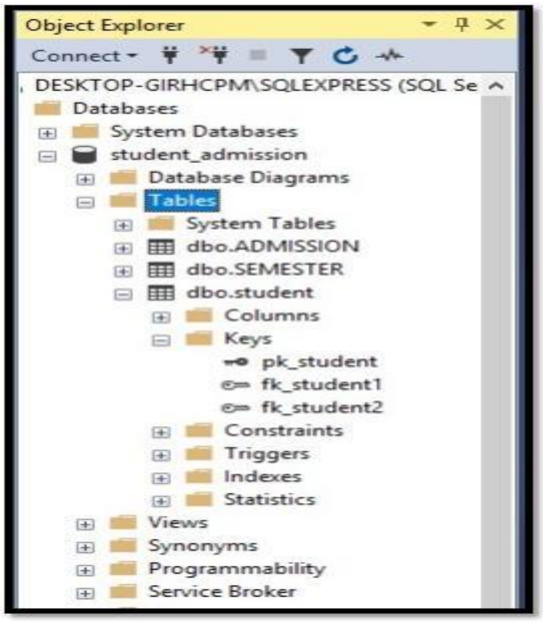
SQLQuery6.sql - D...HCPM\shakeel (52))* SQLQuery4.sql - D...HCPM\shakeel (53))*

```
alter table student
ADD constraint fk_student2 foreign key (Adm_ID) references ADMISSION(Adm_ID);
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-06-13T20:02:24.4112096+05:00
```



RELATIONSHIPS

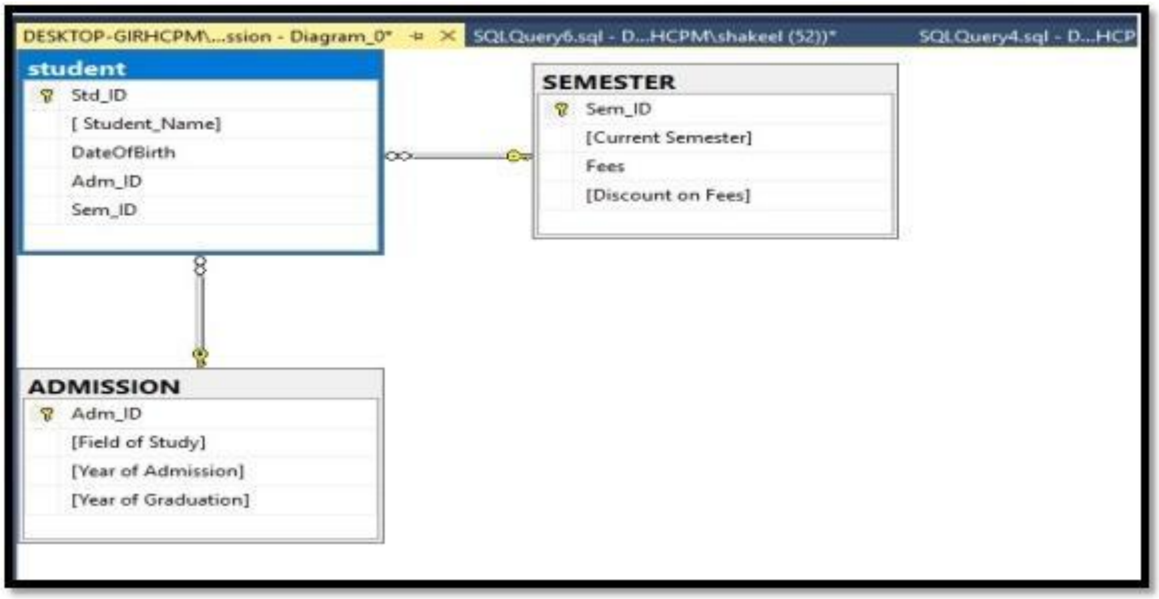
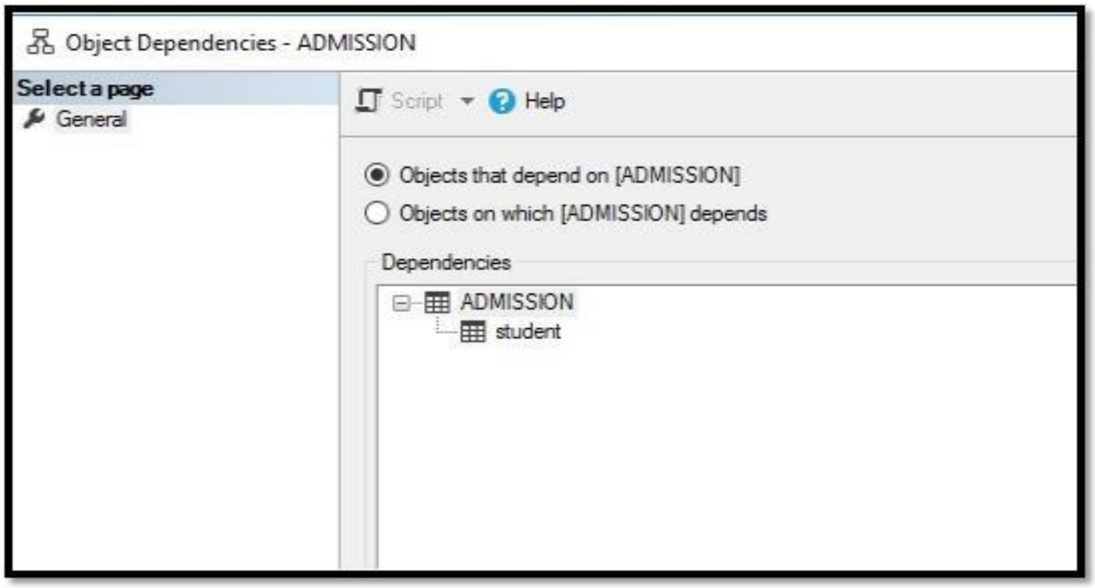


TABLE DEPENDENCIES

ADMISSION TABLE


Object Dependencies - ADMISSION

Select a page
General

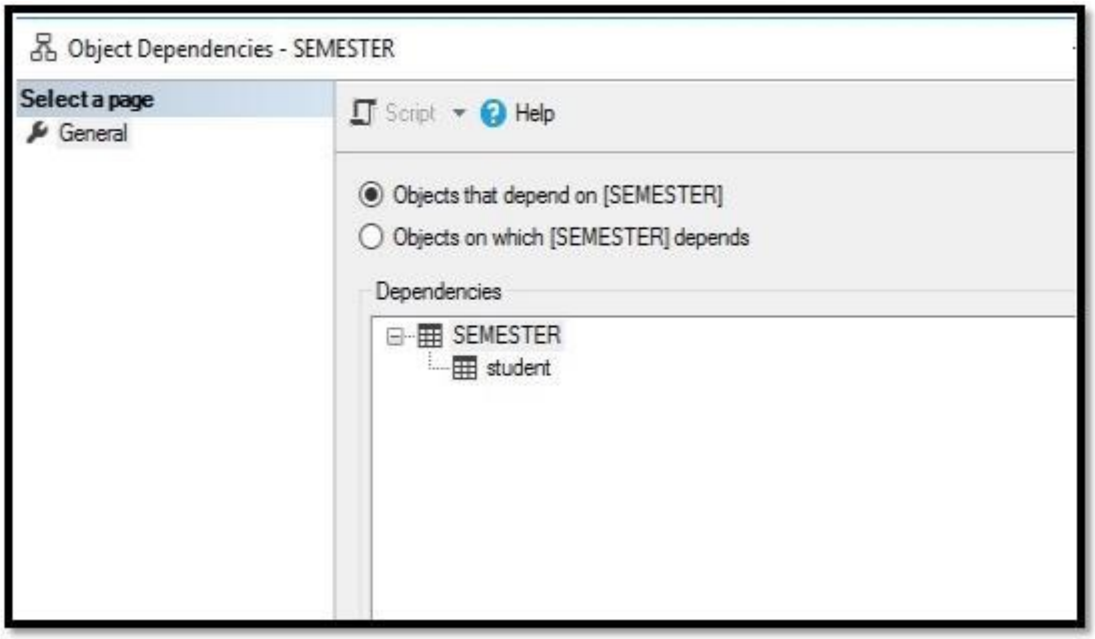
Script Help

(●) Objects that depend on [ADMISSION]
( ) Objects on which [ADMISSION] depends

Dependencies

```
ADMISSION
  student
```

SEMESTER TABLE


Object Dependencies - SEMESTER

Select a page
General

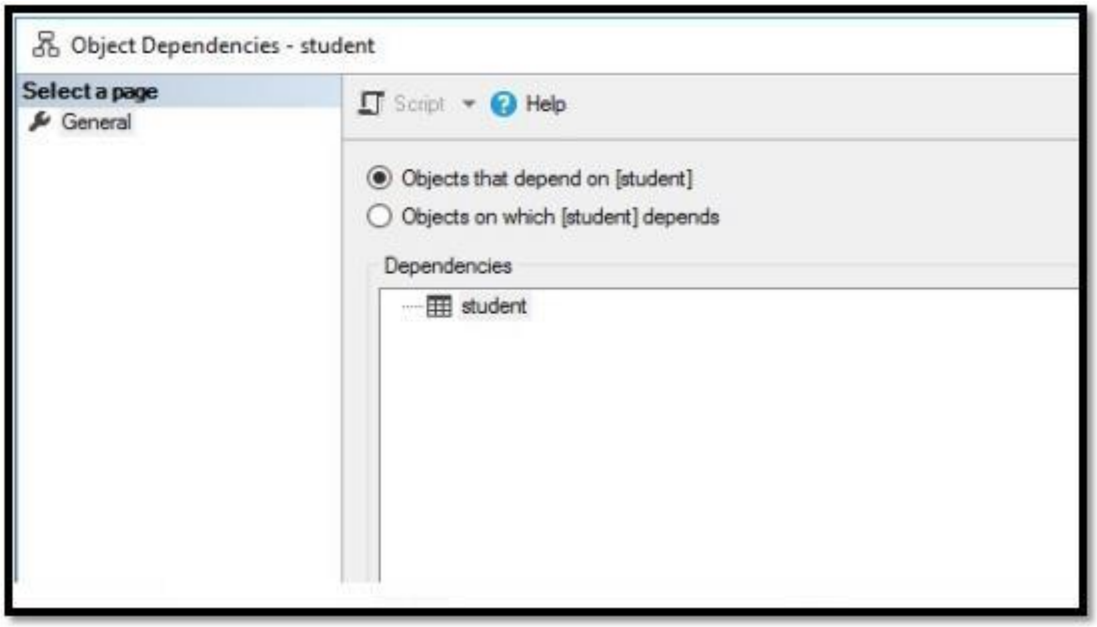
Script Help

(●) Objects that depend on [SEMESTER]
( ) Objects on which [SEMESTER] depends

Dependencies

```
SEMESTER
  student
```

STUDENT TABLE


Object Dependencies - student

Select a page
General

Script Help

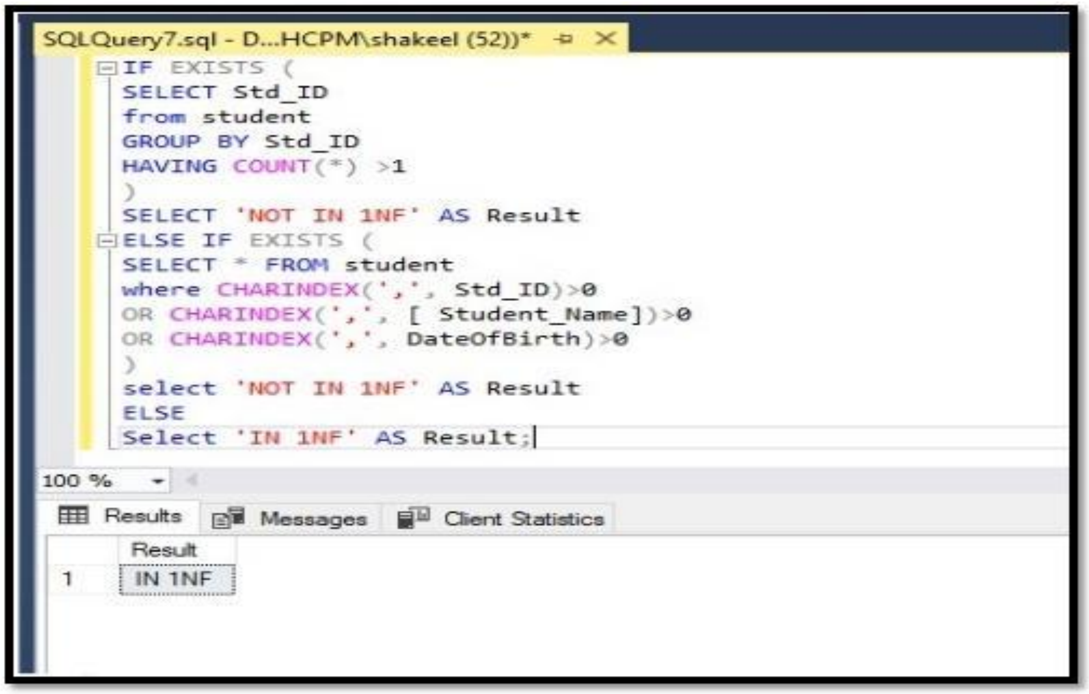
(● Objects that depend on [student]) ○ Objects on which [student] depends

Dependencies
student

SQL NORMALIZATION THROUGH QUERIES

FIRST NORMAL FORM

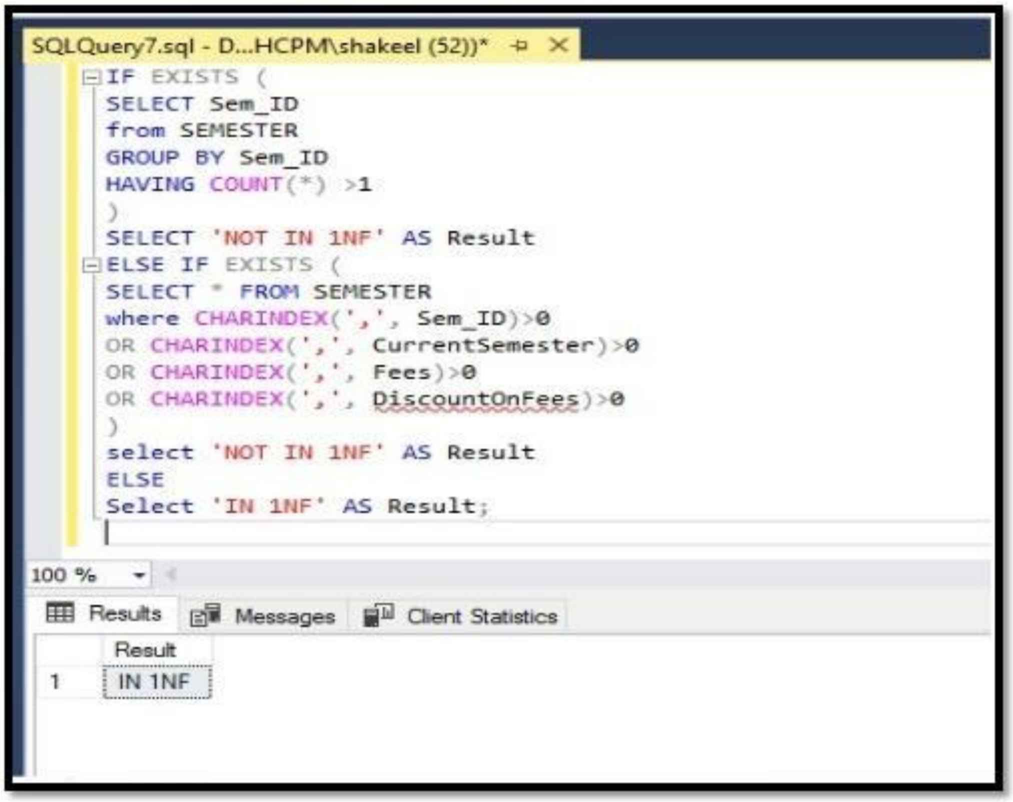
STUDENT TABLE



```
IF EXISTS (
SELECT Std_ID
from student
GROUP BY Std_ID
HAVING COUNT(*) >1
)
SELECT 'NOT IN 1NF' AS Result
ELSE IF EXISTS (
SELECT * FROM student
where CHARINDEX(',', Std_ID)>0
OR CHARINDEX(',', [ Student_Name])>0
OR CHARINDEX(',', DateOfBirth)>0
)
select 'NOT IN 1NF' AS Result
ELSE
Select 'IN 1NF' AS Result;
```

```
   Result
1  IN 1NF
```

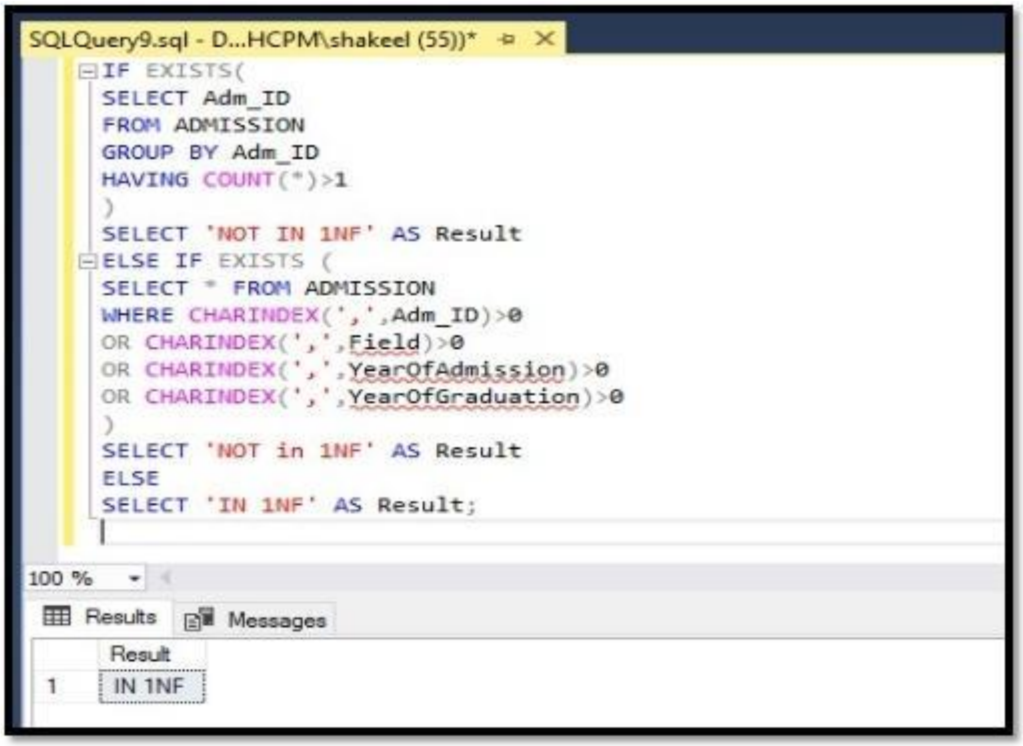
SEMESTER TABLE



```
IF EXISTS (
SELECT Sem_ID
from SEMESTER
GROUP BY Sem_ID
HAVING COUNT(*) >1
)
SELECT 'NOT IN 1NF' AS Result
ELSE IF EXISTS (
SELECT * FROM SEMESTER
where CHARINDEX(',', Sem_ID)>0
OR CHARINDEX(',', CurrentSemester)>0
OR CHARINDEX(',', Fees)>0
OR CHARINDEX(',', DiscountOnFees)>0
)
select 'NOT IN 1NF' AS Result
ELSE
Select 'IN 1NF' AS Result;
```

```
  Result
1 IN 1NF
```

ADMISSION TABLE

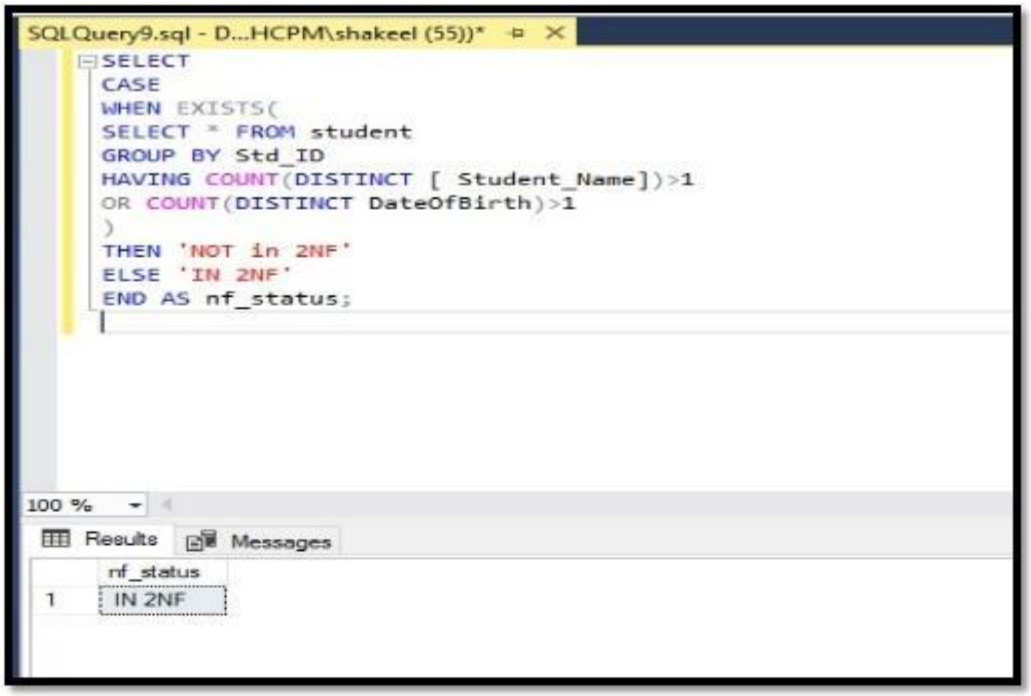


```
IF EXISTS(
SELECT Adm_ID
FROM ADMISSION
GROUP BY Adm_ID
HAVING COUNT(*)>1
)
SELECT 'NOT IN 1NF' AS Result
ELSE IF EXISTS (
SELECT * FROM ADMISSION
WHERE CHARINDEX(',',Adm_ID)>0
OR CHARINDEX(',',Field)>0
OR CHARINDEX(',',YearOfAdmission)>0
OR CHARINDEX(',',YearOfGraduation)>0
)
SELECT 'NOT in 1NF' AS Result
ELSE
SELECT 'IN 1NF' AS Result;
```

```
  Result
1 IN 1NF
```

SECOND NORMAL FORM

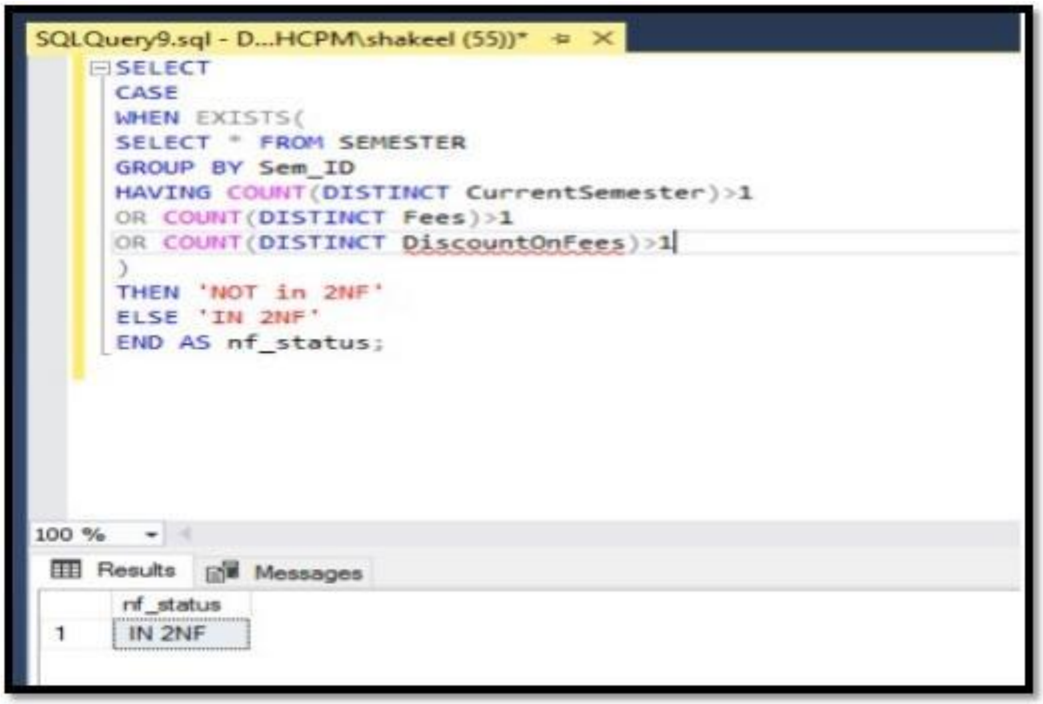
STUDENT TABLE


SQLQuery9.sql - D...HCPM\shakeel (55))*

```
SELECT
CASE
WHEN EXISTS(
SELECT * FROM student
GROUP BY Std_ID
HAVING COUNT(DISTINCT [ Student_Name])>1
OR COUNT(DISTINCT DateOfBirth)>1
)
THEN 'NOT in 2NF'
ELSE 'IN 2NF'
END AS nf_status;
```

```
   nf_status
1  IN 2NF
```

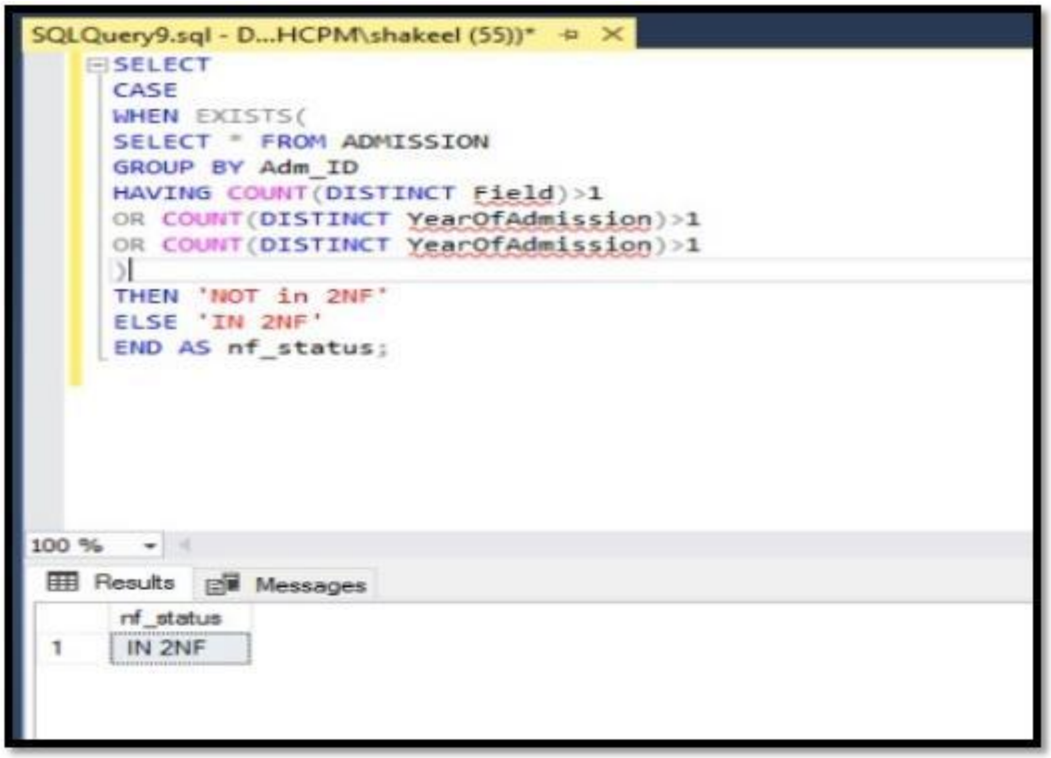
SEMESTER TABLE


SQLQuery9.sql - D...HCPM\shakeel (55))*

```
SELECT
CASE
WHEN EXISTS(
SELECT * FROM SEMESTER
GROUP BY Sem_ID
HAVING COUNT(DISTINCT CurrentSemester)>1
OR COUNT(DISTINCT Fees)>1
OR COUNT(DISTINCT DiscountOnFees)>1
)
THEN 'NOT in 2NF'
ELSE 'IN 2NF'
END AS nf_status;
```

```
   nf_status
1  IN 2NF
```

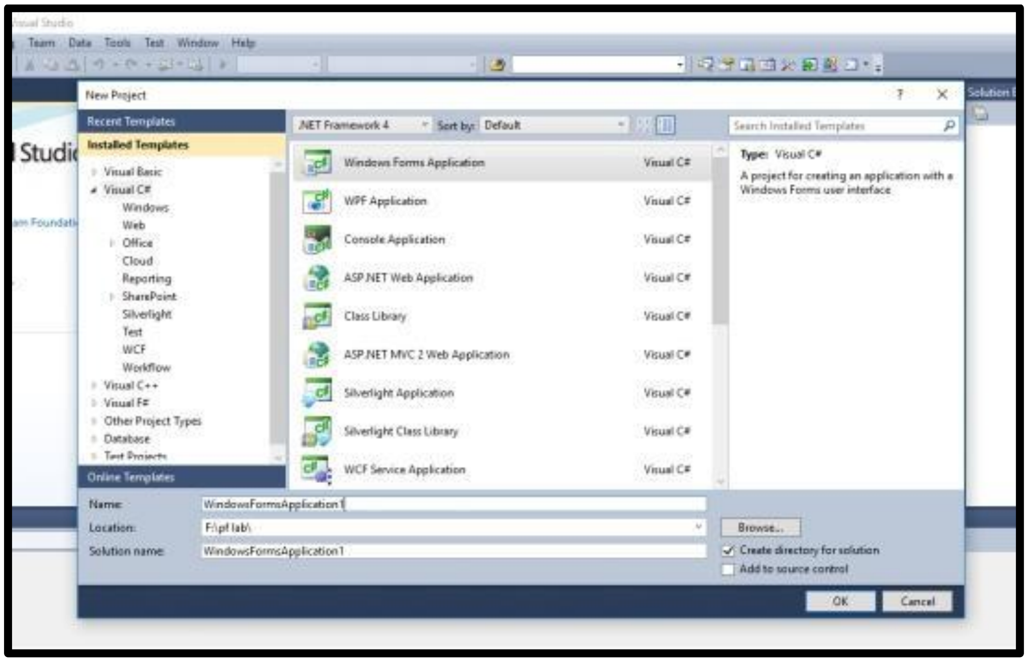
ADMISSION TABLE



```
SELECT
CASE
WHEN EXISTS(
SELECT * FROM ADMISSION
GROUP BY Adm_ID
HAVING COUNT(DISTINCT Field)>1
OR COUNT(DISTINCT YearOfAdmission)>1
OR COUNT(DISTINCT YearOfAdmission)>1
)
THEN 'NOT in 2NF'
ELSE 'IN 2NF'
END AS nf_status;
```

	nf_status
1	IN 2NF

CONNECTING MICROSOFT SQL MANAGEMENT STUDIO TO MICROSOFT VISUAL STUDIO


Team Data Tools Test Window Help
New Project
Recent Templates
Installed Templates
Visual Basic
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
Other Project Types
Database
Test Projects
Online Templates
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
ASP.NET MVC 2 Web Application Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
WCF Service Application Visual C#
Browse...
OK
Cancel

.NET Framework 4 — Sort by: Default

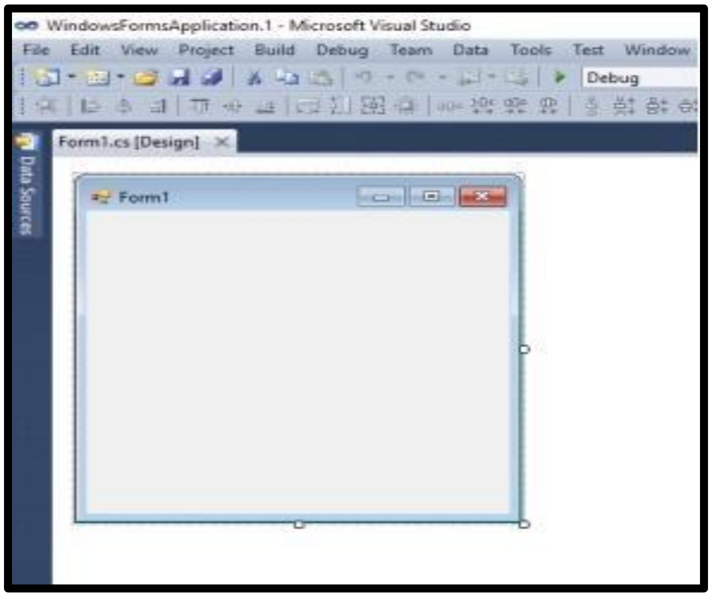
Type: Visual C#
A project for creating an application with a Windows Forms user interface

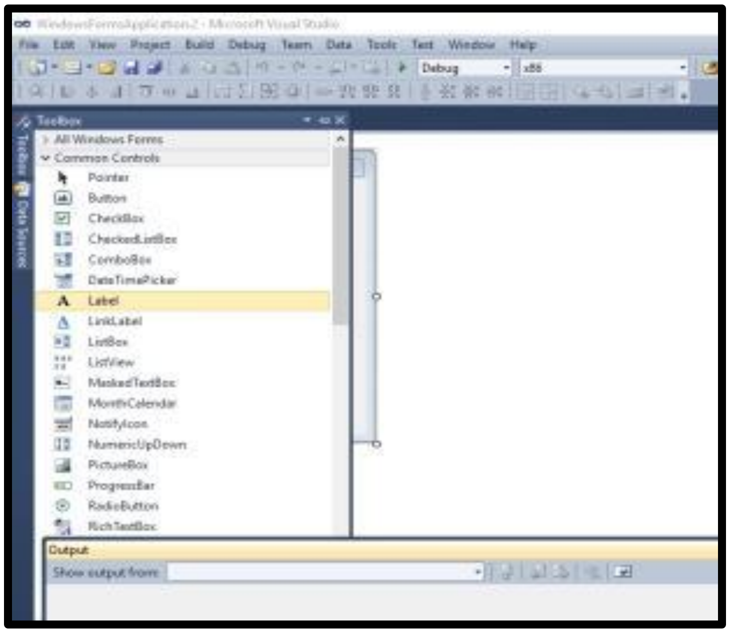
Name: WindowsFormsApplication1
Location: F:\pf lab\
Solution name: WindowsFormsApplication1
✓ Create directory for solution
☐ Add to source control

CREATING FORM

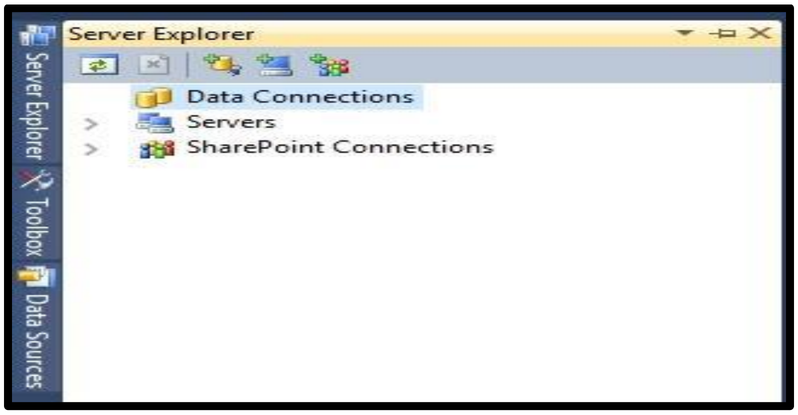
- Creating forms(buttons , labels ,text boxes)
- Grid view is also created in this step.

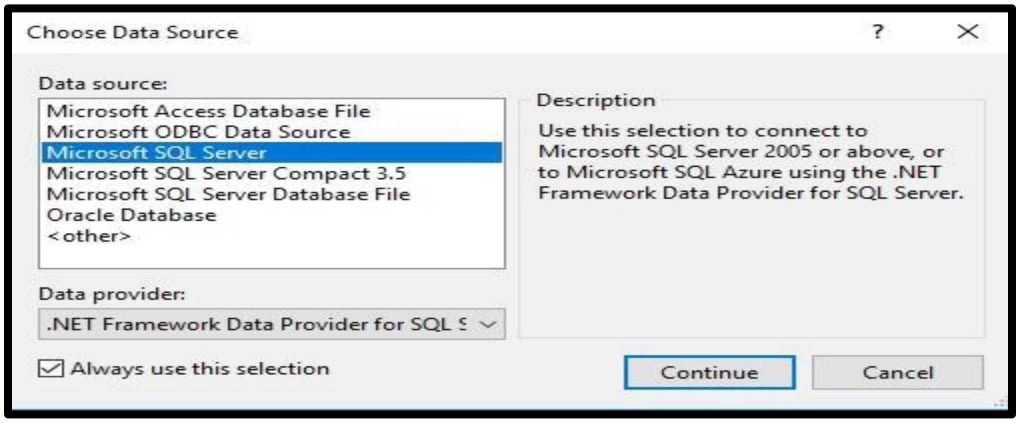
In **Data Grid View Tasks**, it is connected to the source database. Then a connection is established and objects to be used in the grid are selected.


WindowsFormsApplication.1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window
Debug
Data Sources
Form1.cs [Design]
Form1


WindowsFormsApplication2 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
x86
Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
Output
Show output from:

CONNECT TO DATABASE




Choose Data Source

Data source:

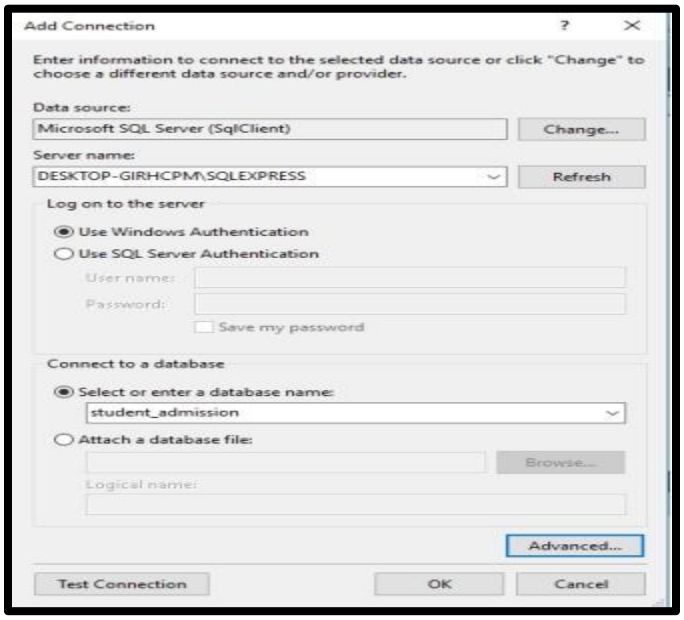
- Microsoft Access Database File
- Microsoft ODBC Data Source
- Microsoft SQL Server
- Microsoft SQL Server Compact 3.5
- Microsoft SQL Server Database File
- Oracle Database
- <other>

Description

Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.

Data provider: .NET Framework Data Provider for SQL S

[✓] Always use this selection


Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: DESKTOP-GIRHCPM\SQLEXPRESS

Log on to the server

- [✓] Use Windows Authentication
- Use SQL Server Authentication

User name:

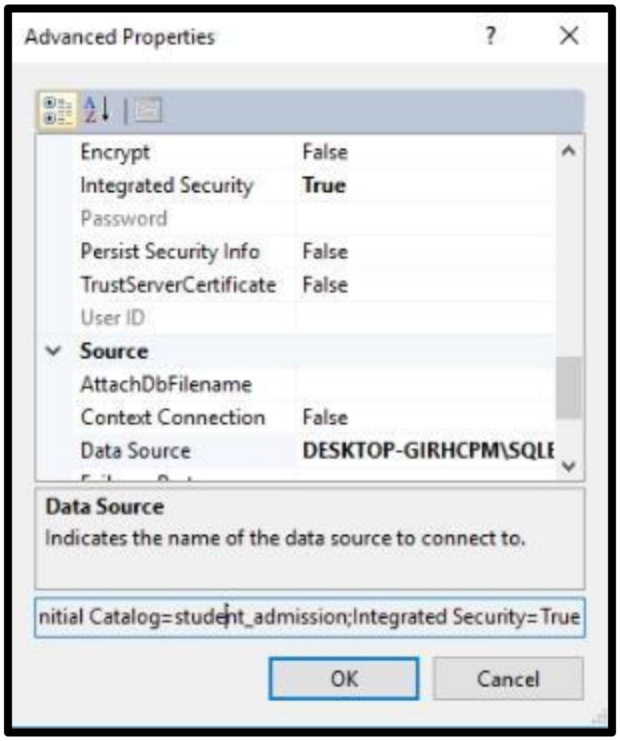
Password:

Save my password

Connect to a database

- [✓] Select or enter a database name: student_admission
- Attach a database file:

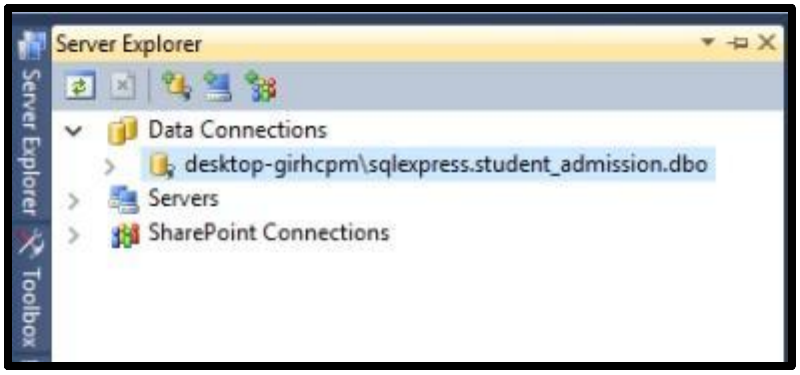
Logical name:



Encrypt	False
Integrated Security	**True**
Password	
Persist Security Info	False
TrustServerCertificate	False
User ID	
Source	
AttachDbFilename	
Context Connection	False
Data Source	**DESKTOP-GIRHCPM\SQLE**

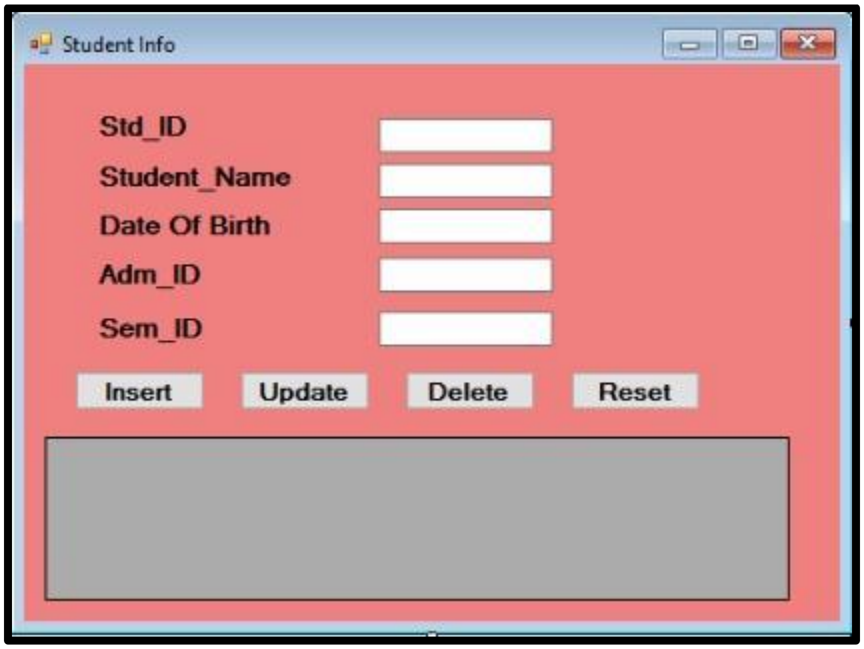
Data Source
Indicates the name of the data source to connect to.

```
nitial Catalog=student_admission;Integrated Security=True
```

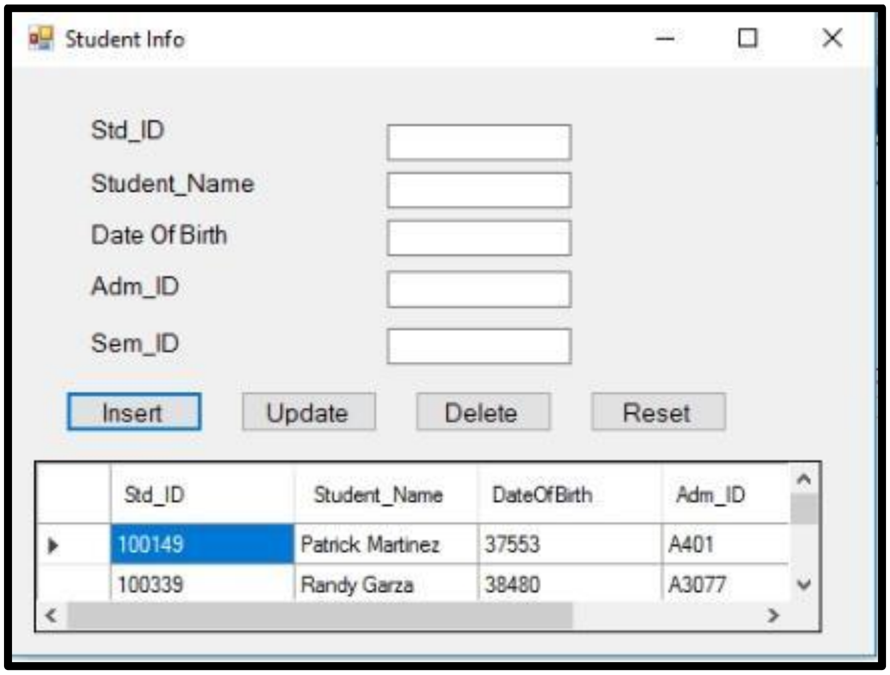


<u>IMPORTING DATA FROM SQL SERVER</u>
<u>FOR STUDENT TABLE</u>

STUDENT TABLE FORM



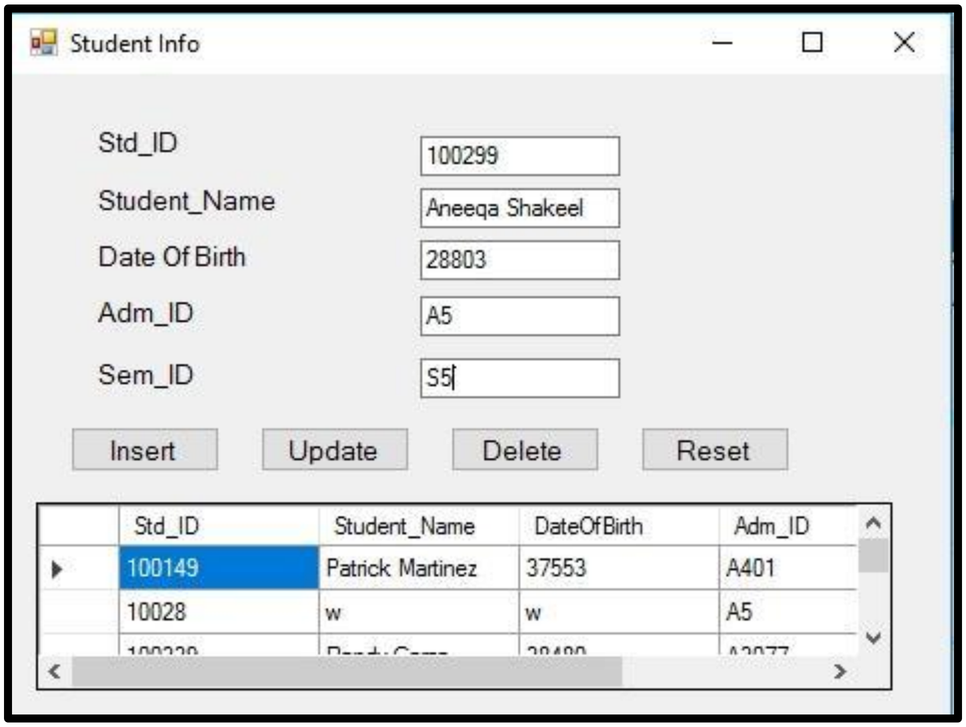
INSERT BUTTON



Student Info

Std_ID	
Student_Name	
Date Of Birth	
Adm_ID	
Sem_ID	

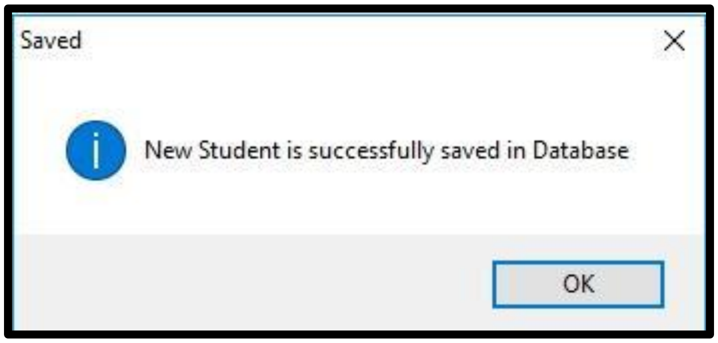
Std_ID	Student_Name	DateOfBirth	Adm_ID
100149	Patrick Martinez	37553	A401
100339	Randy Garza	38480	A3077



Student Info

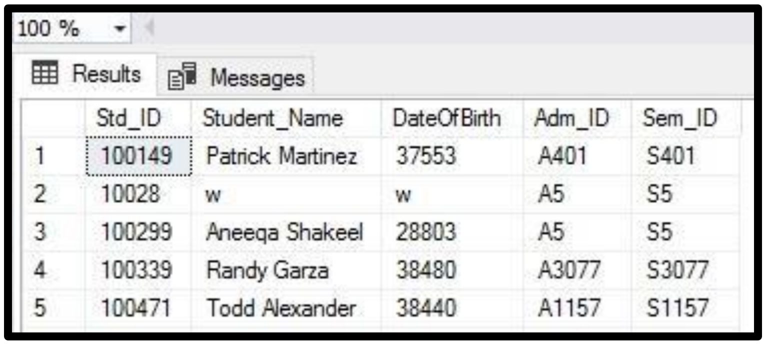
Std_ID	100299
Student_Name	Aneeqa Shakeel
Date Of Birth	28803
Adm_ID	A5
Sem_ID	S5

Std_ID	Student_Name	DateOfBirth	Adm_ID
100149	Patrick Martinez	37553	A401
10028	w	w	A5


Saved

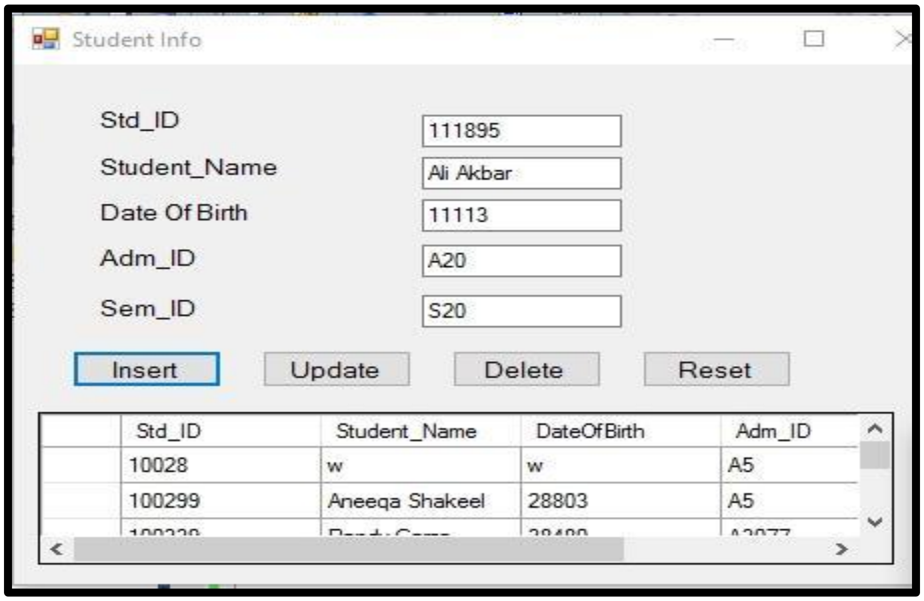
New Student is successfully saved in Database

IN DATABASE SERVER



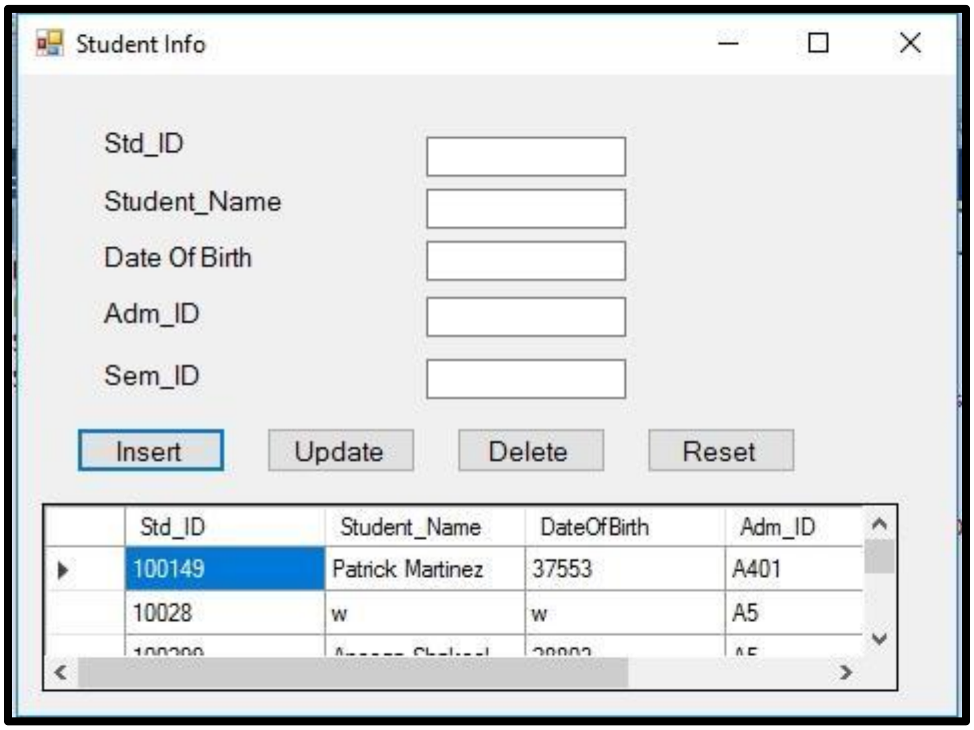
	Std_ID	Student_Name	DateOfBirth	Adm_ID	Sem_ID
1	100149	Patrick Martinez	37553	A401	S401
2	10028	w	w	A5	S5
3	100299	Aneeqa Shakeel	28803	A5	S5
4	100339	Randy Garza	38480	A3077	S3077
5	100471	Todd Alexander	38440	A1157	S1157

RESET BUTTON



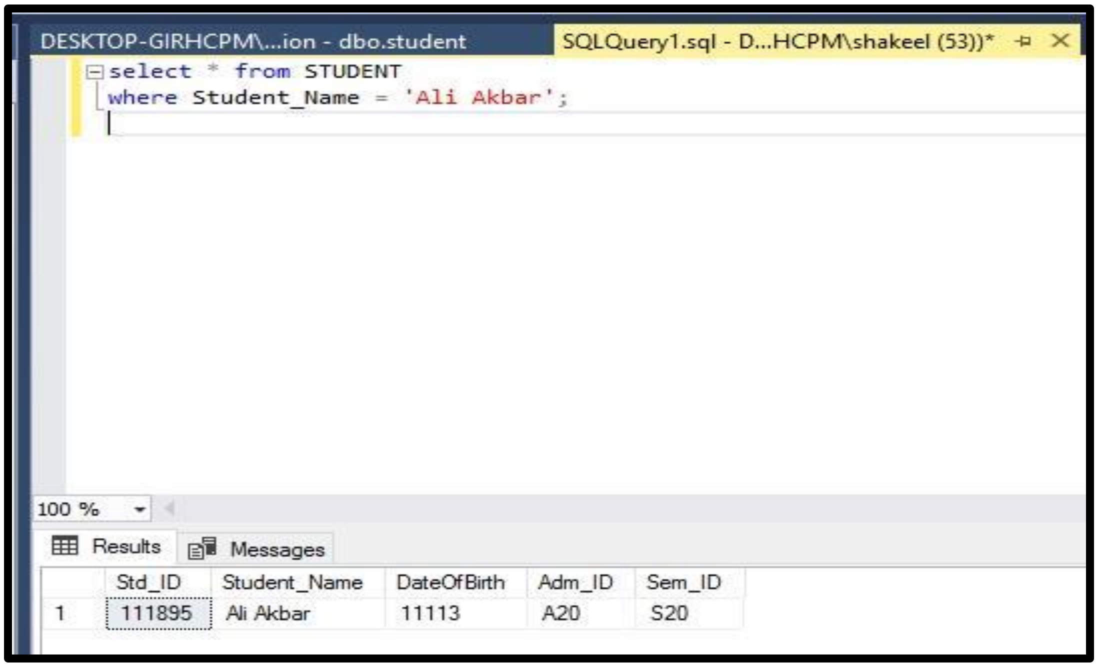
Std_ID	111895
Student_Name	Ali Akbar
Date Of Birth	11113
Adm_ID	A20
Sem_ID	S20

Std_ID	Student_Name	DateOfBirth	Adm_ID
10028	w	w	A5
100299	Aneeqa Shakeel	28803	A5



Std_ID	Student_Name	DateOfBirth	Adm_ID
100149	Patrick Martinez	37553	A401
10028	w	w	A5

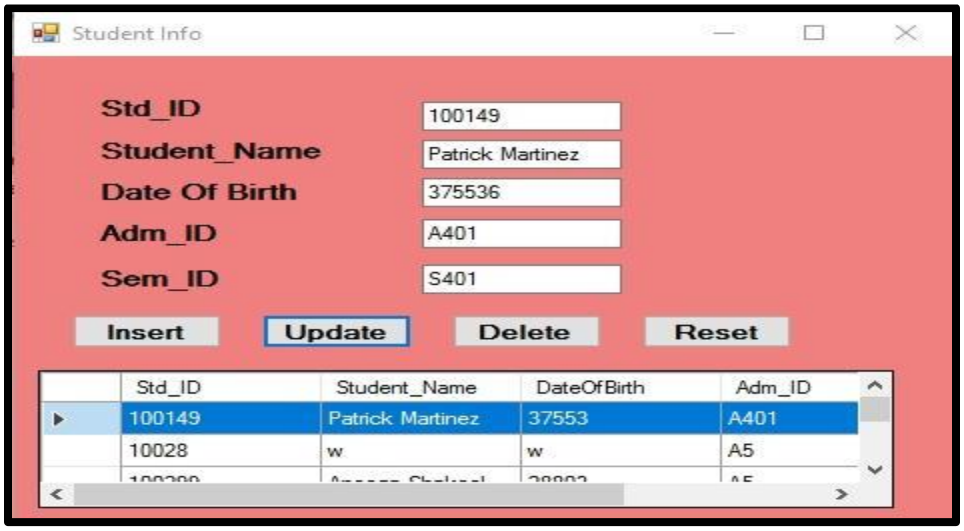
IN DATABASE SERVER


DESKTOP-GIRHCPM\...ion - dbo.student
SQLQuery1.sql - D...HCPM\shakeel (53))*
100 %
Results
Messages

```
select * from STUDENT
where Student_Name = 'Ali Akbar';
```

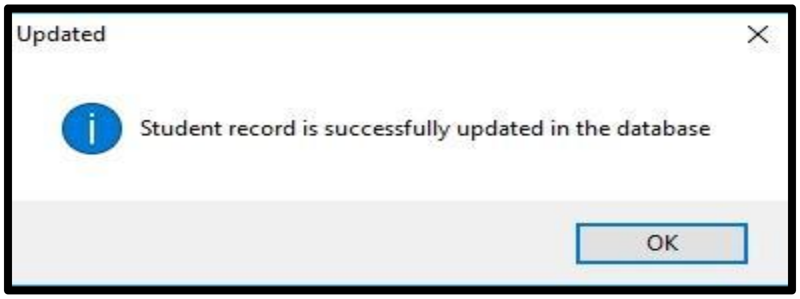
```
   Std_ID   Student_Name   DateOfBirth   Adm_ID   Sem_ID
1  111895   Ali Akbar      11113         A20      S20
```

UPDATE BUTTON



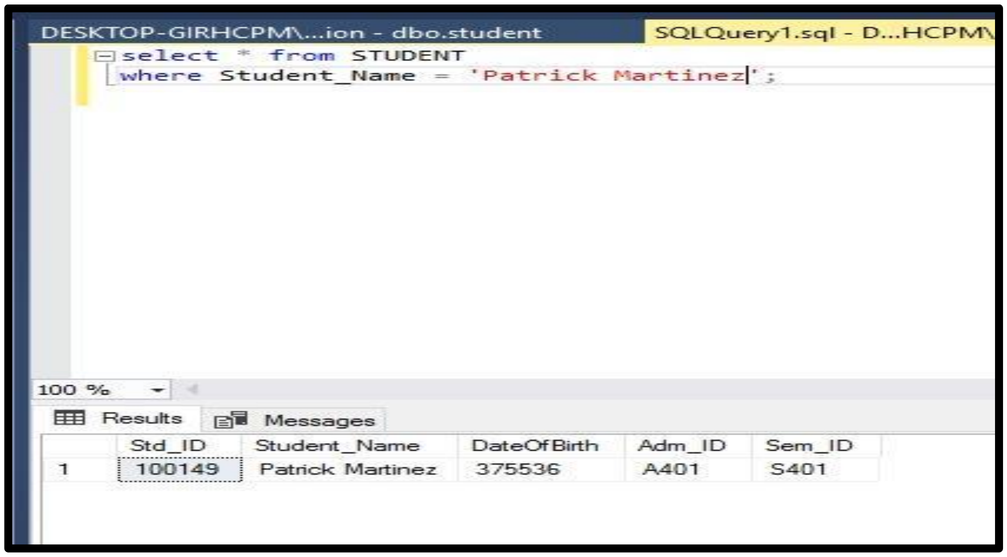
Std_ID 100149
Student_Name Patrick Martinez
Date Of Birth 375536
Adm_ID A401
Sem_ID S401

	Std_ID	Student_Name	DateOfBirth	Adm_ID
▸	100149	Patrick Martinez	37553	A401
	10028	w	w	A5



Student record is successfully updated in the database

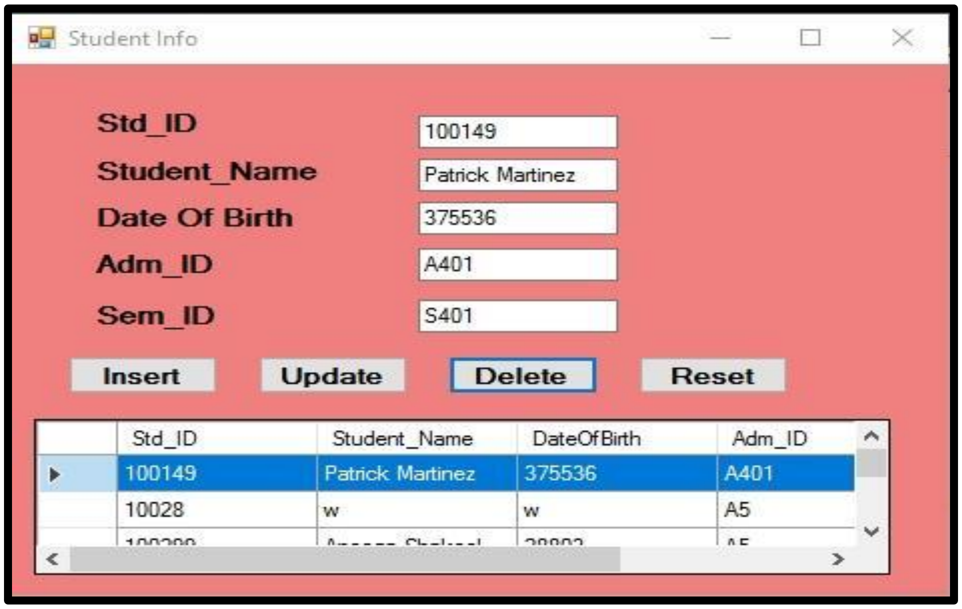
IN DATABASE SERVER


DESKTOP-GIRHCPM\...ion - dbo.student
SQLQuery1.sql - D...HCPM\
100 %
Results
Messages

```
select * from STUDENT
where Student_Name = 'Patrick Martinez';
```

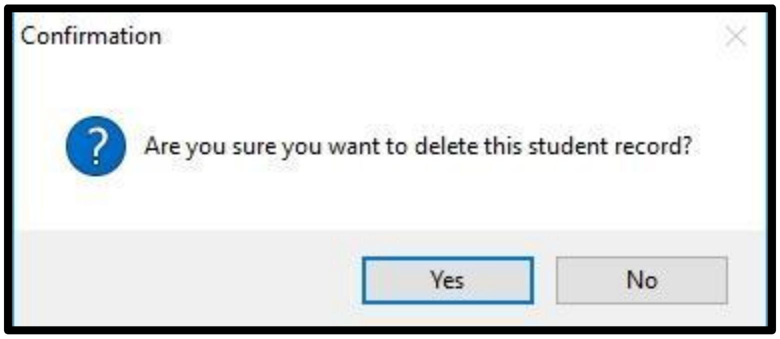
	Std_ID	Student_Name	DateOfBirth	Adm_ID	Sem_ID
1	100149	Patrick Martinez	375536	A401	S401

DELETE BUTTON

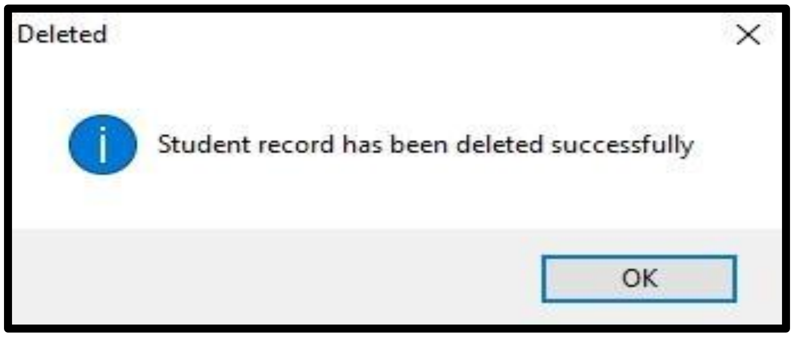

Student Info

Std_ID	100149
Student_Name	Patrick Martinez
Date Of Birth	375536
Adm_ID	A401
Sem_ID	S401

Std_ID	Student_Name	DateOfBirth	Adm_ID
100149	Patrick Martinez	375536	A401
10028	w	w	A5

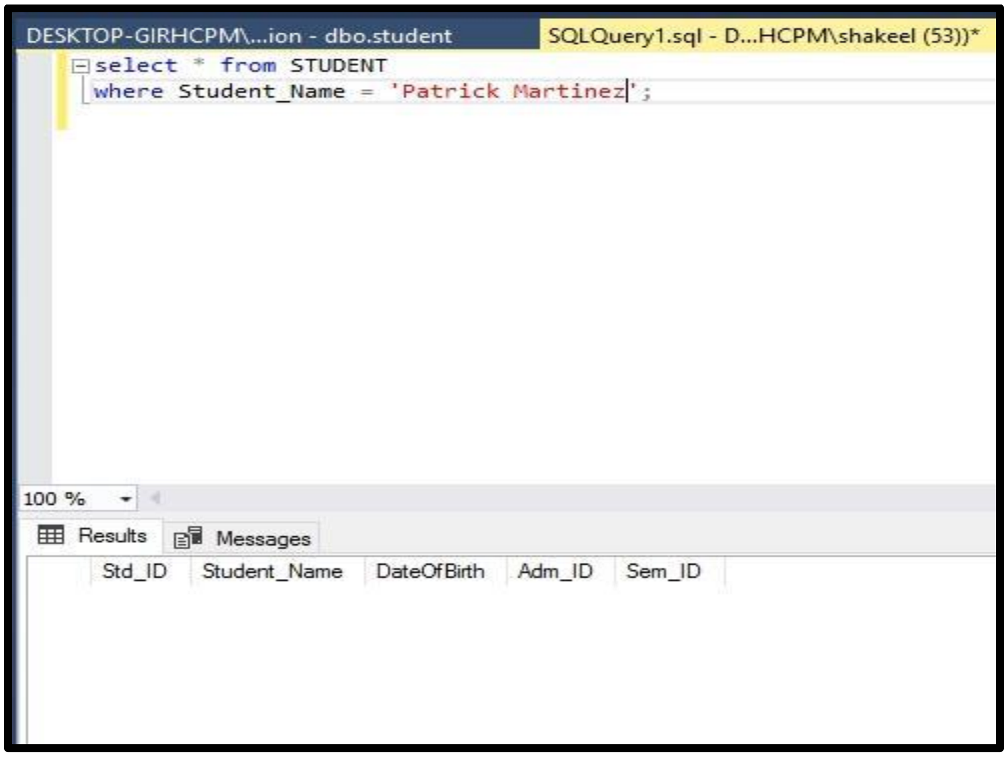

Confirmation

Are you sure you want to delete this student record?


Deleted

Student record has been deleted successfully

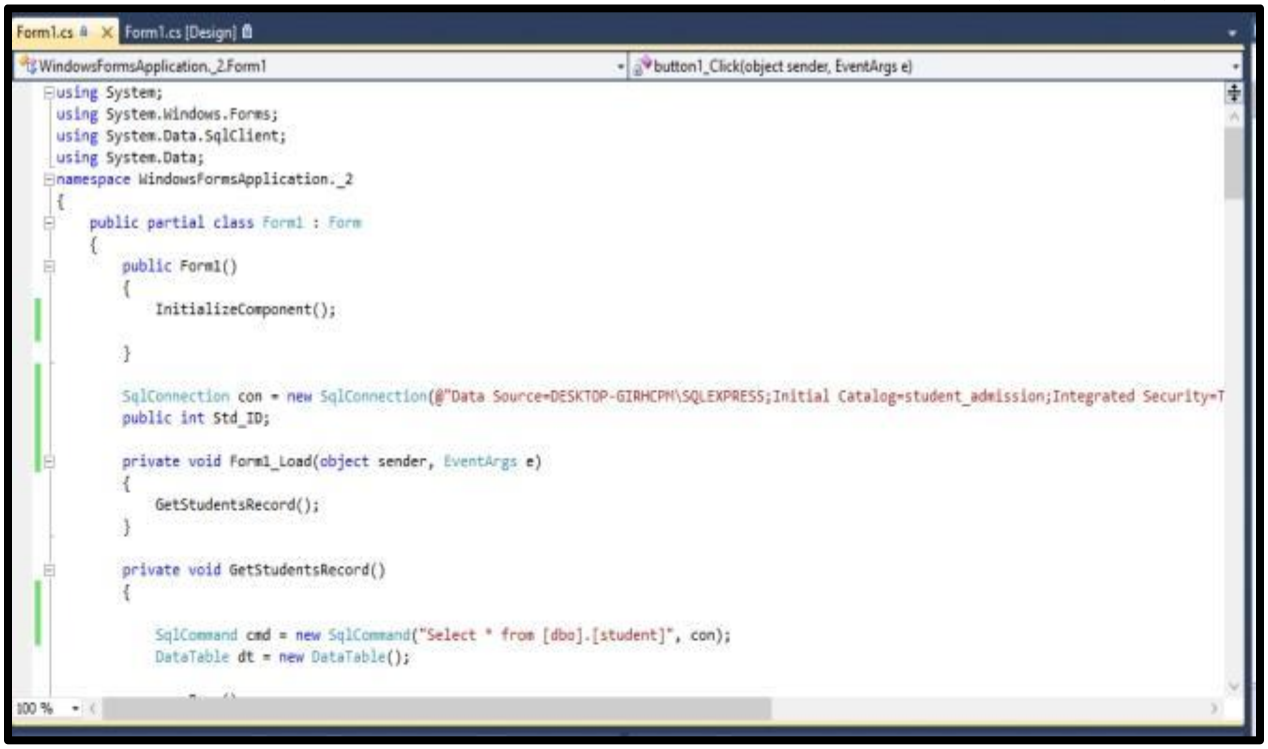
IN DATABASE SERVER


DESKTOP-GIRHCPM\...ion - dbo.student
SQLQuery1.sql - D...HCPM\shakeel (53))*
100 %
Results
Messages

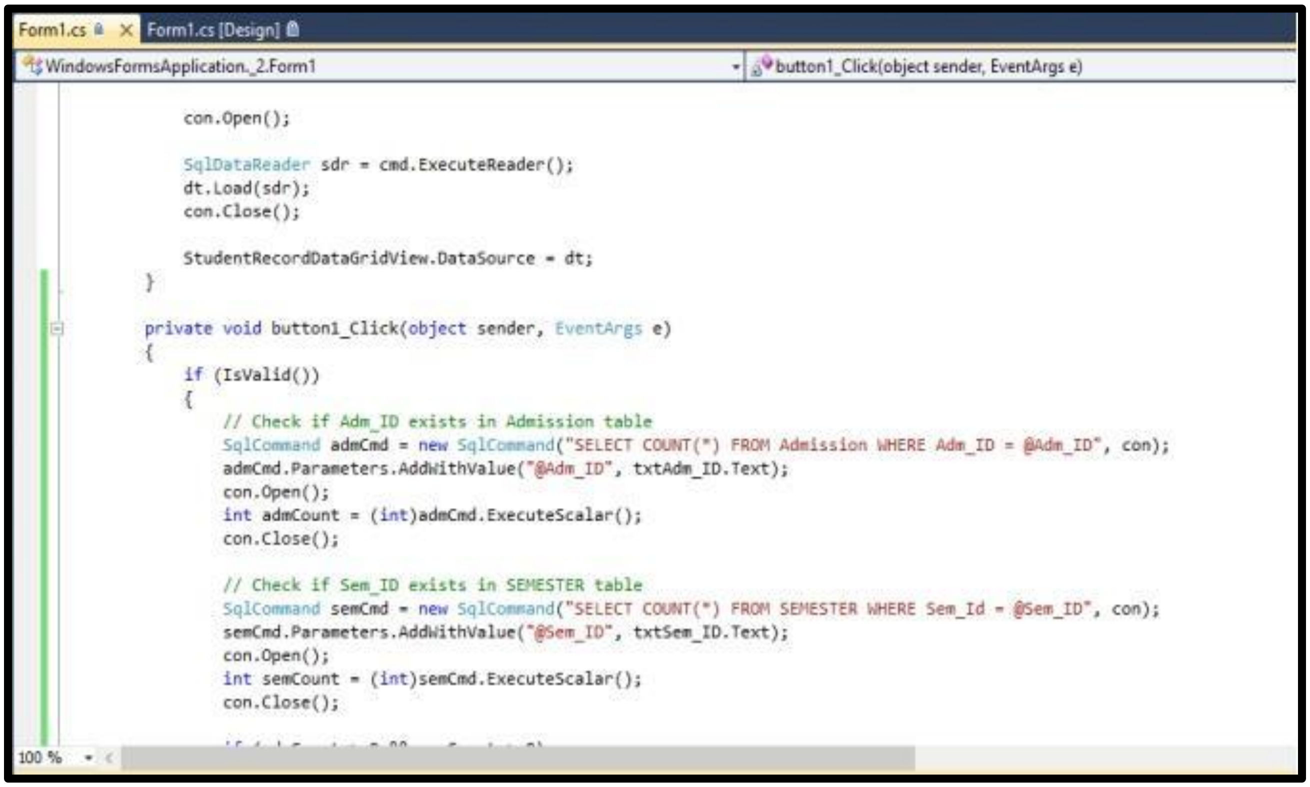
```
select * from STUDENT
where Student_Name = 'Patrick Martinez';
```

Std_ID	Student_Name	DateOfBirth	Adm_ID	Sem_ID

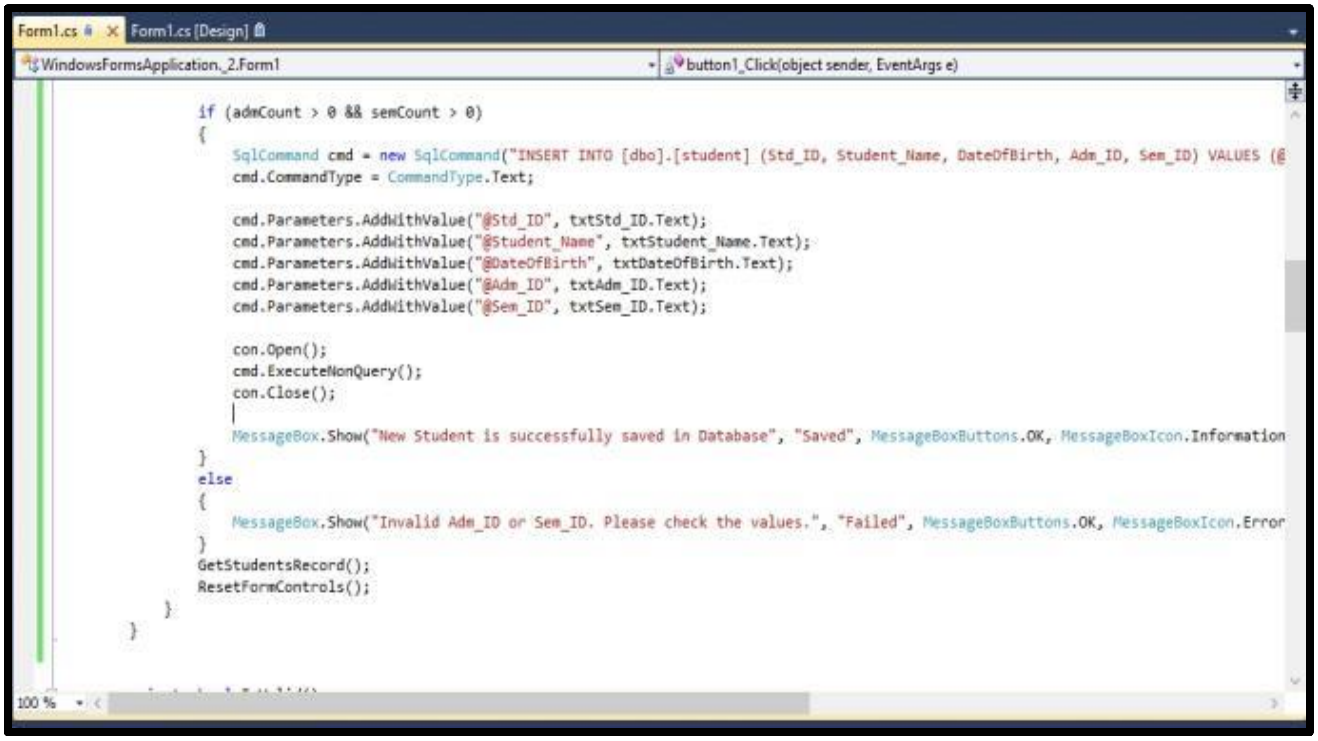
CODE AND QUERIES FOR STUDENT TABLE (FORM 1)


Form1.cs
Form1.cs [Design]
WindowsFormsApplication._2.Form1
button1_Click(object sender, EventArgs e)
100 %

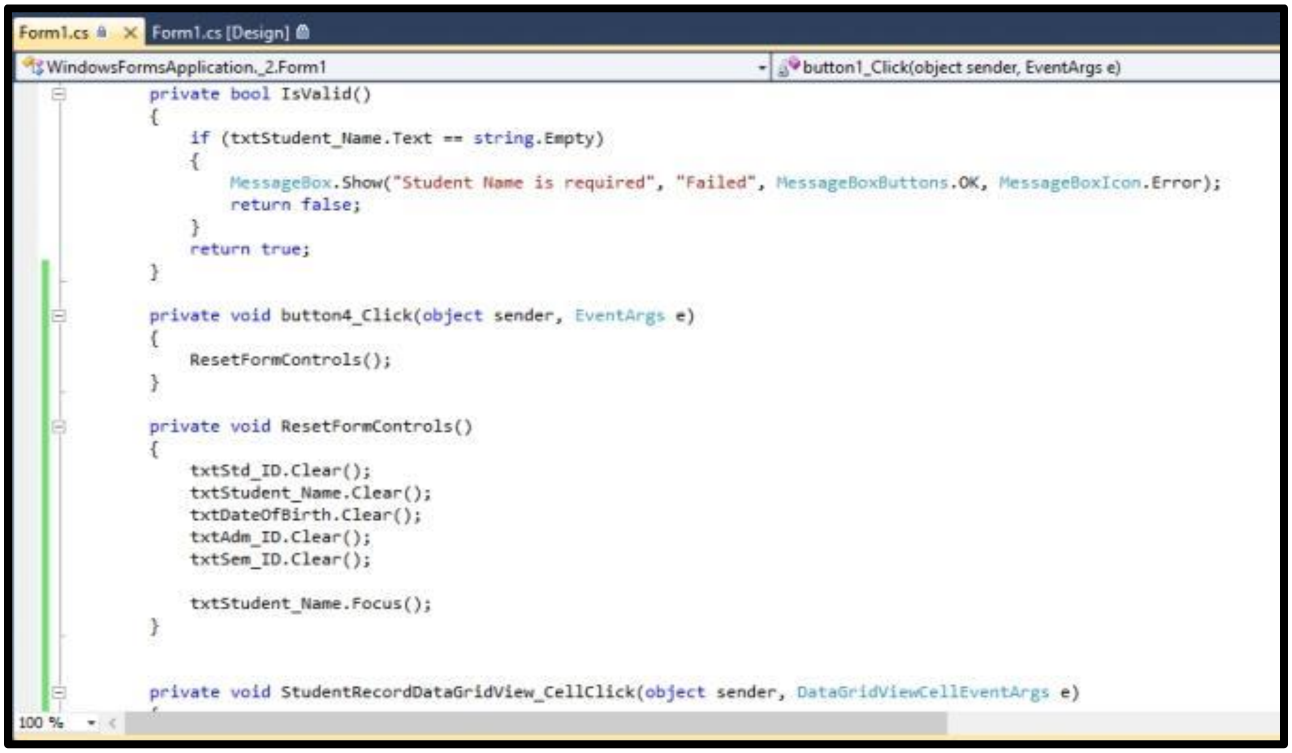
```
using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Data;
namespace WindowsFormsApplication._2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-GIRHCPM\SQLEXPRESS;Initial Catalog=student_admission;Integrated Security=T
        public int Std_ID;

        private void Form1_Load(object sender, EventArgs e)
        {
            GetStudentsRecord();
        }

        private void GetStudentsRecord()
        {

            SqlCommand cmd = new SqlCommand("Select * from [dbo].[student]", con);
            DataTable dt = new DataTable();
```


Form1.cs
Form1.cs [Design]
WindowsFormsApplication._2.Form1
button1_Click(object sender, EventArgs e)
100 %

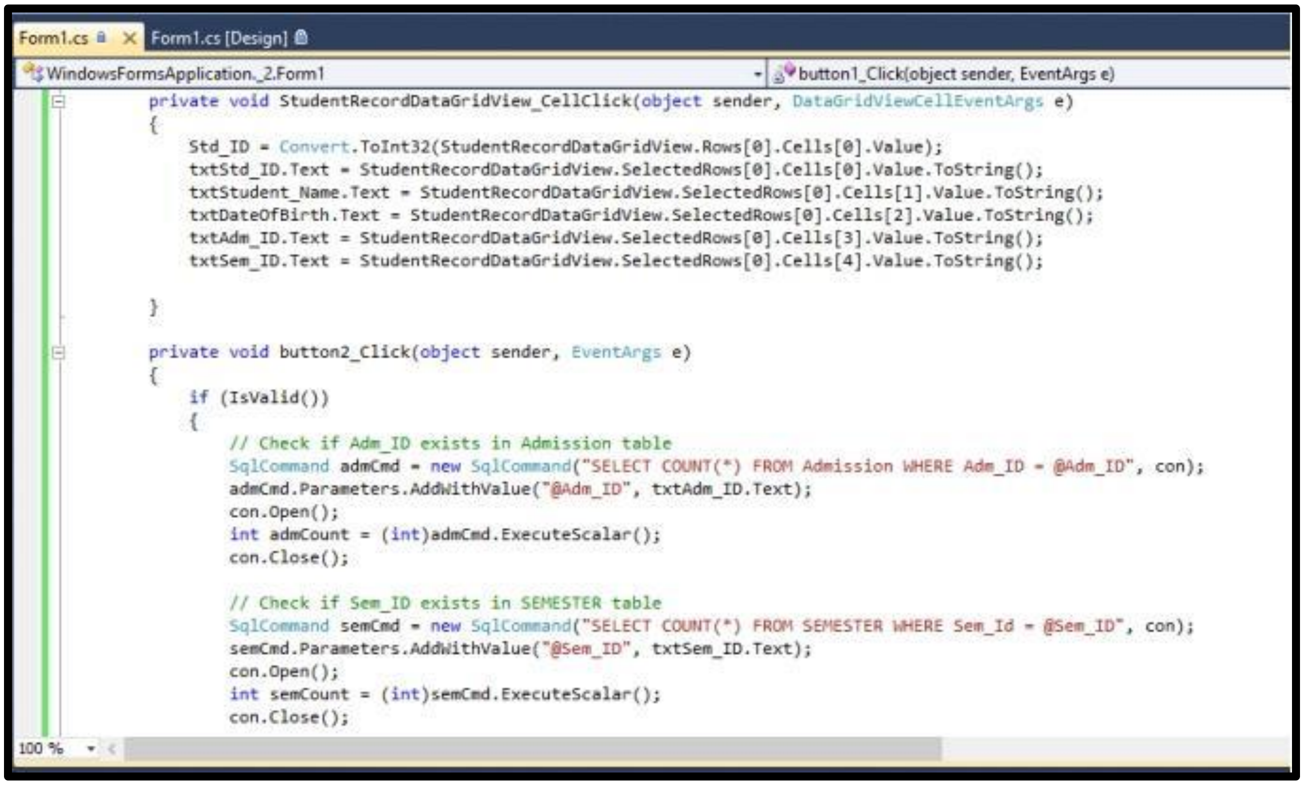
```
            con.Open();

            SqlDataReader sdr = cmd.ExecuteReader();
            dt.Load(sdr);
            con.Close();

            StudentRecordDataGridView.DataSource = dt;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (IsValid())
            {
                // Check if Adm_ID exists in Admission table
                SqlCommand admCmd = new SqlCommand("SELECT COUNT(*) FROM Admission WHERE Adm_ID = @Adm_ID", con);
                admCmd.Parameters.AddWithValue("@Adm_ID", txtAdm_ID.Text);
                con.Open();
                int admCount = (int)admCmd.ExecuteScalar();
                con.Close();

                // Check if Sem_ID exists in SEMESTER table
                SqlCommand semCmd = new SqlCommand("SELECT COUNT(*) FROM SEMESTER WHERE Sem_Id = @Sem_ID", con);
                semCmd.Parameters.AddWithValue("@Sem_ID", txtSem_ID.Text);
                con.Open();
                int semCount = (int)semCmd.ExecuteScalar();
                con.Close();
```


Form1.cs
Form1.cs [Design]
WindowsFormsApplication._2.Form1
button1_Click(object sender, EventArgs e)
100 %

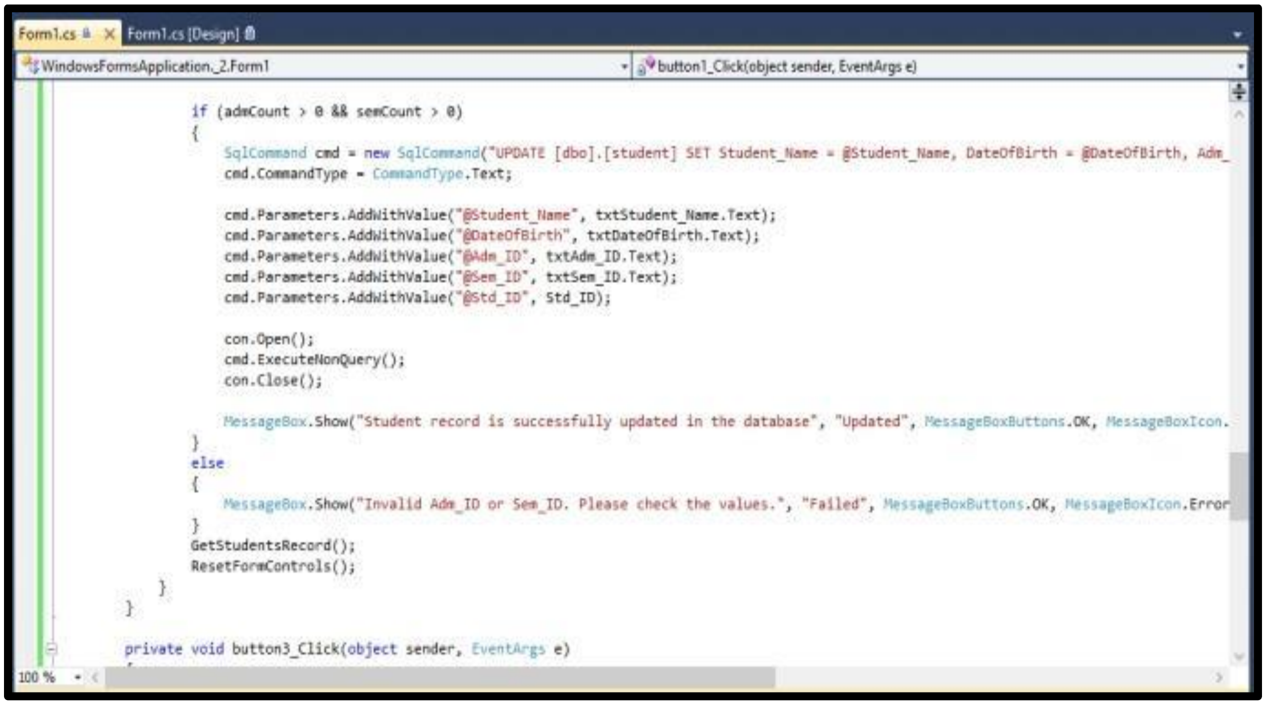
```
                if (admCount > 0 && semCount > 0)
                {
                    SqlCommand cmd = new SqlCommand("INSERT INTO [dbo].[student] (Std_ID, Student_Name, DateOfBirth, Adm_ID, Sem_ID) VALUES (@
                    cmd.CommandType = CommandType.Text;

                    cmd.Parameters.AddWithValue("@Std_ID", txtStd_ID.Text);
                    cmd.Parameters.AddWithValue("@Student_Name", txtStudent_Name.Text);
                    cmd.Parameters.AddWithValue("@DateOfBirth", txtDateOfBirth.Text);
                    cmd.Parameters.AddWithValue("@Adm_ID", txtAdm_ID.Text);
                    cmd.Parameters.AddWithValue("@Sem_ID", txtSem_ID.Text);

                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();

                    MessageBox.Show("New Student is successfully saved in Database", "Saved", MessageBoxButtons.OK, MessageBoxIcon.Information
                }
                else
                {
                    MessageBox.Show("Invalid Adm_ID or Sem_ID. Please check the values.", "Failed", MessageBoxButtons.OK, MessageBoxIcon.Error
                }
                GetStudentsRecord();
                ResetFormControls();
            }
        }
```


Form1.cs
Form1.cs [Design]
WindowsFormsApplication._2.Form1
button1_Click(object sender, EventArgs e)
100 %

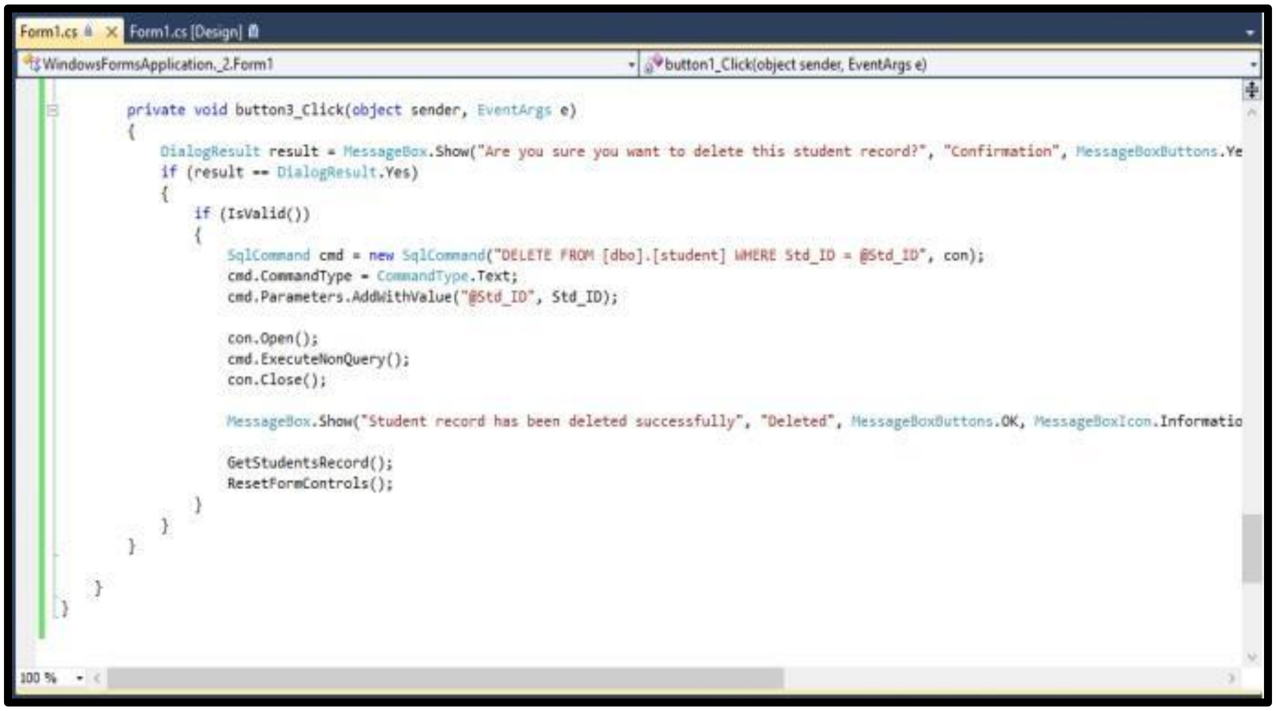
```
        private bool IsValid()
        {
            if (txtStudent_Name.Text == string.Empty)
            {
                MessageBox.Show("Student Name is required", "Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
            return true;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            ResetFormControls();
        }

        private void ResetFormControls()
        {
            txtStd_ID.Clear();
            txtStudent_Name.Clear();
            txtDateOfBirth.Clear();
            txtAdm_ID.Clear();
            txtSem_ID.Clear();

            txtStudent_Name.Focus();
        }


        private void StudentRecordDataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
```


Form1.cs
Form1.cs [Design]
WindowsFormsApplication._2.Form1
button1_Click(object sender, EventArgs e)
100 %

```
        private void StudentRecordDataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            Std_ID = Convert.ToInt32(StudentRecordDataGridView.Rows[0].Cells[0].Value);
            txtStd_ID.Text = StudentRecordDataGridView.SelectedRows[0].Cells[0].Value.ToString();
            txtStudent_Name.Text = StudentRecordDataGridView.SelectedRows[0].Cells[1].Value.ToString();
            txtDateOfBirth.Text = StudentRecordDataGridView.SelectedRows[0].Cells[2].Value.ToString();
            txtAdm_ID.Text = StudentRecordDataGridView.SelectedRows[0].Cells[3].Value.ToString();
            txtSem_ID.Text = StudentRecordDataGridView.SelectedRows[0].Cells[4].Value.ToString();

        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (IsValid())
            {
                // Check if Adm_ID exists in Admission table
                SqlCommand admCmd = new SqlCommand("SELECT COUNT(*) FROM Admission WHERE Adm_ID = @Adm_ID", con);
                admCmd.Parameters.AddWithValue("@Adm_ID", txtAdm_ID.Text);
                con.Open();
                int admCount = (int)admCmd.ExecuteScalar();
                con.Close();

                // Check if Sem_ID exists in SEMESTER table
                SqlCommand semCmd = new SqlCommand("SELECT COUNT(*) FROM SEMESTER WHERE Sem_Id = @Sem_ID", con);
                semCmd.Parameters.AddWithValue("@Sem_ID", txtSem_ID.Text);
                con.Open();
                int semCount = (int)semCmd.ExecuteScalar();
                con.Close();
```

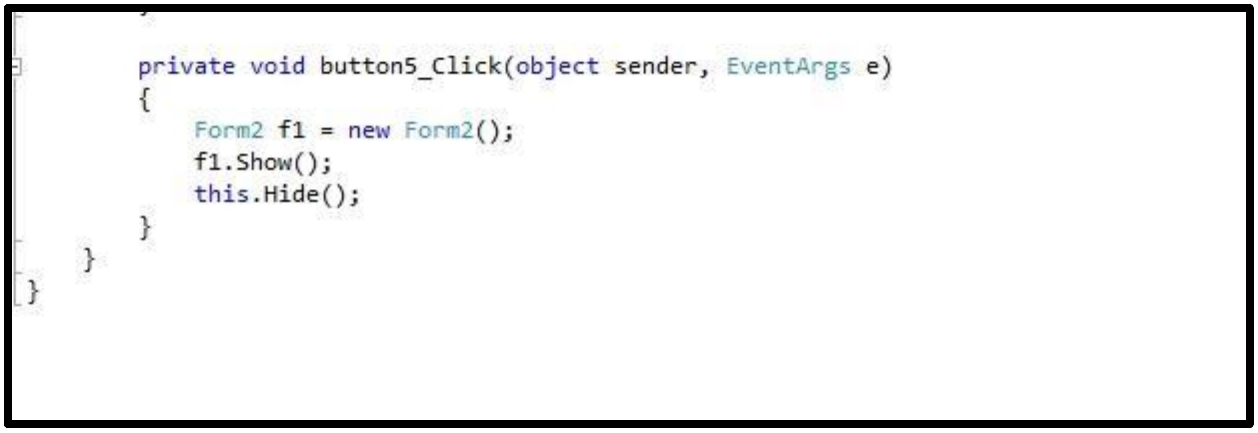

Form1.cs
Form1.cs [Design]
WindowsFormsApplication._2.Form1
button1_Click(object sender, EventArgs e)
100 %

```
                if (admCount > 0 && semCount > 0)
                {
                    SqlCommand cmd = new SqlCommand("UPDATE [dbo].[student] SET Student_Name = @Student_Name, DateOfBirth = @DateOfBirth, Adm_
                    cmd.CommandType = CommandType.Text;

                    cmd.Parameters.AddWithValue("@Student_Name", txtStudent_Name.Text);
                    cmd.Parameters.AddWithValue("@DateOfBirth", txtDateOfBirth.Text);
                    cmd.Parameters.AddWithValue("@Adm_ID", txtAdm_ID.Text);
                    cmd.Parameters.AddWithValue("@Sem_ID", txtSem_ID.Text);
                    cmd.Parameters.AddWithValue("@Std_ID", Std_ID);

                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();

                    MessageBox.Show("Student record is successfully updated in the database", "Updated", MessageBoxButtons.OK, MessageBoxIcon.
                }
                else
                {
                    MessageBox.Show("Invalid Adm_ID or Sem_ID. Please check the values.", "Failed", MessageBoxButtons.OK, MessageBoxIcon.Error
                }
                GetStudentsRecord();
                ResetFormControls();
            }
        }

        private void button3_Click(object sender, EventArgs e)
```



Form1.cs
Form1.cs [Design]
WindowsFormsApplication._2.Form1
button1_Click(object sender, EventArgs e)
100 %

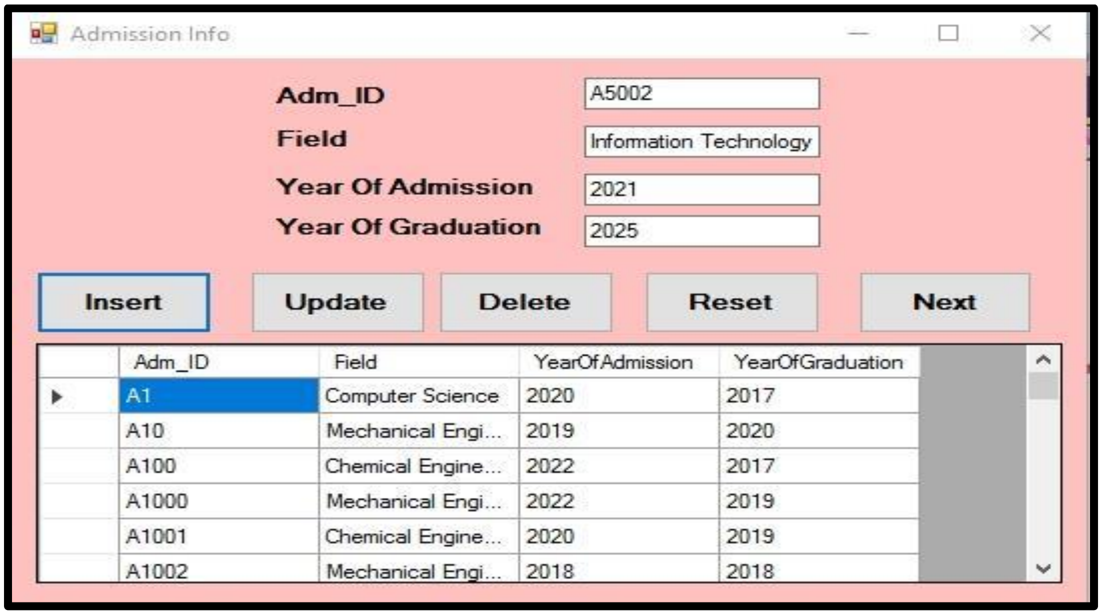
```
        private void button3_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to delete this student record?", "Confirmation", MessageBoxButtons.Ye
            if (result == DialogResult.Yes)
            {
                if (IsValid())
                {
                    SqlCommand cmd = new SqlCommand("DELETE FROM [dbo].[student] WHERE Std_ID = @Std_ID", con);
                    cmd.CommandType = CommandType.Text;
                    cmd.Parameters.AddWithValue("@Std_ID", Std_ID);

                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();

                    MessageBox.Show("Student record has been deleted successfully", "Deleted", MessageBoxButtons.OK, MessageBoxIcon.Informatio

                    GetStudentsRecord();
                    ResetFormControls();
                }
            }
        }

    }
}
```



```
        private void button5_Click(object sender, EventArgs e)
        {
            Form2 f1 = new Form2();
            f1.Show();
            this.Hide();
        }
    }
}
```

FOR ADMISSION TABLE

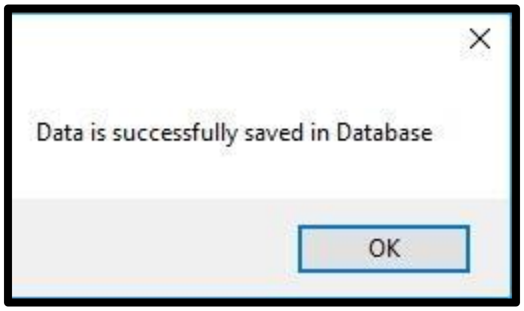
INSERT BUTTON



Admission Info

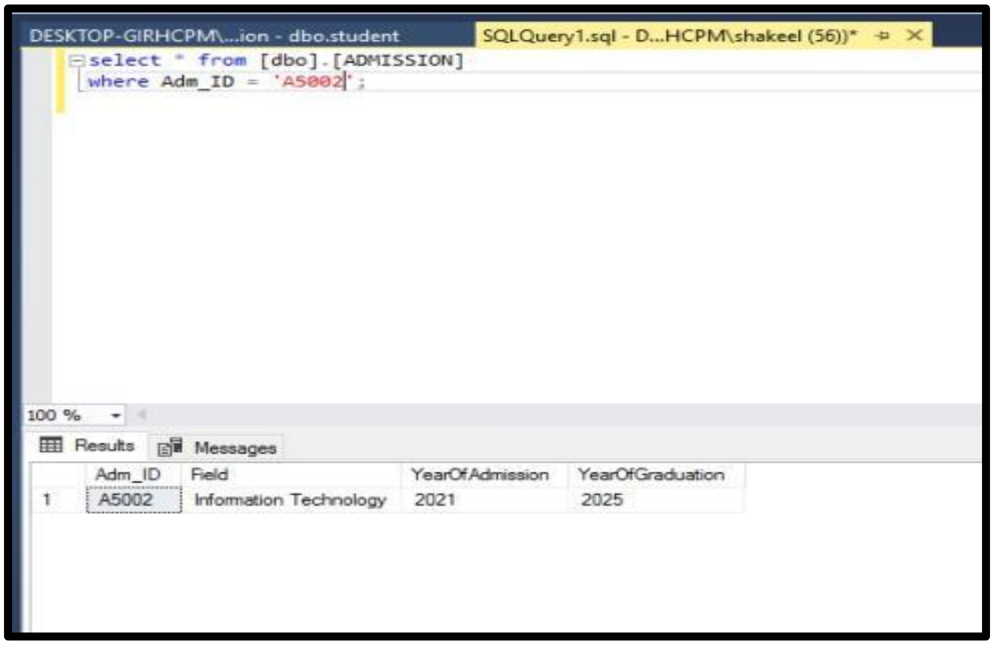
Adm_ID	A5002
Field	Information Technology
Year Of Admission	2021
Year Of Graduation	2025

Adm_ID	Field	YearOfAdmission	YearOfGraduation
A1	Computer Science	2020	2017
A10	Mechanical Engi...	2019	2020
A100	Chemical Engine...	2022	2017
A1000	Mechanical Engi...	2022	2019
A1001	Chemical Engine...	2020	2019
A1002	Mechanical Engi...	2018	2018



Data is successfully saved in Database

IN DATABASE SERVER

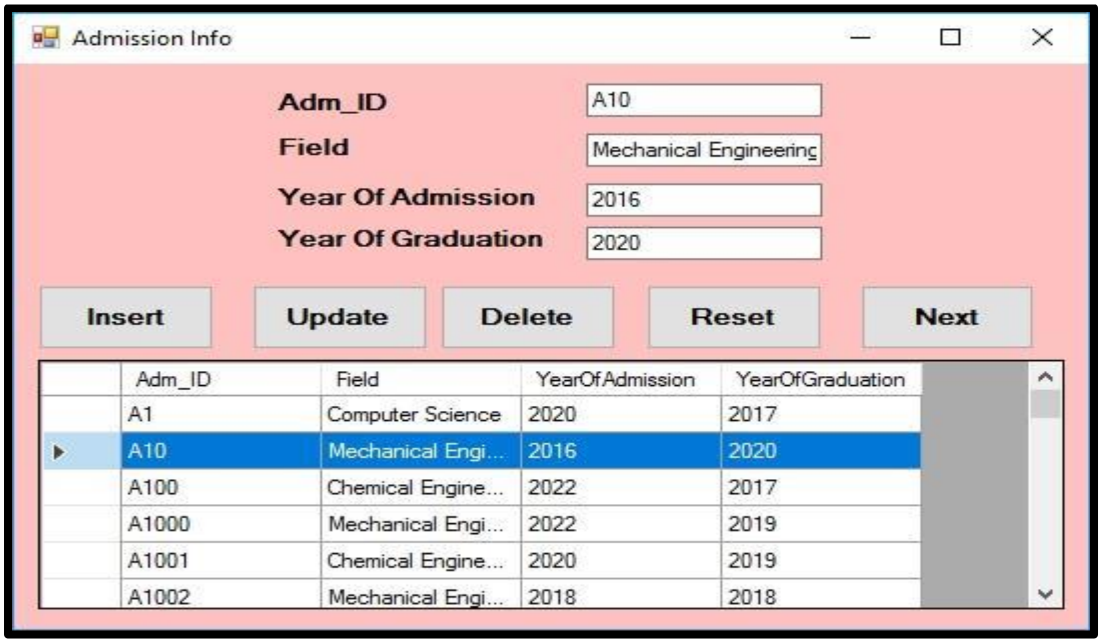

DESKTOP-GIRHCPM\...ion - dbo.student
SQLQuery1.sql - D...HCPM\shakeel (56))*
100 %
Results
Messages

```
select * from [dbo].[ADMISSION]
where Adm_ID = 'A5002';
```

	Adm_ID	Field	YearOfAdmission	YearOfGraduation
1	A5002	Information Technology	2021	2025

UPDATE BUTTON

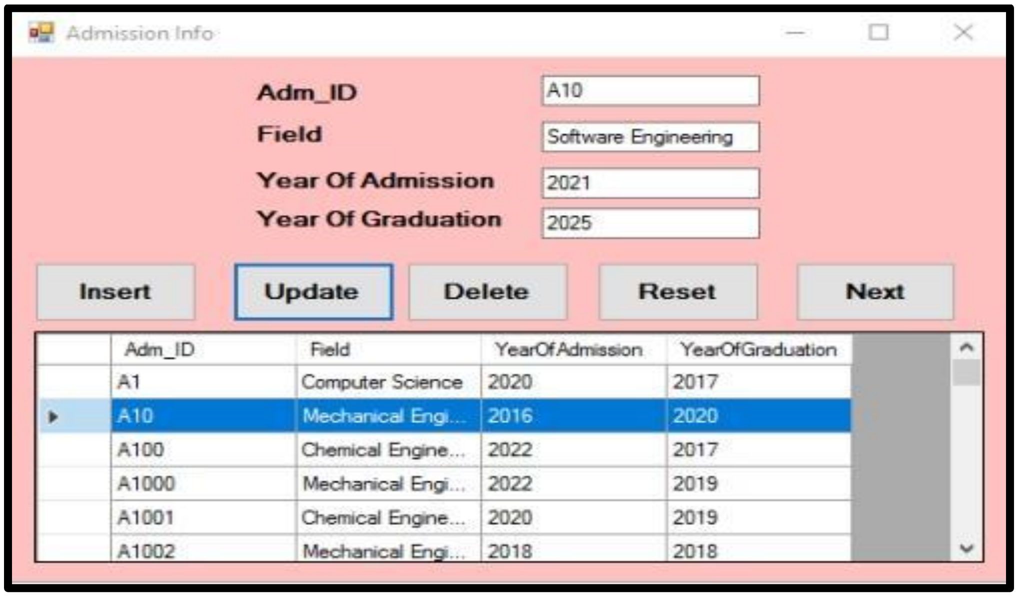
BEFORE UPDATE



Adm_ID	A10
Field	Mechanical Engineering
Year Of Admission	2016
Year Of Graduation	2020

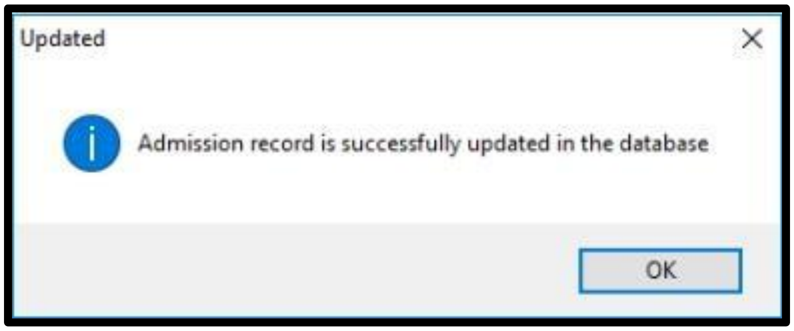
	Adm_ID	Field	YearOfAdmission	YearOfGraduation
	A1	Computer Science	2020	2017
▶	A10	Mechanical Engi...	2016	2020
	A100	Chemical Engine...	2022	2017
	A1000	Mechanical Engi...	2022	2019
	A1001	Chemical Engine...	2020	2019
	A1002	Mechanical Engi...	2018	2018

AFTER UPDATE



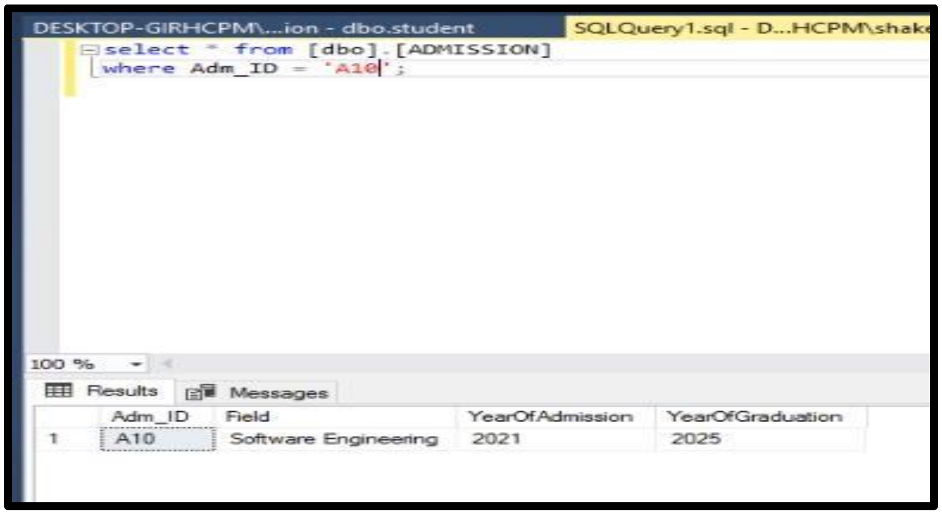
Adm_ID	A10
Field	Software Engineering
Year Of Admission	2021
Year Of Graduation	2025

Adm_ID	Field	YearOfAdmission	YearOfGraduation
A1	Computer Science	2020	2017
A10	Mechanical Engi...	2016	2020
A100	Chemical Engine...	2022	2017
A1000	Mechanical Engi...	2022	2019
A1001	Chemical Engine...	2020	2019
A1002	Mechanical Engi...	2018	2018



Admission record is successfully updated in the database

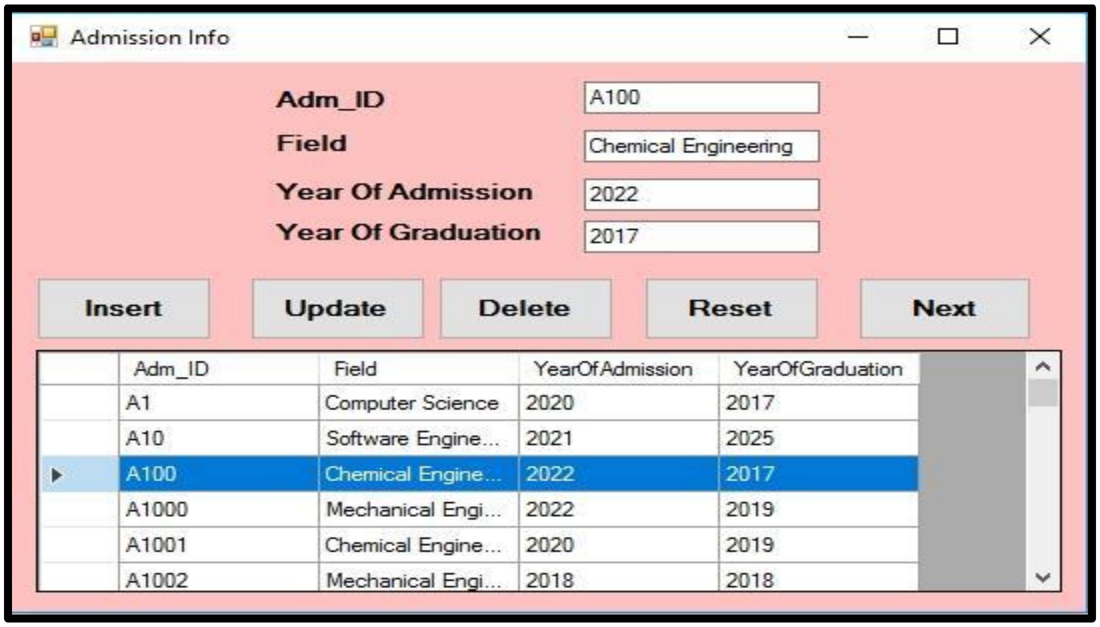
IN DATABASE SERVER


DESKTOP-GIRHCPM\...ion - dbo.student
SQLQuery1.sql - D...HCPM\shak
100 %
Results
Messages

```
select * from [dbo].[ADMISSION]
where Adm_ID = 'A10';
```

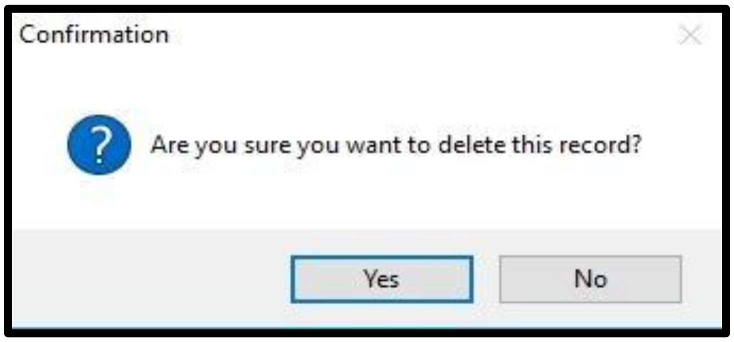
	Adm_ID	Field	YearOfAdmission	YearOfGraduation
1	A10	Software Engineering	2021	2025

DELETE BUTTON



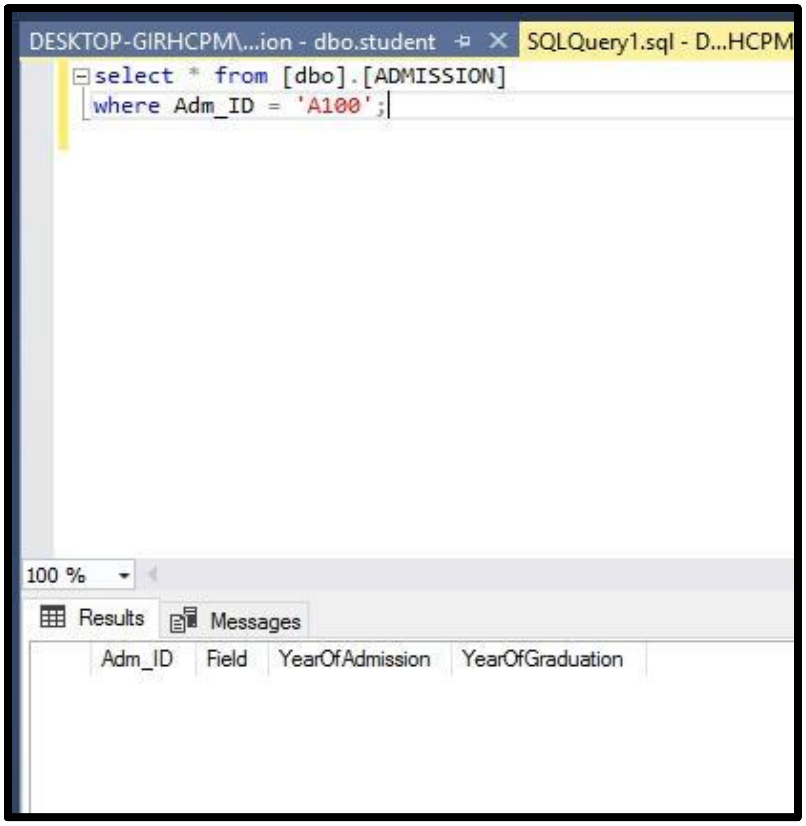
Adm_ID	A100
Field	Chemical Engineering
Year Of Admission	2022
Year Of Graduation	2017

Adm_ID	Field	YearOfAdmission	YearOfGraduation
A1	Computer Science	2020	2017
A10	Software Engine...	2021	2025
A100	Chemical Engine...	2022	2017
A1000	Mechanical Engi...	2022	2019
A1001	Chemical Engine...	2020	2019
A1002	Mechanical Engi...	2018	2018



Are you sure you want to delete this record?

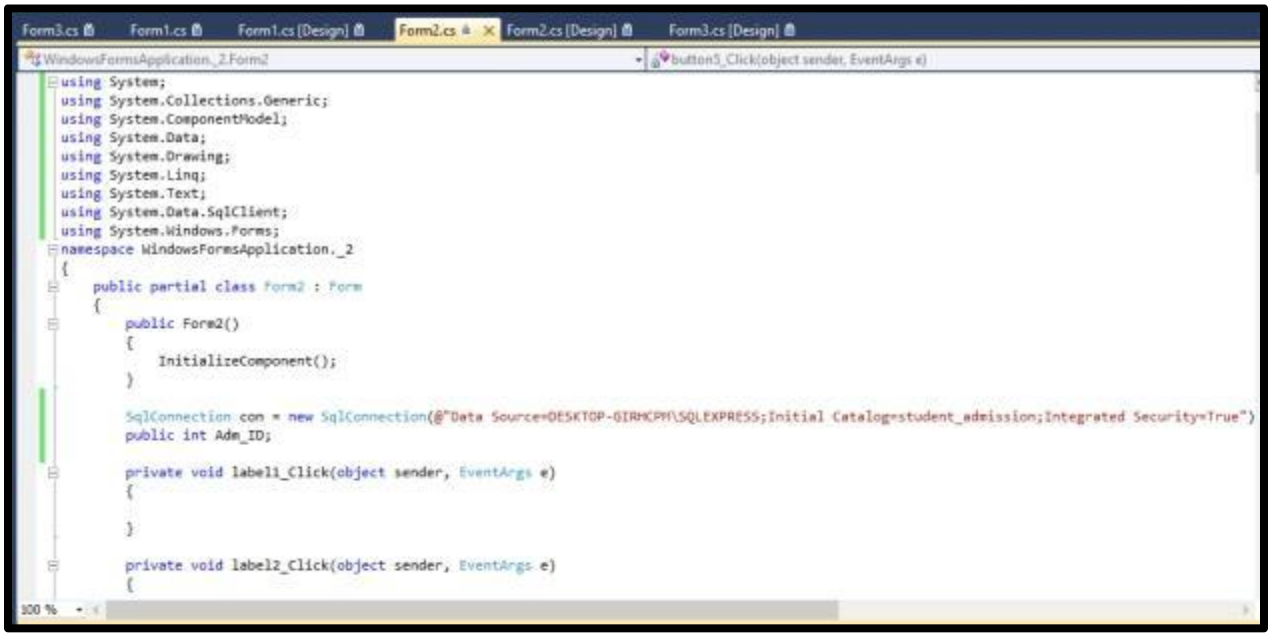
IN DATABASE SERVER



```
select * from [dbo].[ADMISSION]
where Adm_ID = 'A100';
```

```
Adm_ID  Field  YearOfAdmission  YearOfGraduation
```

CODE AND QUERIES FOR ADMISSION TABLE (FORM 2)

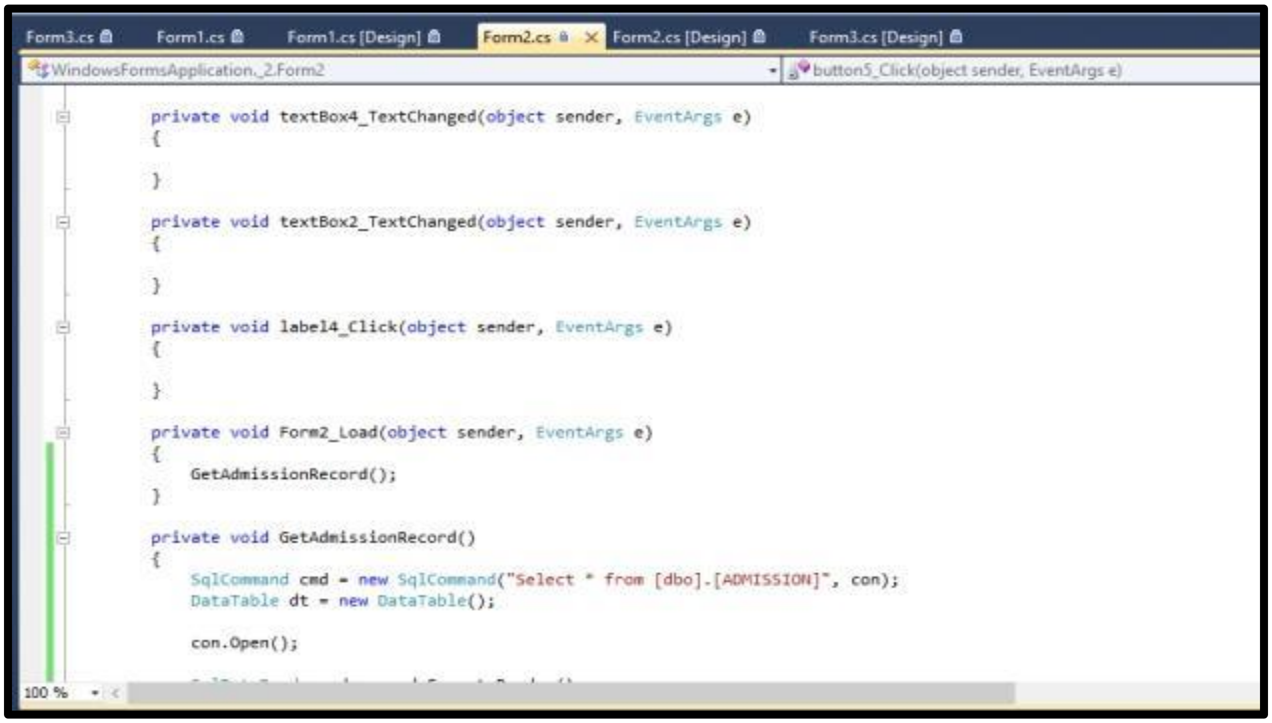


Form3.cs Form1.cs Form1.cs [Design] Form2.cs Form2.cs [Design] Form3.cs [Design]

WindowsFormsApplication._2.Form2 button5_Click(object sender, EventArgs e)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Windows.Forms;
namespace WindowsFormsApplication._2
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-GIRHCPM\SQLEXPRESS;Initial Catalog=student_admission;Integrated Security=True")
        public int Adm_ID;

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {
```

100 %

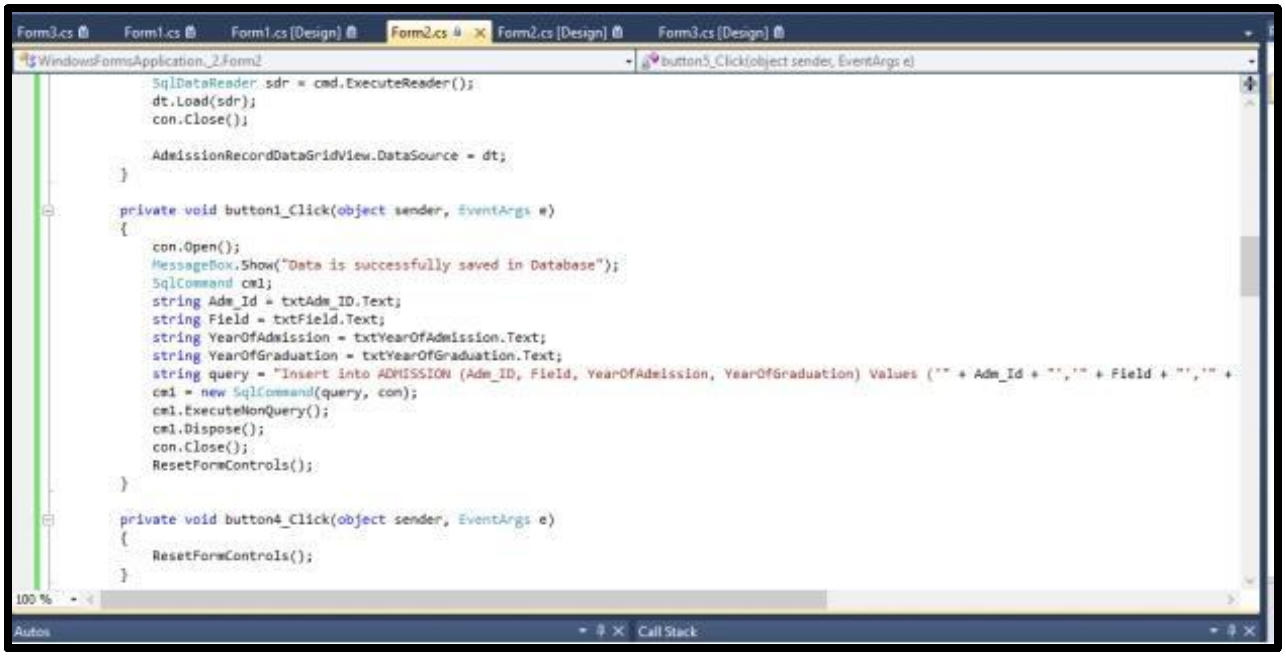


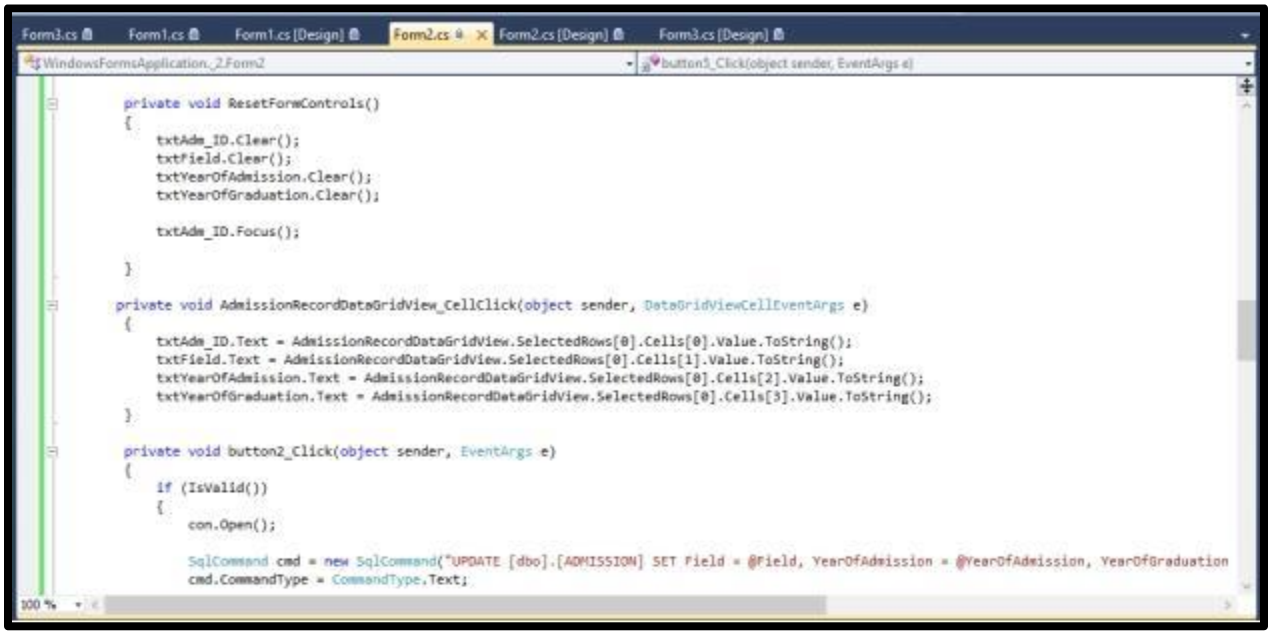
Form3.cs Form1.cs Form1.cs [Design] Form2.cs Form2.cs [Design] Form3.cs [Design]

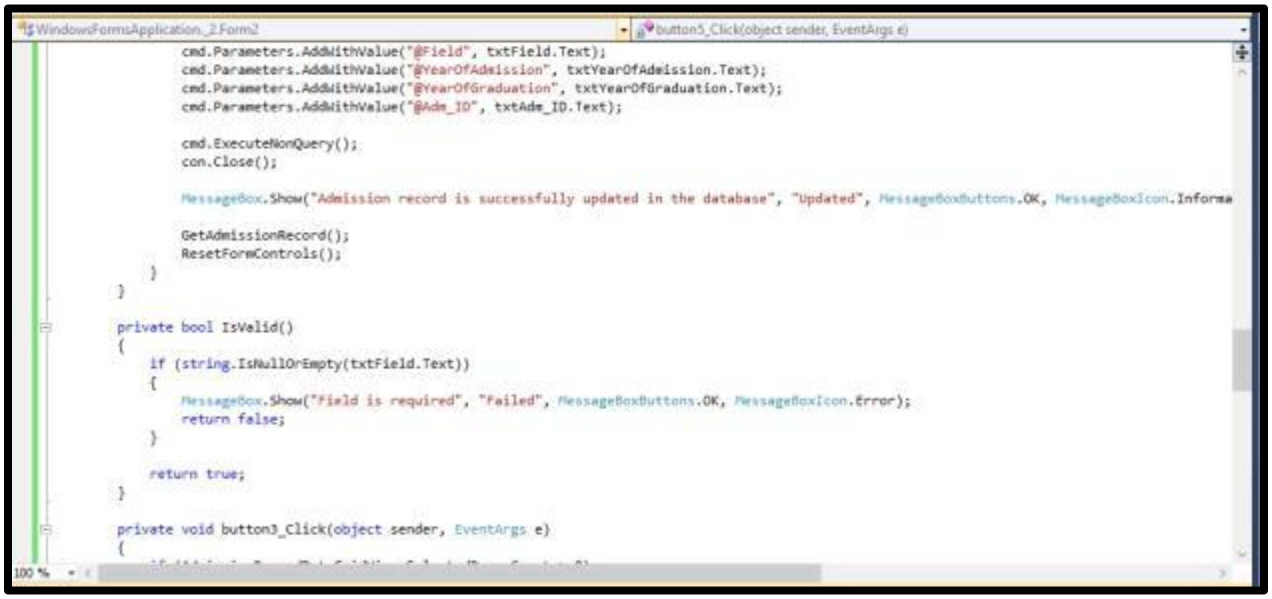
WindowsFormsApplication._2.Form2 button5_Click(object sender, EventArgs e)

```
        private void textBox4_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void Form2_Load(object sender, EventArgs e)
        {
            GetAdmissionRecord();
        }

        private void GetAdmissionRecord()
        {
            SqlCommand cmd = new SqlCommand("Select * from [dbo].[ADMISSION]", con);
            DataTable dt = new DataTable();

            con.Open();
```

100 %

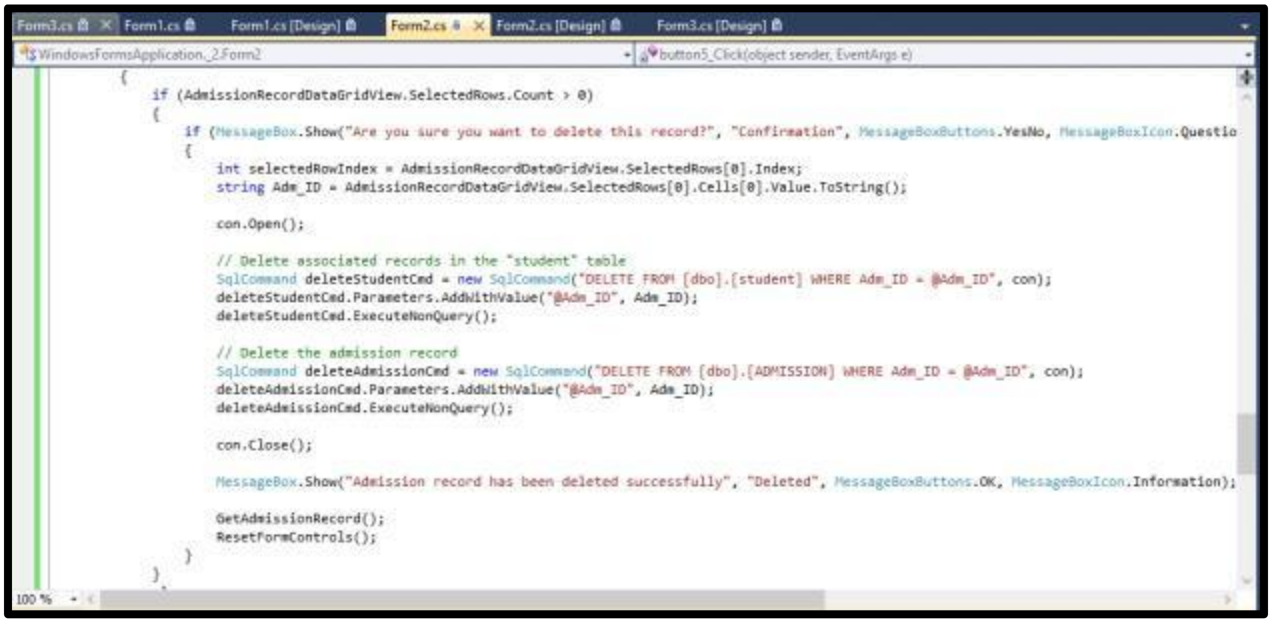

```
            SqlDataReader sdr = cmd.ExecuteReader();
            dt.Load(sdr);
            con.Close();

            AdmissionRecordDataGridView.DataSource = dt;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            con.Open();
            MessageBox.Show("Data is successfully saved in Database");
            SqlCommand cm1;
            string Adm_Id = txtAdm_ID.Text;
            string Field = txtField.Text;
            string YearOfAdmission = txtYearOfAdmission.Text;
            string YearOfGraduation = txtYearOfGraduation.Text;
            string query = "Insert into ADMISSION (Adm_ID, Field, YearOfAdmission, YearOfGraduation) Values ('" + Adm_Id + "','" + Field + "','" +
            cm1 = new SqlCommand(query, con);
            cm1.ExecuteNonQuery();
            cm1.Dispose();
            con.Close();
            ResetFormControls();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            ResetFormControls();
        }
```


```
        private void ResetFormControls()
        {
            txtAdm_ID.Clear();
            txtField.Clear();
            txtYearOfAdmission.Clear();
            txtYearOfGraduation.Clear();

            txtAdm_ID.Focus();

        }

       private void AdmissionRecordDataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            txtAdm_ID.Text = AdmissionRecordDataGridView.SelectedRows[0].Cells[0].Value.ToString();
            txtField.Text = AdmissionRecordDataGridView.SelectedRows[0].Cells[1].Value.ToString();
            txtYearOfAdmission.Text = AdmissionRecordDataGridView.SelectedRows[0].Cells[2].Value.ToString();
            txtYearOfGraduation.Text = AdmissionRecordDataGridView.SelectedRows[0].Cells[3].Value.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (IsValid())
            {
                con.Open();

                SqlCommand cmd = new SqlCommand("UPDATE [dbo].[ADMISSION] SET Field = @Field, YearOfAdmission = @YearOfAdmission, YearOfGraduation
                cmd.CommandType = CommandType.Text;
```

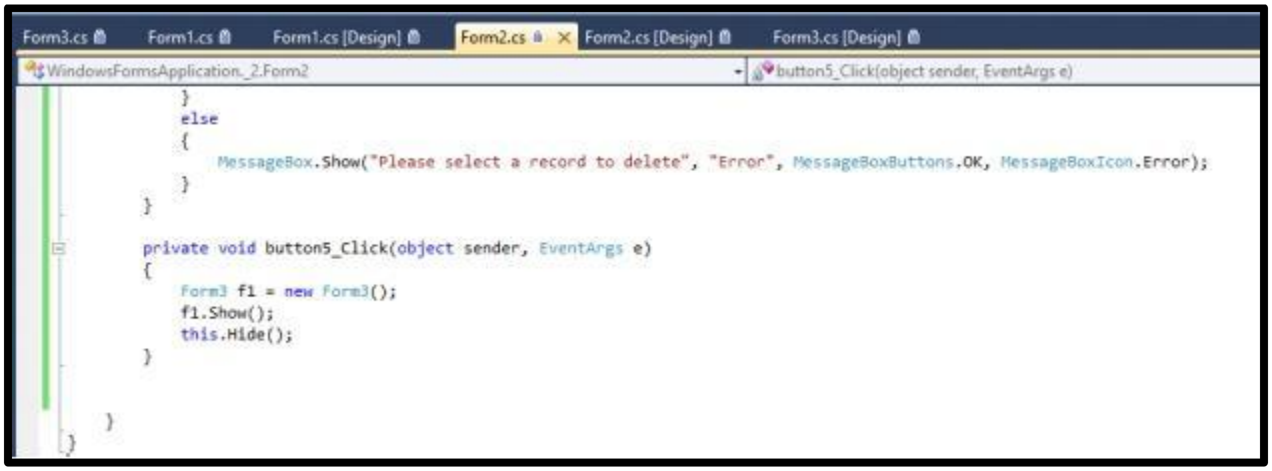


WindowsFormsApplication._2.Form2
button5_Click(object sender, EventArgs e)
100 %

```
                cmd.Parameters.AddWithValue("@Field", txtField.Text);
                cmd.Parameters.AddWithValue("@YearOfAdmission", txtYearOfAdmission.Text);
                cmd.Parameters.AddWithValue("@YearOfGraduation", txtYearOfGraduation.Text);
                cmd.Parameters.AddWithValue("@Adm_ID", txtAdm_ID.Text);

                cmd.ExecuteNonQuery();
                con.Close();

                MessageBox.Show("Admission record is successfully updated in the database", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Informa

                GetAdmissionRecord();
                ResetFormControls();
            }
        }

        private bool IsValid()
        {
            if (string.IsNullOrEmpty(txtField.Text))
            {
                MessageBox.Show("Field is required", "Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }

            return true;
        }

        private void button3_Click(object sender, EventArgs e)
        {
```



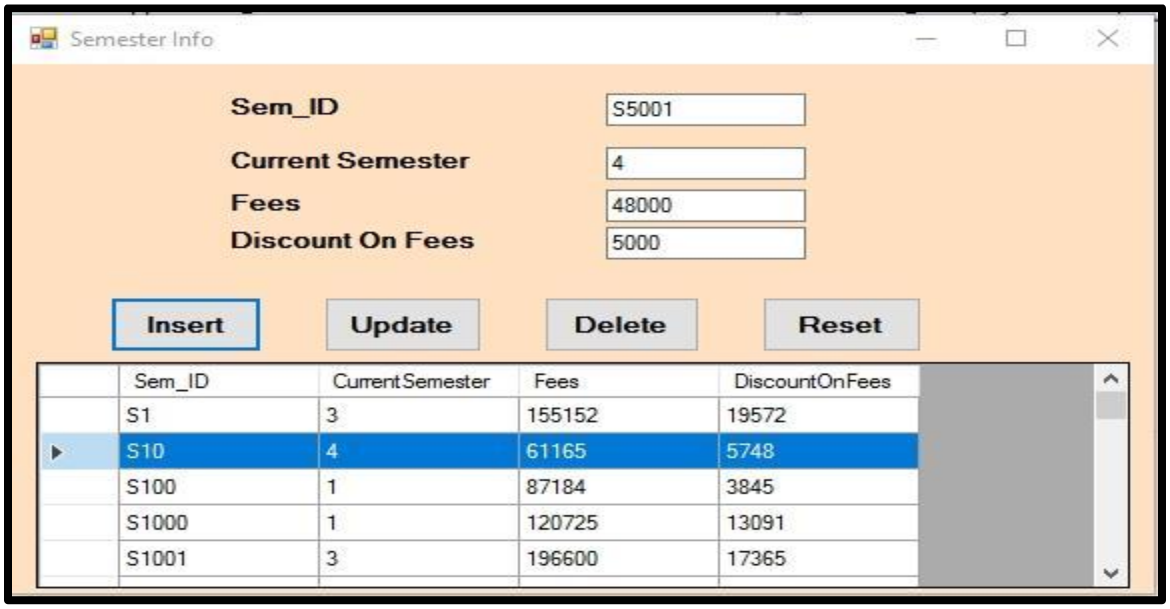
```
        {
            if (AdmissionRecordDataGridView.SelectedRows.Count > 0)
            {
                if (MessageBox.Show("Are you sure you want to delete this record?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Questio
                {
                    int selectedRowIndex = AdmissionRecordDataGridView.SelectedRows[0].Index;
                    string Adm_ID = AdmissionRecordDataGridView.SelectedRows[0].Cells[0].Value.ToString();

                    con.Open();

                    // Delete associated records in the "student" table
                    SqlCommand deleteStudentCmd = new SqlCommand("DELETE FROM [dbo].[student] WHERE Adm_ID = @Adm_ID", con);
                    deleteStudentCmd.Parameters.AddWithValue("@Adm_ID", Adm_ID);
                    deleteStudentCmd.ExecuteNonQuery();

                    // Delete the admission record
                    SqlCommand deleteAdmissionCmd = new SqlCommand("DELETE FROM [dbo].[ADMISSION] WHERE Adm_ID = @Adm_ID", con);
                    deleteAdmissionCmd.Parameters.AddWithValue("@Adm_ID", Adm_ID);
                    deleteAdmissionCmd.ExecuteNonQuery();

                    con.Close();

                    MessageBox.Show("Admission record has been deleted successfully", "Deleted", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    GetAdmissionRecord();
                    ResetFormControls();
                }
            }
```



```
            }
            else
            {
                MessageBox.Show("Please select a record to delete", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            Form3 f1 = new Form3();
            f1.Show();
            this.Hide();
        }


    }
}
```

FOR SEMESTER TABLE

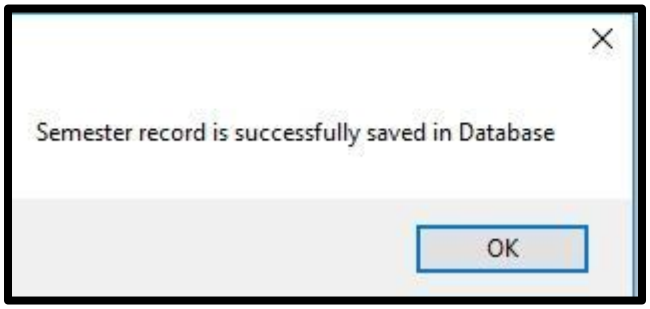
INSERT BUTTON



Semester Info

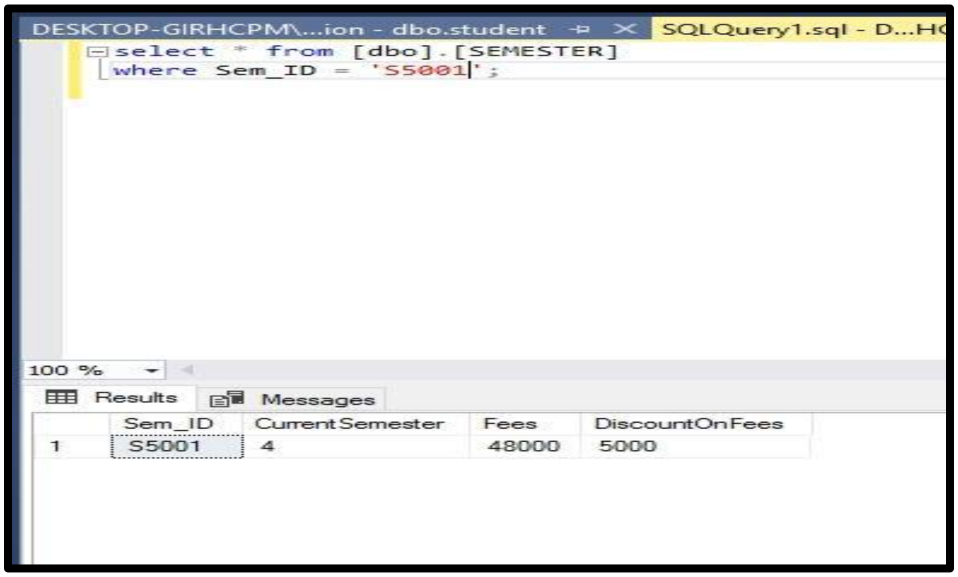
Sem_ID	S5001
Current Semester	4
Fees	48000
Discount On Fees	5000

Sem_ID	CurrentSemester	Fees	DiscountOnFees
S1	3	155152	19572
S10	4	61165	5748
S100	1	87184	3845
S1000	1	120725	13091
S1001	3	196600	17365



Semester record is successfully saved in Database

IN DATABASE SERVER



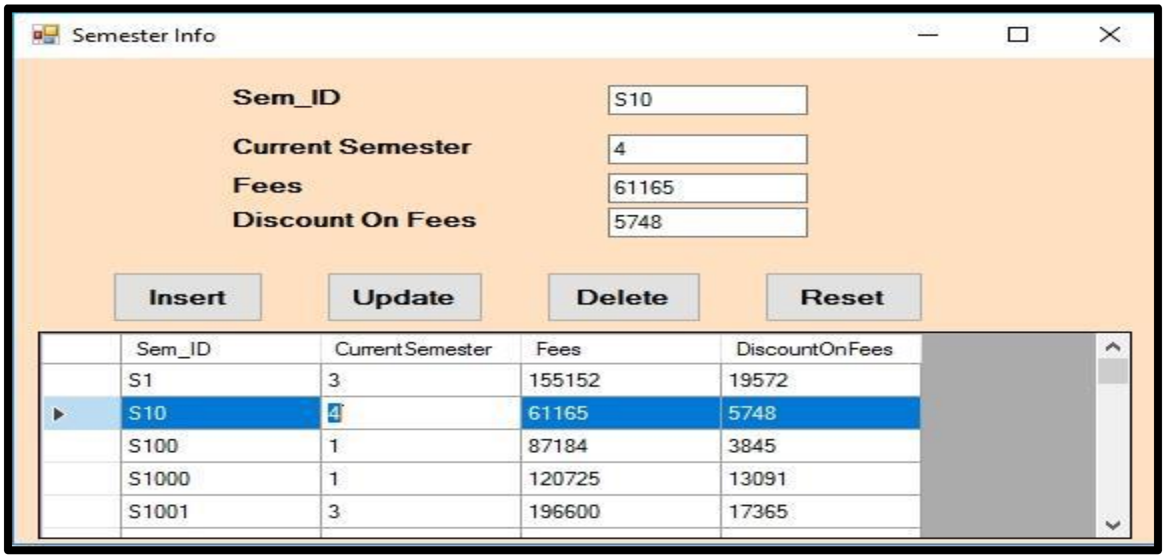
DESKTOP-GIRHCPM\...ion - dbo.student SQLQuery1.sql - D...H

```
select * from [dbo].[SEMESTER]
where Sem_ID = 'S5001';
```

	Sem_ID	CurrentSemester	Fees	DiscountOnFees
1	S5001	4	48000	5000

<u>UPDATE BUTTON</u>

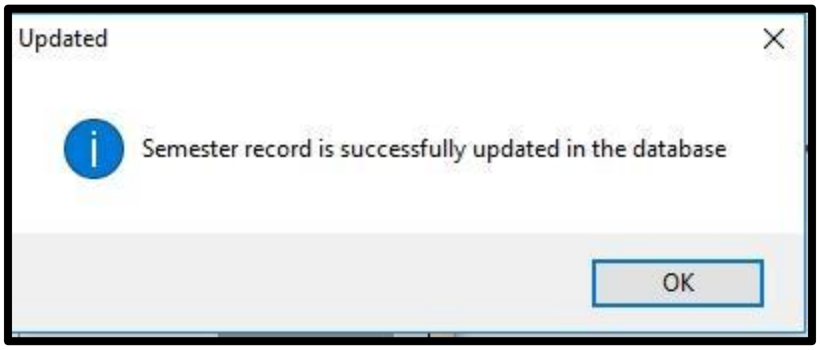
BEFORE UPDATE



Sem_ID	S10
Current Semester	4
Fees	61165
Discount On Fees	5748

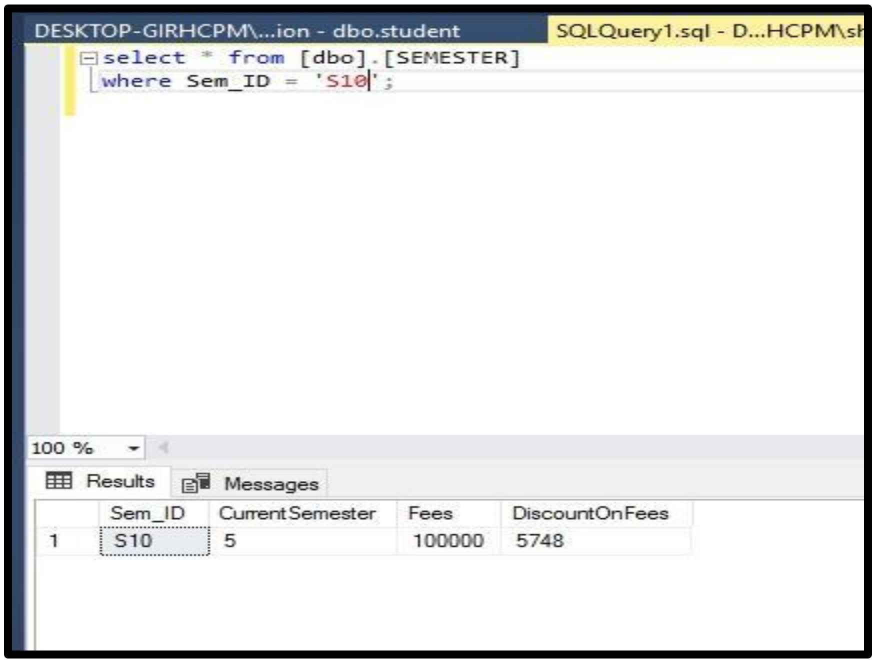
Sem_ID	CurrentSemester	Fees	DiscountOnFees
S1	3	155152	19572
S10	4	61165	5748
S100	1	87184	3845
S1000	1	120725	13091
S1001	3	196600	17365

AFTER UPDATE



Semester record is successfully updated in the database

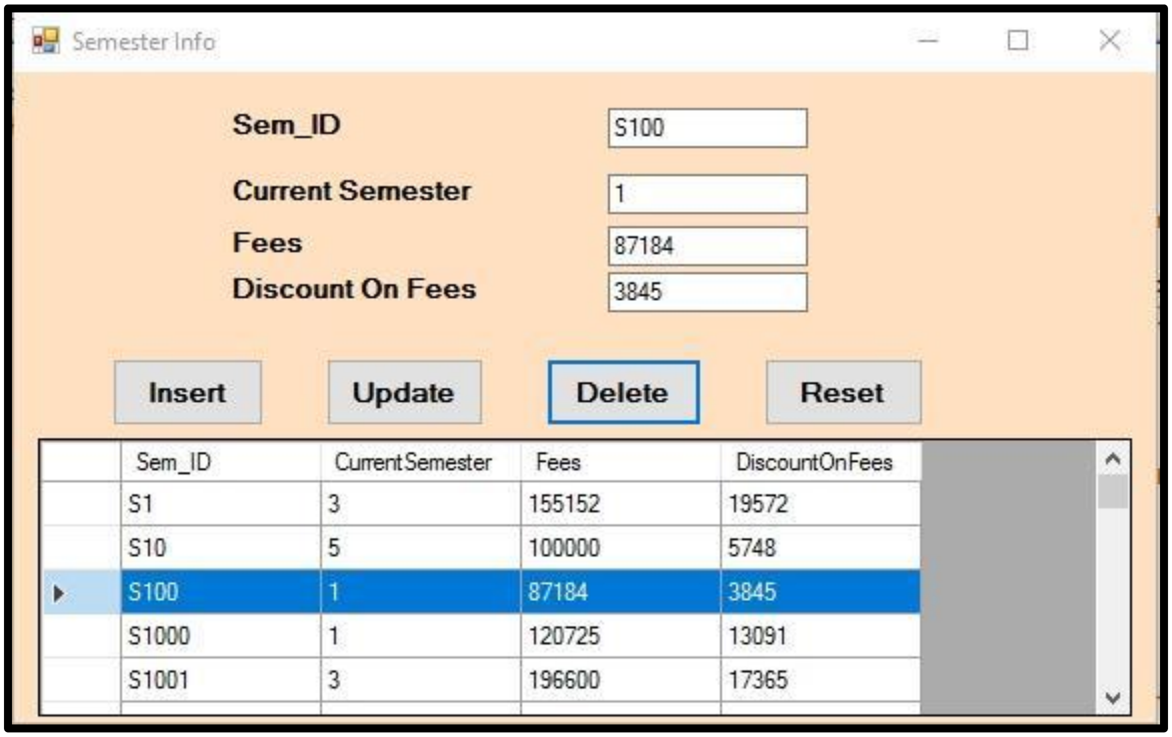
IN DATABASE SERVER


DESKTOP-GIRHCPM\...ion - dbo.student
SQLQuery1.sql - D...HCPM\sh
100 %
Results
Messages

```
select * from [dbo].[SEMESTER]
where Sem_ID = 'S10';
```

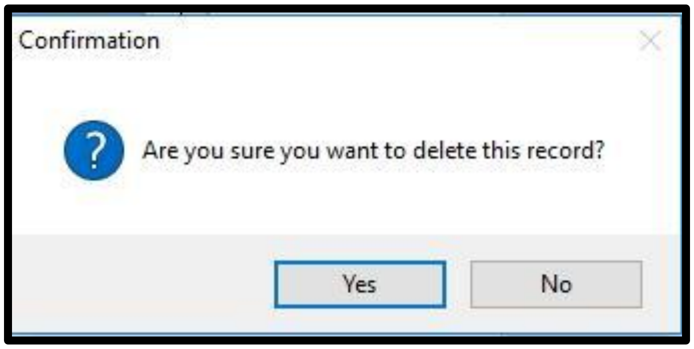
	Sem_ID	CurrentSemester	Fees	DiscountOnFees
1	S10	5	100000	5748

DELETE BUTTON

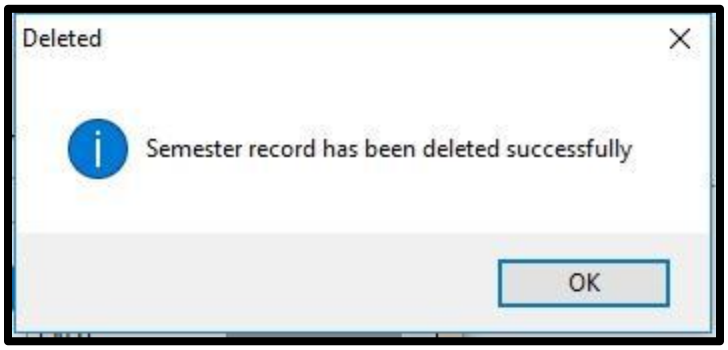


Sem_ID	S100
Current Semester	1
Fees	87184
Discount On Fees	3845

Sem_ID	CurrentSemester	Fees	DiscountOnFees
S1	3	155152	19572
S10	5	100000	5748
S100	1	87184	3845
S1000	1	120725	13091
S1001	3	196600	17365

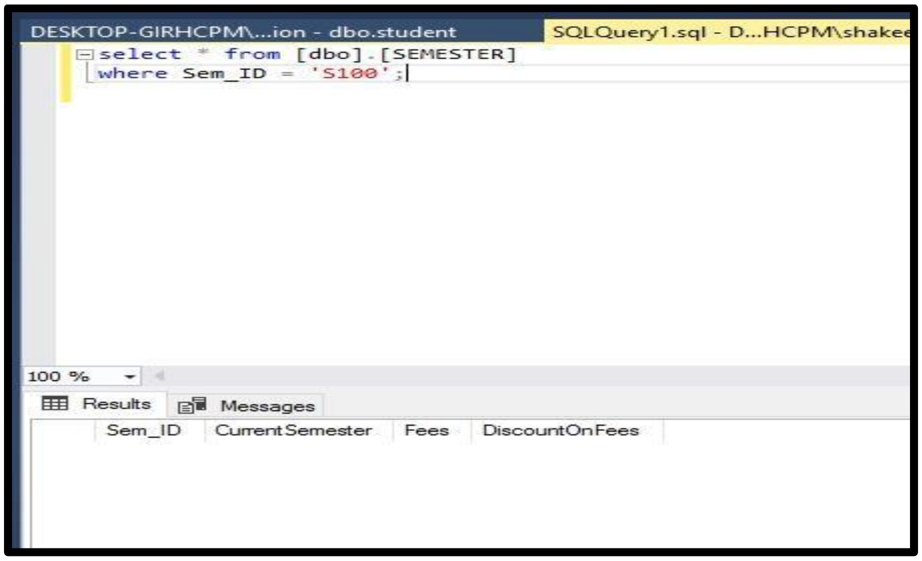


Are you sure you want to delete this record?



Semester record has been deleted successfully

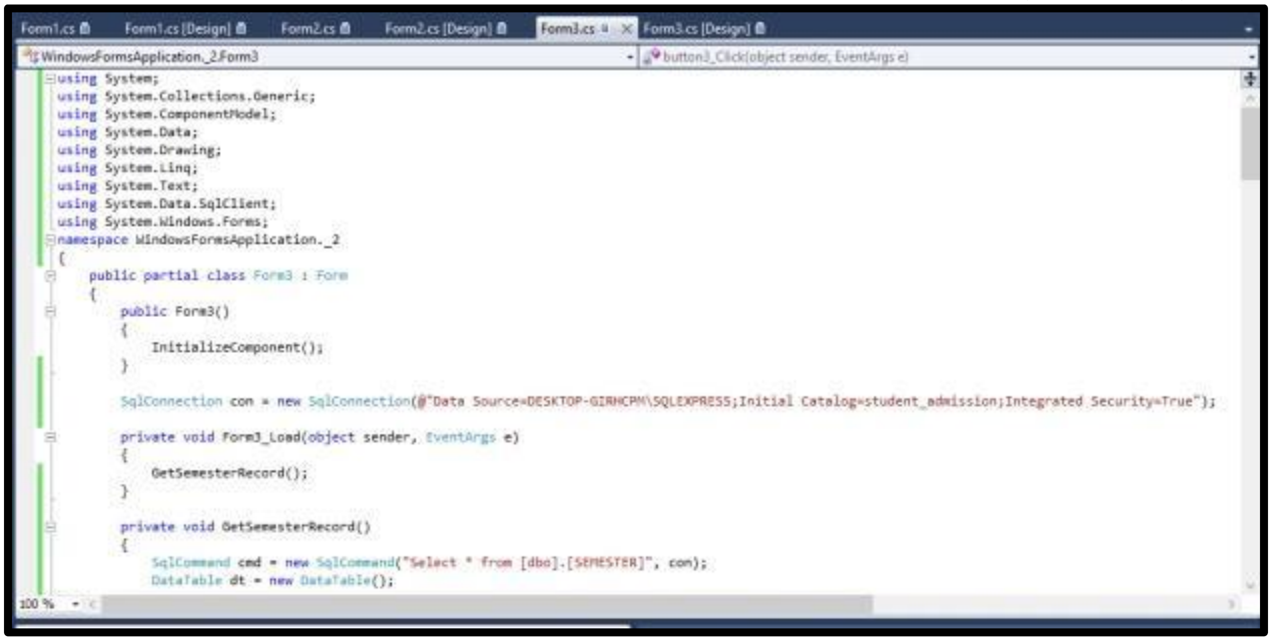
IN DATABASE SERVER


DESKTOP-GIRHCPM\...ion - dbo.student
SQLQuery1.sql - D...HCPM\shakee
100 %
Results
Messages

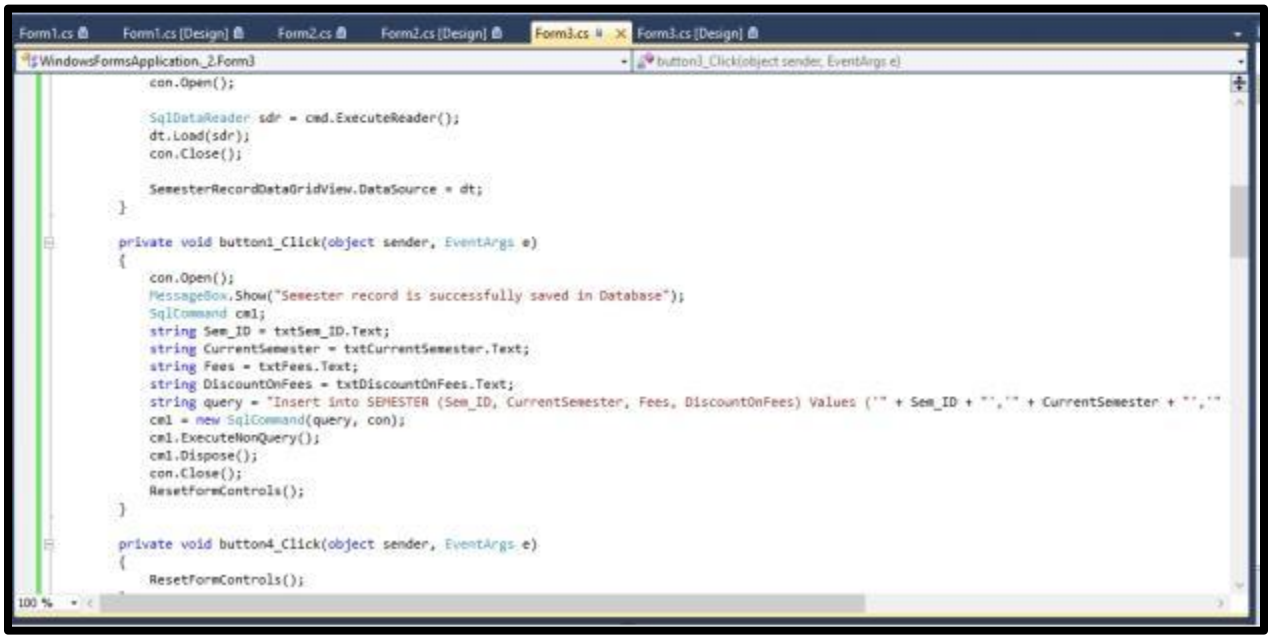
```
select * from [dbo].[SEMESTER]
where Sem_ID = 'S100';
```

Sem_ID	CurrentSemester	Fees	DiscountOnFees

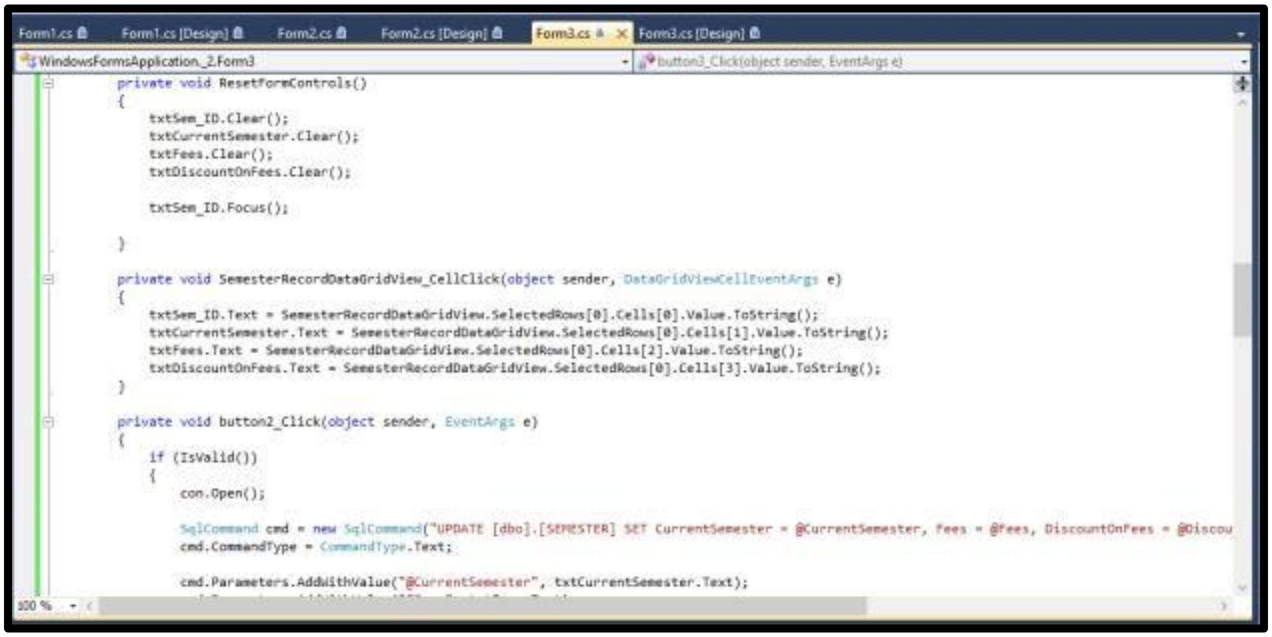
CODE AND QUERIES FOR SEMESTER TABLE: (FORM3)



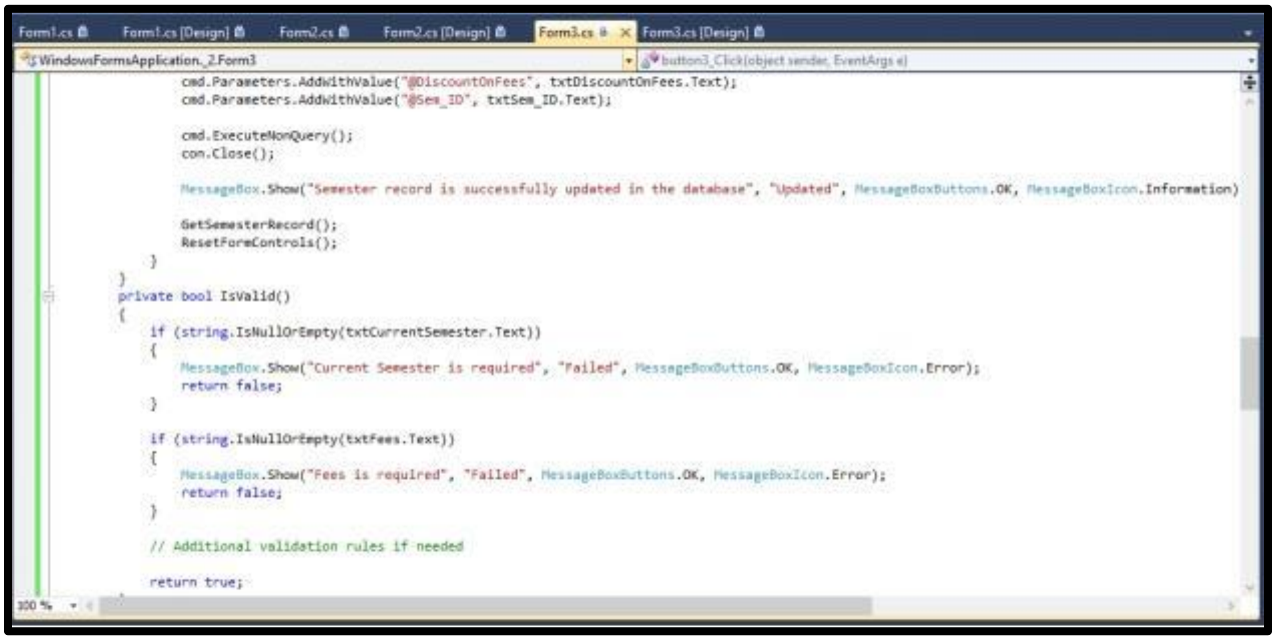
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Windows.Forms;
namespace WindowsFormsApplication._2
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-GIRHCPM\SQLEXPRESS;Initial Catalog=student_admission;Integrated Security=True");

        private void Form3_Load(object sender, EventArgs e)
        {
            GetSemesterRecord();
        }

        private void GetSemesterRecord()
        {
            SqlCommand cmd = new SqlCommand("Select * from [dbo].[SEMESTER]", con);
            DataTable dt = new DataTable();
```



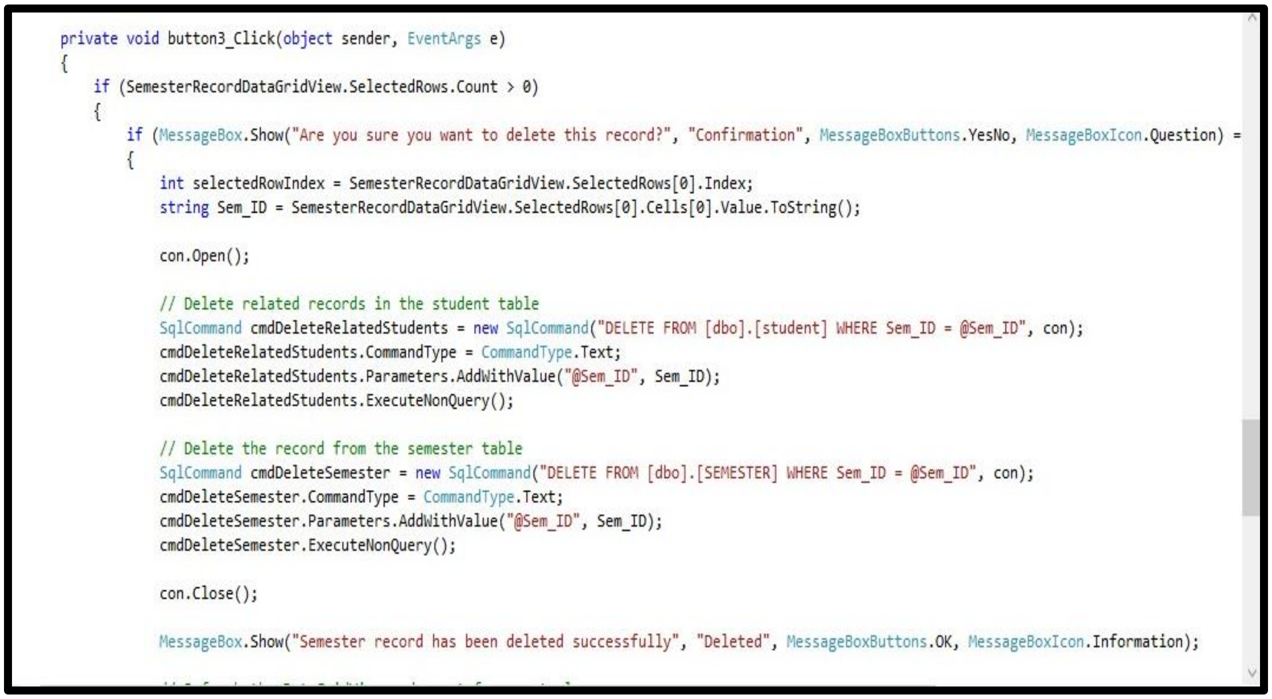
```
            con.Open();

            SqlDataReader sdr = cmd.ExecuteReader();
            dt.Load(sdr);
            con.Close();

            SemesterRecordDataGridView.DataSource = dt;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            con.Open();
            MessageBox.Show("Semester record is successfully saved in Database");
            SqlCommand cm1;
            string Sem_ID = txtSem_ID.Text;
            string CurrentSemester = txtCurrentSemester.Text;
            string Fees = txtFees.Text;
            string DiscountOnFees = txtDiscountOnFees.Text;
            string query = "Insert into SEMESTER (Sem_ID, CurrentSemester, Fees, DiscountOnFees) Values ('" + Sem_ID + "','" + CurrentSemester + "','"
            cm1 = new SqlCommand(query, con);
            cm1.ExecuteNonQuery();
            cm1.Dispose();
            con.Close();
            ResetFormControls();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            ResetFormControls();
```



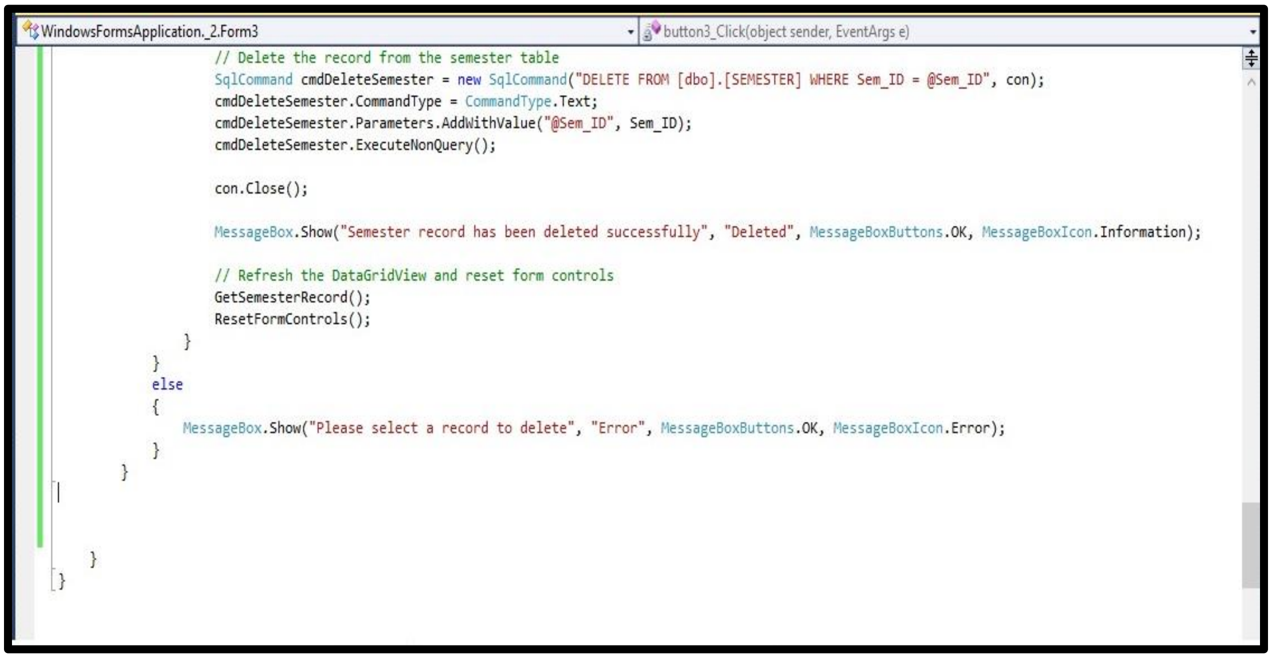
```
        private void ResetFormControls()
        {
            txtSem_ID.Clear();
            txtCurrentSemester.Clear();
            txtFees.Clear();
            txtDiscountOnFees.Clear();

            txtSem_ID.Focus();

        }

        private void SemesterRecordDataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            txtSem_ID.Text = SemesterRecordDataGridView.SelectedRows[0].Cells[0].Value.ToString();
            txtCurrentSemester.Text = SemesterRecordDataGridView.SelectedRows[0].Cells[1].Value.ToString();
            txtFees.Text = SemesterRecordDataGridView.SelectedRows[0].Cells[2].Value.ToString();
            txtDiscountOnFees.Text = SemesterRecordDataGridView.SelectedRows[0].Cells[3].Value.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (IsValid())
            {
                con.Open();

                SqlCommand cmd = new SqlCommand("UPDATE [dbo].[SEMESTER] SET CurrentSemester = @CurrentSemester, Fees = @Fees, DiscountOnFees = @Discou
                cmd.CommandType = CommandType.Text;

                cmd.Parameters.AddWithValue("@CurrentSemester", txtCurrentSemester.Text);
```



```
                cmd.Parameters.AddWithValue("@DiscountOnFees", txtDiscountOnFees.Text);
                cmd.Parameters.AddWithValue("@Sem_ID", txtSem_ID.Text);

                cmd.ExecuteNonQuery();
                con.Close();

                MessageBox.Show("Semester record is successfully updated in the database", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

                GetSemesterRecord();
                ResetFormControls();
            }
        }
        private bool IsValid()
        {
            if (string.IsNullOrEmpty(txtCurrentSemester.Text))
            {
                MessageBox.Show("Current Semester is required", "Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }

            if (string.IsNullOrEmpty(txtFees.Text))
            {
                MessageBox.Show("Fees is required", "Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }

            // Additional validation rules if needed

            return true;
```



```
private void button3_Click(object sender, EventArgs e)
{
    if (SemesterRecordDataGridView.SelectedRows.Count > 0)
    {
        if (MessageBox.Show("Are you sure you want to delete this record?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question) =
        {
            int selectedRowIndex = SemesterRecordDataGridView.SelectedRows[0].Index;
            string Sem_ID = SemesterRecordDataGridView.SelectedRows[0].Cells[0].Value.ToString();

            con.Open();

            // Delete related records in the student table
            SqlCommand cmdDeleteRelatedStudents = new SqlCommand("DELETE FROM [dbo].[student] WHERE Sem_ID = @Sem_ID", con);
            cmdDeleteRelatedStudents.CommandType = CommandType.Text;
            cmdDeleteRelatedStudents.Parameters.AddWithValue("@Sem_ID", Sem_ID);
            cmdDeleteRelatedStudents.ExecuteNonQuery();

            // Delete the record from the semester table
            SqlCommand cmdDeleteSemester = new SqlCommand("DELETE FROM [dbo].[SEMESTER] WHERE Sem_ID = @Sem_ID", con);
            cmdDeleteSemester.CommandType = CommandType.Text;
            cmdDeleteSemester.Parameters.AddWithValue("@Sem_ID", Sem_ID);
            cmdDeleteSemester.ExecuteNonQuery();

            con.Close();

            MessageBox.Show("Semester record has been deleted successfully", "Deleted", MessageBoxButtons.OK, MessageBoxIcon.Information);
```



WindowsFormsApplication._2.Form3
button3_Click(object sender, EventArgs e)

```
            // Delete the record from the semester table
            SqlCommand cmdDeleteSemester = new SqlCommand("DELETE FROM [dbo].[SEMESTER] WHERE Sem_ID = @Sem_ID", con);
            cmdDeleteSemester.CommandType = CommandType.Text;
            cmdDeleteSemester.Parameters.AddWithValue("@Sem_ID", Sem_ID);
            cmdDeleteSemester.ExecuteNonQuery();

            con.Close();

            MessageBox.Show("Semester record has been deleted successfully", "Deleted", MessageBoxButtons.OK, MessageBoxIcon.Information);

            // Refresh the DataGridView and reset form controls
            GetSemesterRecord();
            ResetFormControls();
        }
    }
    else
    {
        MessageBox.Show("Please select a record to delete", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}


    }
}
```

DATA FLOW MODEL

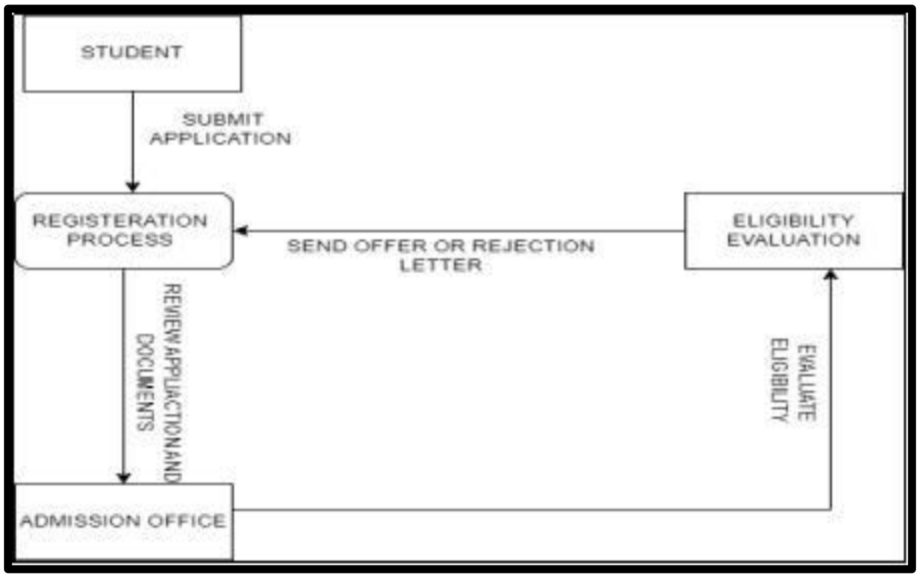
A data flow model is a visual representation of how data moves and transforms within a system or process. It depicts the flow of data, the processes that operate on the data, and the entities involved in the system. Data flow models are commonly used in system analysis and design to understand and communicate how information is processed and exchanged.

TYPES OF DATA FLOW MODEL

- Context Diagram (level 0)
- Level 1 Diagram
- Level 2 Diagram

LEVEL: 0

A context model at level 0 provides a high-level overview of the system and its external entities. It helps to understand the interactions between the system and its external entities without going into detailed processes or data flows.



EXPLANATION

The **"Student"** entity submits an application to the **Admission Office** entity. The **Admission Office** entity then proceeds to review the application and supporting documents. After the review process, the application moves to the **Review Process** where the eligibility of the student is evaluated. The evaluation of eligibility takes place in the **Eligibility Evaluation** process. Based on the outcome of the evaluation, the Eligibility Evaluation process sends either an offer letter or a rejection letter. The process can loop back, allowing the **Admission Office** entity to review additional applications and repeat the evaluation process.

LEVEL: 1

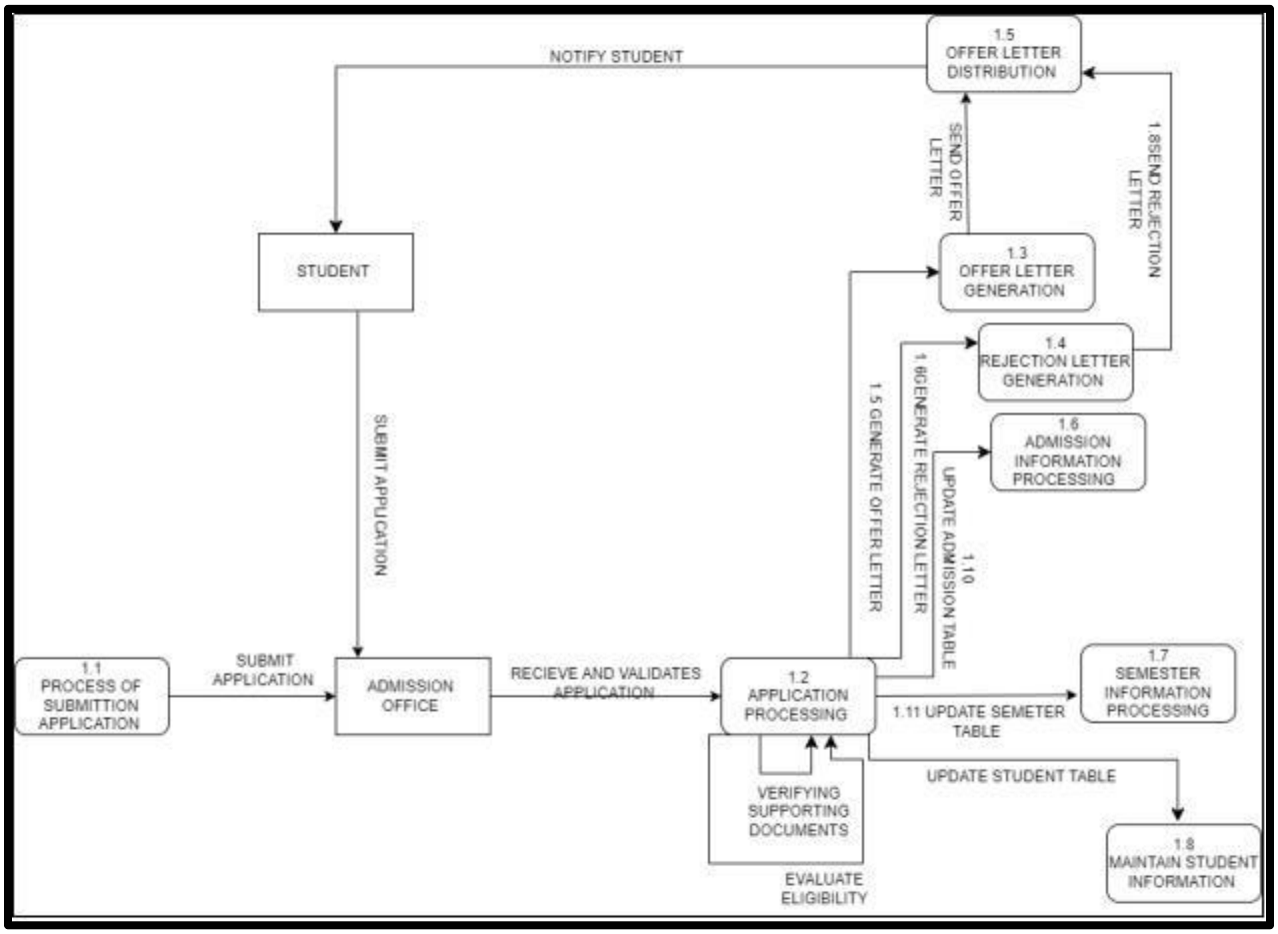
The level 1 data flow model provides a visual representation of the data flows and processes within the IIT Admission System. It helps stakeholders to understand the sequence of activities and interactions between entities, supporting the analysis and improvement of the admission process.

By breaking down the admission process into distinct steps and interactions, the level 1 data flow model offers a detailed overview of how data moves and processes are executed within the system.

EXPLANATION

The level 1 data flow model provides a more detailed representation of the **II Admission System,** focusing on the data flows and processes involved in the admission process. It showcases the interactions between various entities, such as students, the admission office, and the associated processes and data tables.

In this model, the process starts with a student submitting an application to the admission office. The admission office then receives and validates the application, verifies supporting documents, and evaluates the eligibility of the student. Based on the evaluation, the system generates either an offer letter or a rejection letter. The offer letter is sent to the student, while the rejection letter is also sent to the student with appropriate notification. The admission office also updates the admission table, semester table, and student table with the relevant information during the application processing.


1.5 OFFER LETTER DISTRIBUTION
NOTIFY STUDENT
SEND OFFER LETTER
1.8SEND REJECTION LETTER
STUDENT
1.3 OFFER LETTER GENERATION
1.4 REJECTION LETTER GENERATION
1.6 ADMISSION INFORMATION PROCESSING
SUBMIT APPLICATION
1.5 GENERATE OFFER LETTER
1.6GENERATE REJECTION LETTER
1.10 UPDATE ADMISSION TABLE
1.1 PROCESS OF SUBMITTION APPLICATION
SUBMIT APPLICATION
ADMISSION OFFICE
RECIEVE AND VALIDATES APPLICATION
1.2 APPLICATION PROCESSING
1.11 UPDATE SEMETER TABLE
1.7 SEMESTER INFORMATION PROCESSING
UPDATE STUDENT TABLE
VERIFYING SUPPORTING DOCUMENTS
EVALUATE ELIGIBILITY
1.8 MAINTAIN STUDENT INFORMATION

LEVEL: 2

In the context of the IIT Admission System, the level 2 data flow model would involve breaking down the sub-processes from the level 1 diagram into more detailed components. This decomposition allows for a more comprehensive understanding of the system's activities, data flows, and data transformations.

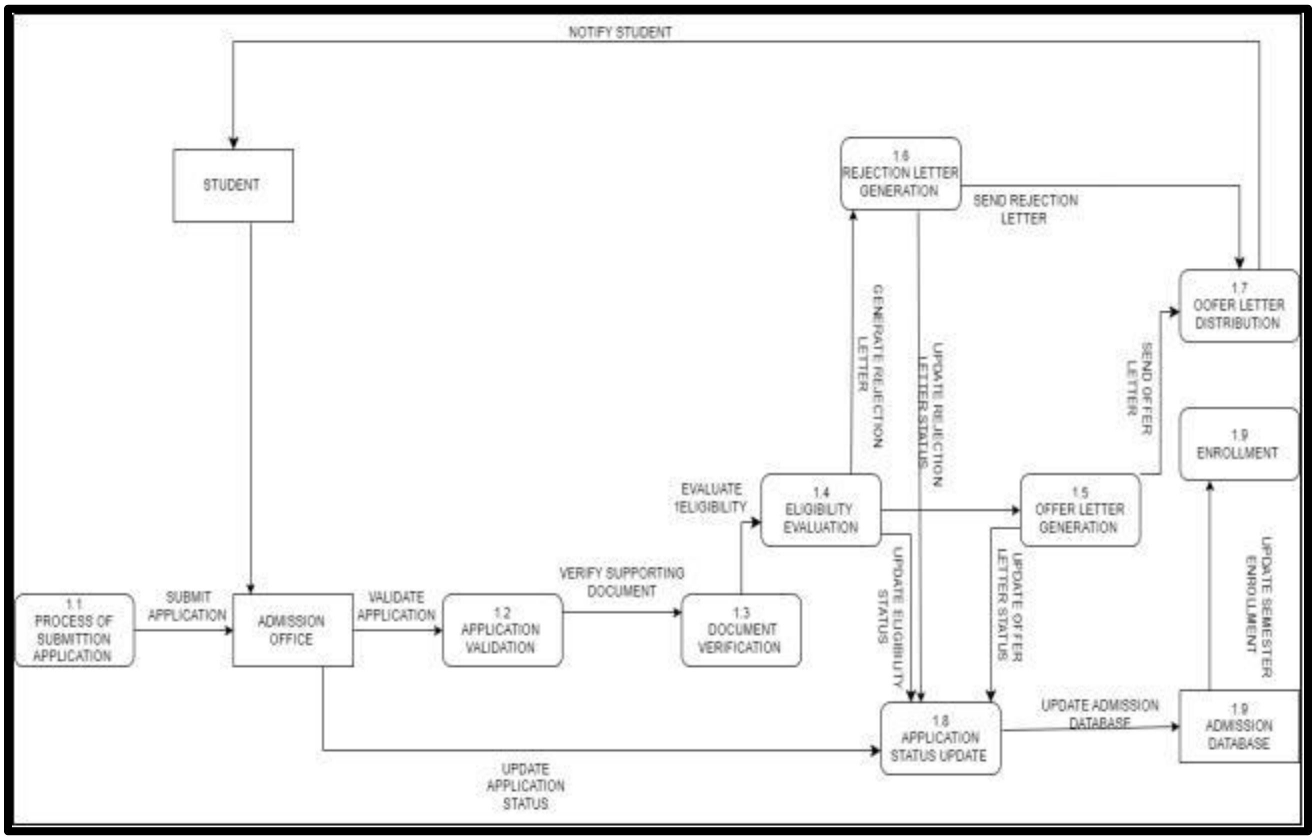
EXAMPLE

IN the IIT Admission System, the level 1 diagram may have a process called "Application Processing" which includes activities such as application validation, document verification, eligibility evaluation, and letter generation. In the level 2 diagram, each of these activities can be further decomposed into smaller components.

EXPLANATION

The processes have been expanded to include more specific actions related to application validation, document verification, eligibility evaluation, and letter generation. The Application Status Update process is introduced to update the status of the application based on the validation and verification results.

The Admission Database represents the central database where admission-related data is stored.The Semester Enrollment represents the process of enrolling students in the appropriate semester.


NOTIFY STUDENT
STUDENT
1.6 REJECTION LETTER GENERATION
SEND REJECTION LETTER
1.7 OOFER LETTER DISTRIBUTION
GENERATE REJECTION LETTER
UPDATE REJECTION LETTER STATUS
SEND OFFER LETTER
1.9 ENROLLMENT
EVALUATE 1ELIGIBILITY
1.4 ELIGIBILITY EVALUATION
1.5 OFFER LETTER GENERATION
UPDATE SEMESTER ENROLLMENT
VERIFY SUPPORTING DOCUMENT
1.1 PROCESS OF SUBMITTION APPLICATION
SUBMIT APPLICATION
ADMISSION OFFICE
VALIDATE APPLICATION
1.2 APPLICATION VALIDATION
1.3 DOCUMENT VERIFICATION
UPDATE ELIGIBILITY STATUS
UPDATE OFFER LETTER STATUS
UPDATE ADMISSION DATABASE
1.9 ADMISSION DATABASE
1.8 APPLICATION STATUS UPDATE
UPDATE APPLICATION STATUS