**LIBRARY MANAGEMENT SYSTEM**

**AIM: Implementation of Parallel Join & Parallel Sort**

**INTRODUCTION:** This project maintains a library management database having four kinds of tables in it. The tables are mentioned as follow:

1. **Books** – This table basically contains all information related to the books in the library.  
   Attributes: Book ID, Title, Author, Price, Availability
2. **Member** – This table contains information on all library members and users.  
   Attributes: Member ID, Name, Membership Date, Expiry Date, Membership type
3. **Publisher** – This table contains information entirely about various publishers who provide books to the libraries.  
   Attributes: Publisher ID, Name, Address
4. **Borrow** – This table basically contains information on which books have been borrowed by which members.  
   Attributes: Book ID, Member ID, Due Date, Return Date, Issue

**SOFTWARE AND TOOLS USED:**

1. **Visual Studio 2017 with .NET framework 4.6.1**  
   - used to create Windows forms
2. **Oracle Database Desktop class 12c**  
   - used to create an Oracle database
3. **ODP.NET Managed Driver**  
   - used to create a database connection to the windows forms

**SYSTEM REQUIREMENTS:**

1. **Visual Studio 2017 with .NET framework 4.6.1**

**Supported Operating Systems:** Visual Studio 2017 will install and run on the following operating systems:

* Windows 10 version 1507 or higher: Home, Professional, Education, and Enterprise (LTSC and S are not supported)
* Windows Server 2016: Standard and Datacentre
* Windows 8.1 (with Update 2919355): Core, Professional, and Enterprise
* Windows Server 2012 R2 (with Update 2919355): Essentials, Standard, Datacentre
* Windows 7 SP1 (with latest Windows Updates): Home Premium, Professional, Enterprise, Ultimate

**Hardware:**

* 1.8 GHz or faster processor. Dual-core or better recommended
* 2 GB of RAM; 4 GB of RAM recommended (2.5 GB minimum if running on a virtual machine)
* Hard disk space: up to 130 GB of available space, depending on features installed; typical installations require 20-50 GB of free space.
* Hard disk speed: to improve performance, install Windows and Visual Studio on a solid-state drive (SSD).
* Video card that supports a minimum display resolution of 720p (1280 by 720); Visual Studio will work best at a resolution of WXGA (1366 by 768) or higher.

**Additional Requirements:**

* .NET Framework 4.5.2 or above is required to **install** Visual Studio. Visual Studio requires .NET Framework 4.7.2 to run, but this will be installed during setup.

1. **Oracle Database Desktop class 12c**

**Operating system general requirements:** Oracle Database for Windows x64 is supported on the following operating system versions:

* Windows 7 x64 - Professional, Enterprise, and Ultimate editions
* Windows 8 x64 and Windows 8.1 x64 - Pro and Enterprise editions
* Windows 8.1 x64 - Pro and Enterprise editions
* Windows 10 x64 - Pro, Enterprise, and Education editions
* Windows Server 2012 x64 - Standard, Datacentre, Essentials, and Foundation editions
* Windows Server 2012 R2 x64 - Standard, Datacentre, Essentials, and Foundation editions
* Windows Server 2016 x64 - Standard, Datacentre, and Essentials editions

**System Architecture: Processor:** AMD64 and Intel EM64T

**Physical memory (RAM):** 2 GB minimum

**Virtual memory (swap):**

* If physical memory is between 2 GB and 16 GB, then set virtual memory to 1 time the size of the RAM
* If physical memory is more than 16 GB, then set virtual memory to 16 GB

**Disk space:**

* Typical Install Type total: **10 GB**
* Advanced Install Types total: **10 GB**

**Video adapter:** 256 colours

**Screen Resolution:** 1024 X 768 minimum

**PROCEDURE:**

Backend:

1. Open SQL Plus for Oracle and login with credentials
2. **Create table:**

**CREATE TABLE BOOKS(**

**BOOK\_ID NUMBER(5) NOT NULL PRIMARY KEY,**

**TITLE VARCHAR2(50),**

**AUTHOR VARCHAR2(10),**

**PRICE NUMBER(4),**

**AVAILABLE CHAR(1)**

**);**

**CREATE TABLE PUBLISHER(**

**PUB\_ID NUMBER(5) NOT NULL PRIMARY KEY,**

**NAME VARCHAR2(10),**

**ADDRESS VARCHAR2(50)**

**);**

**CREATE TABLE MEMBER(**

**MEMBER\_ID NUMBER(5) NOT NULL PRIMARY KEY,**

**NAME VARCHAR2(10),**

**MEM\_DATE DATE,**

**EXP\_DATE DATE,**

**MEM\_TYPE CHAR(1)**

**);**

**CREATE TABLE BORROW(**

**BOOK\_ID NUMBER(5) NOT NULL FOREIGN KEY REFERENCES BOOKS(BOOK\_ID),**

**MEMBER\_ID NUMBER(5) NOT NULL FOREIGN KEY REFERENCES MEMBER(MEMBER\_ID),**

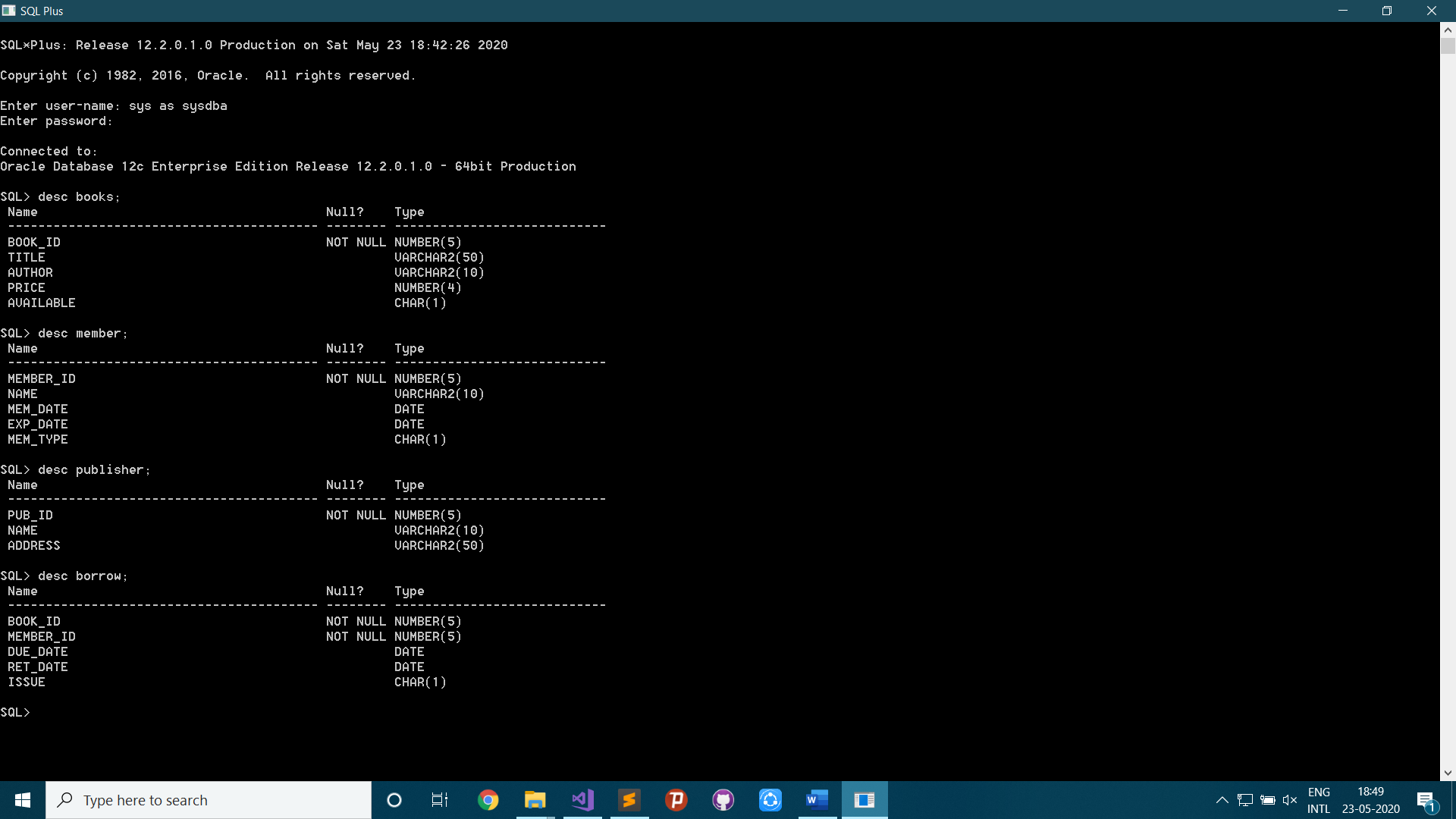
**DUE\_DATE DATE,**

**RET\_DATE DATE,**

**ISSUE CHAR(1)**

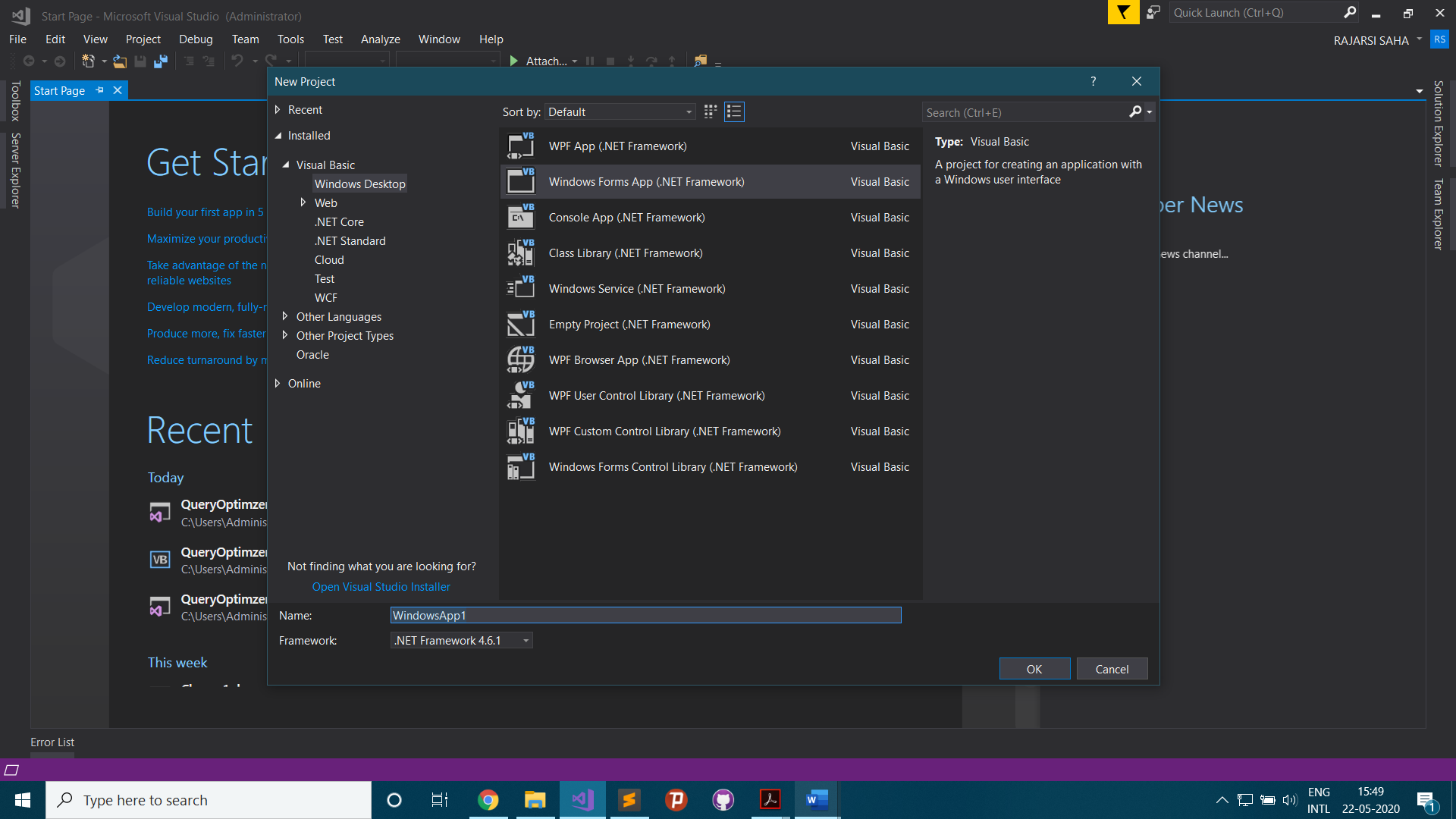
**);**

1. Schema: A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

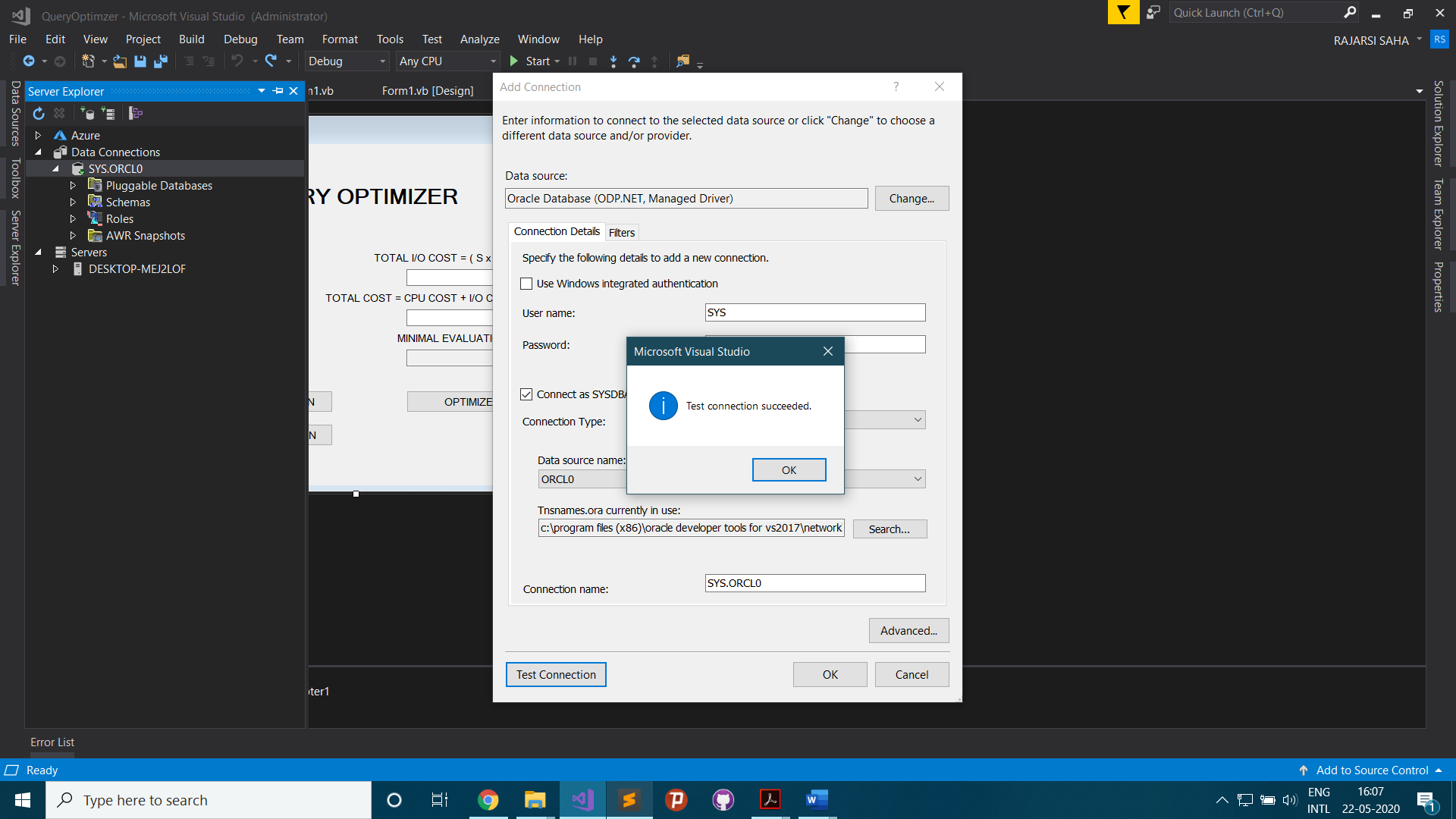


Frontend:

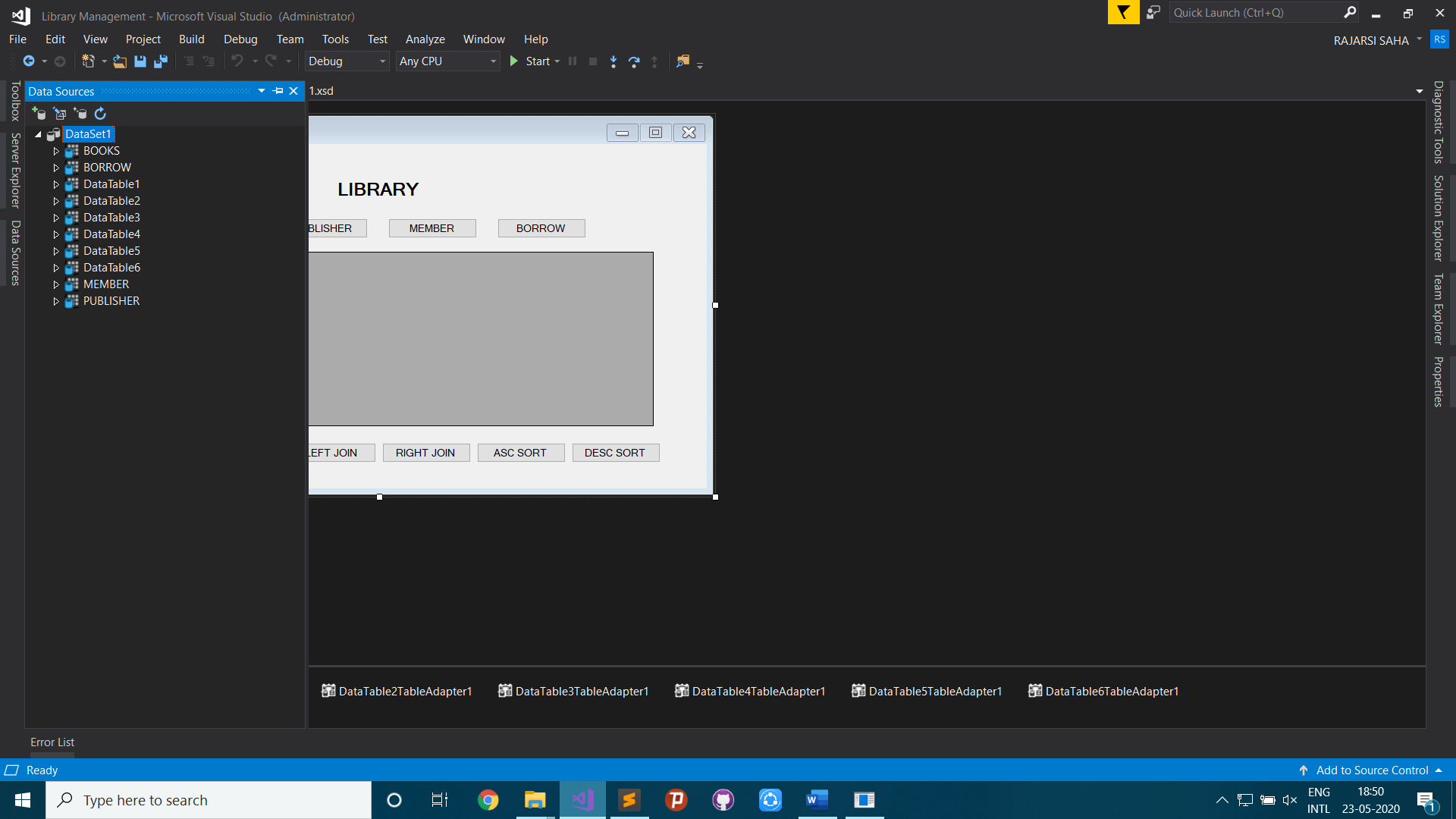
1. Create new Windows Forms App (.NET Framework) in VB Project



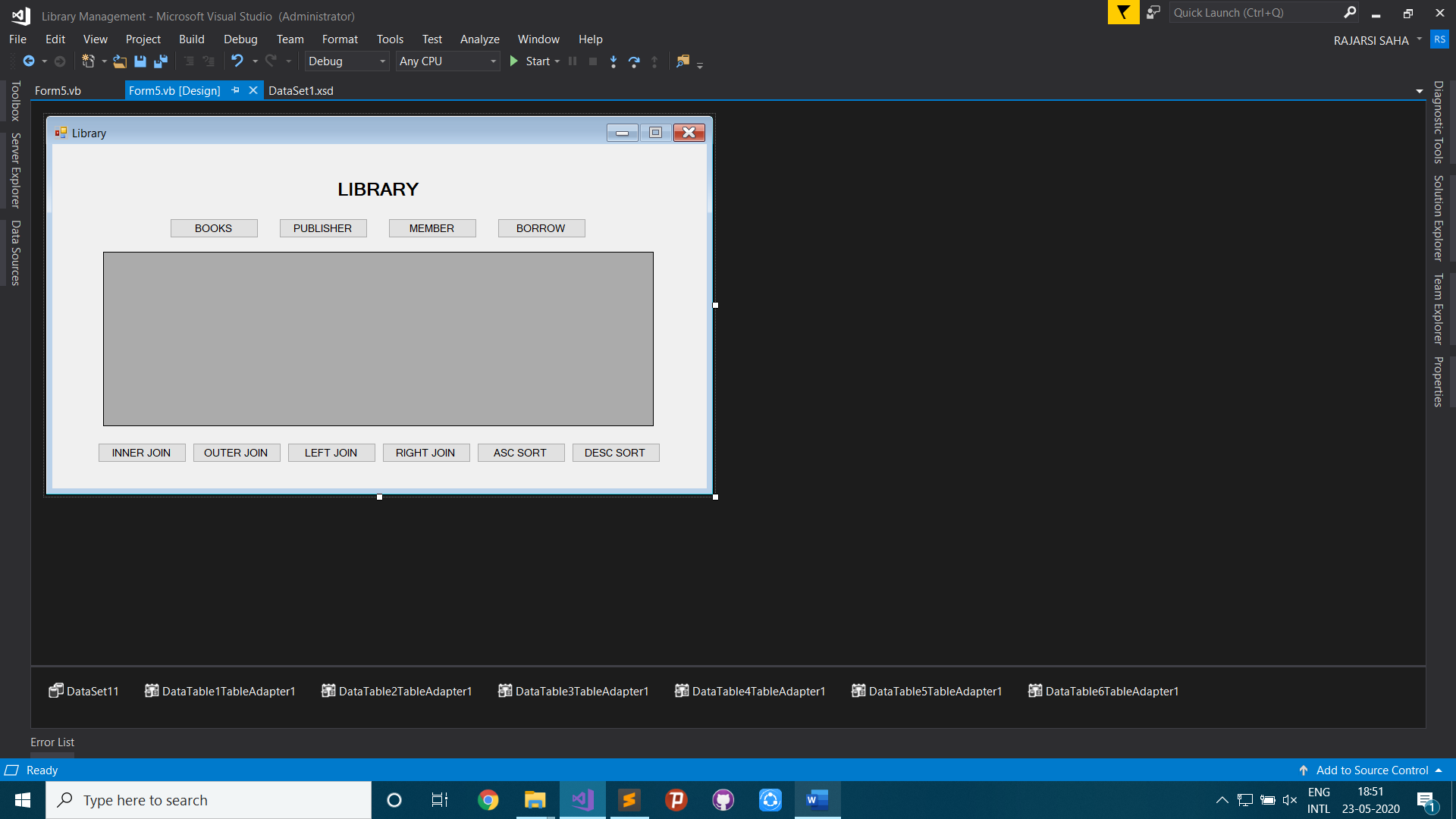
1. Create Database Connection



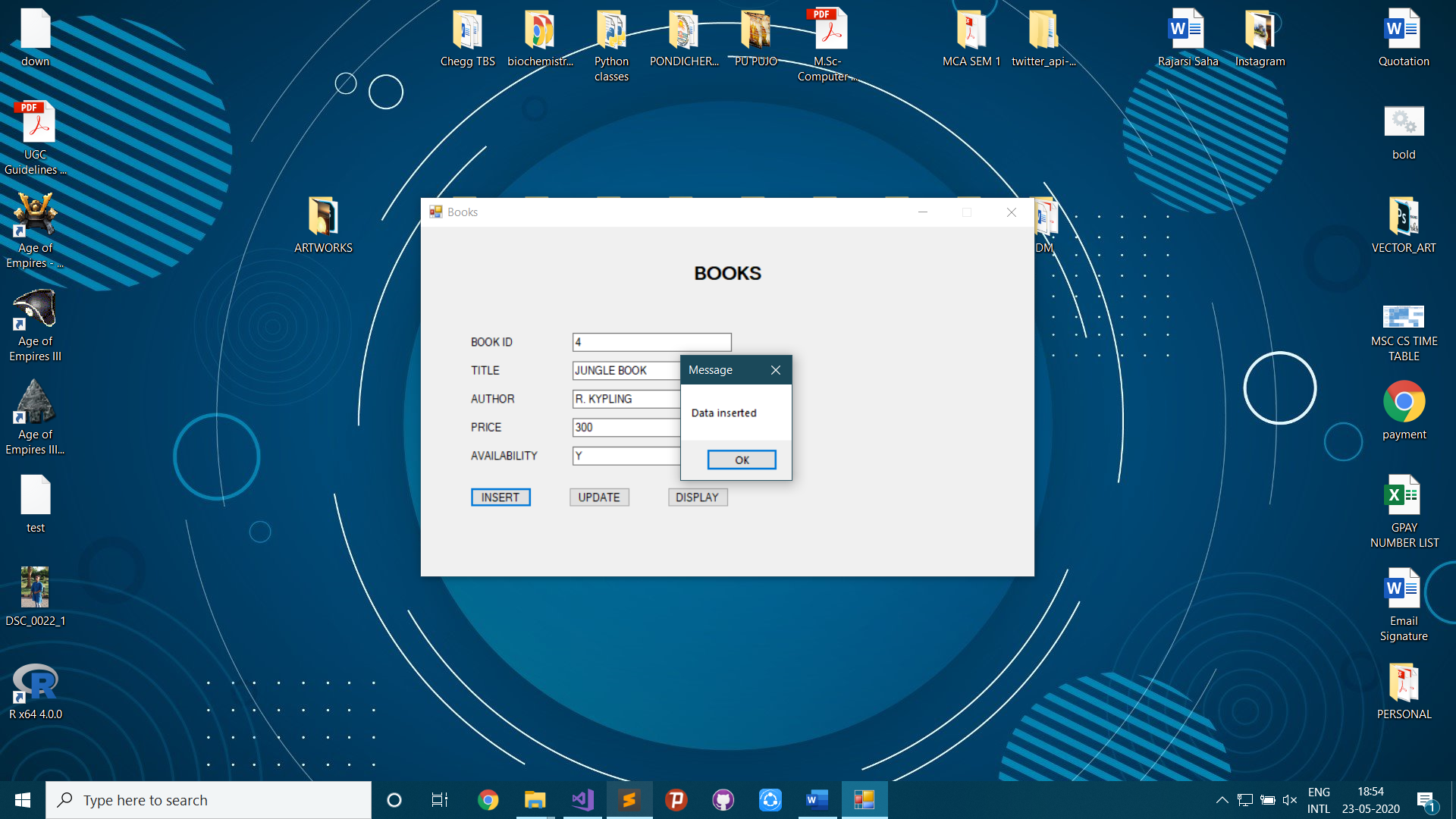
1. Add Dataset



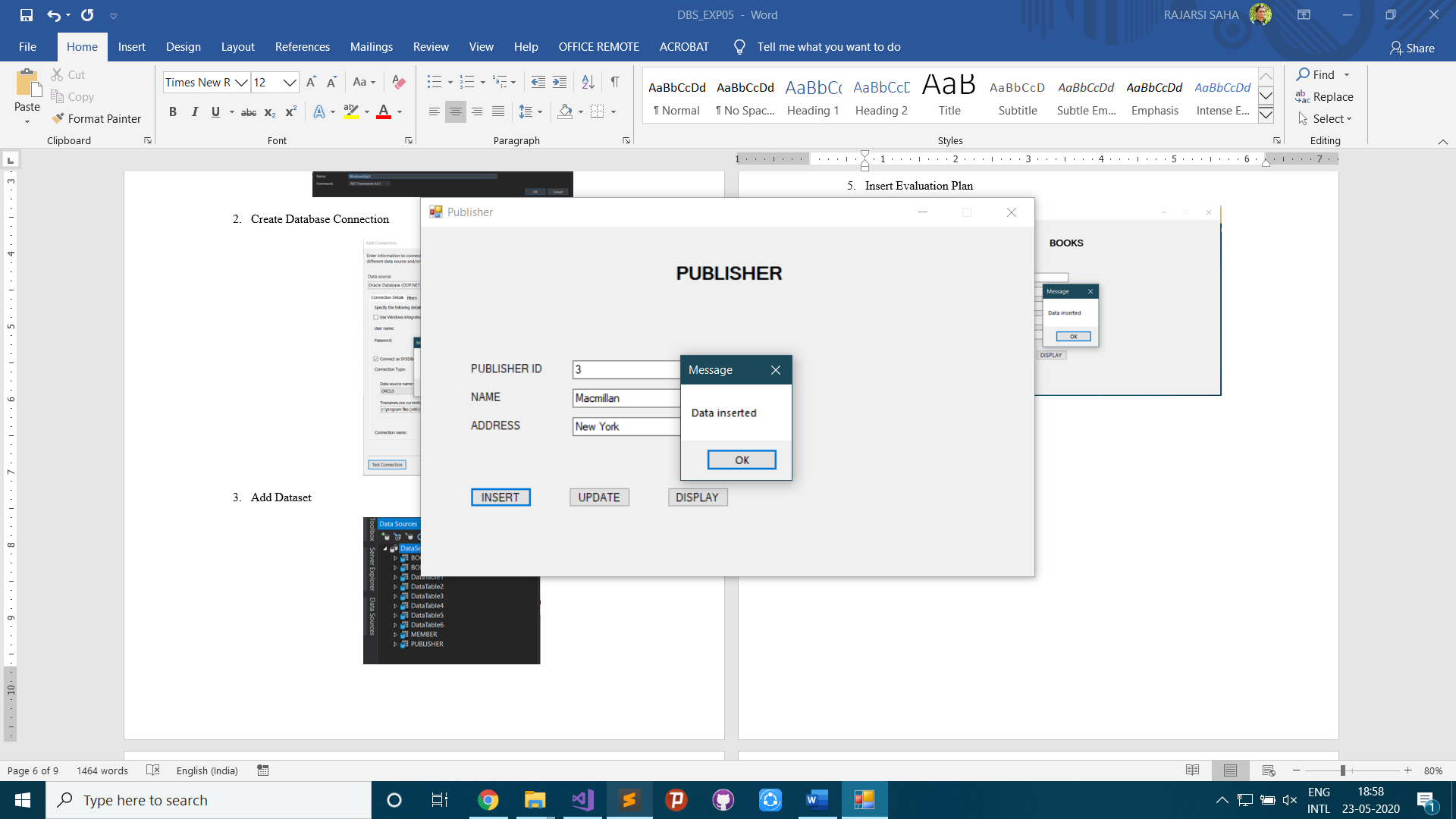
1. Front-end Design



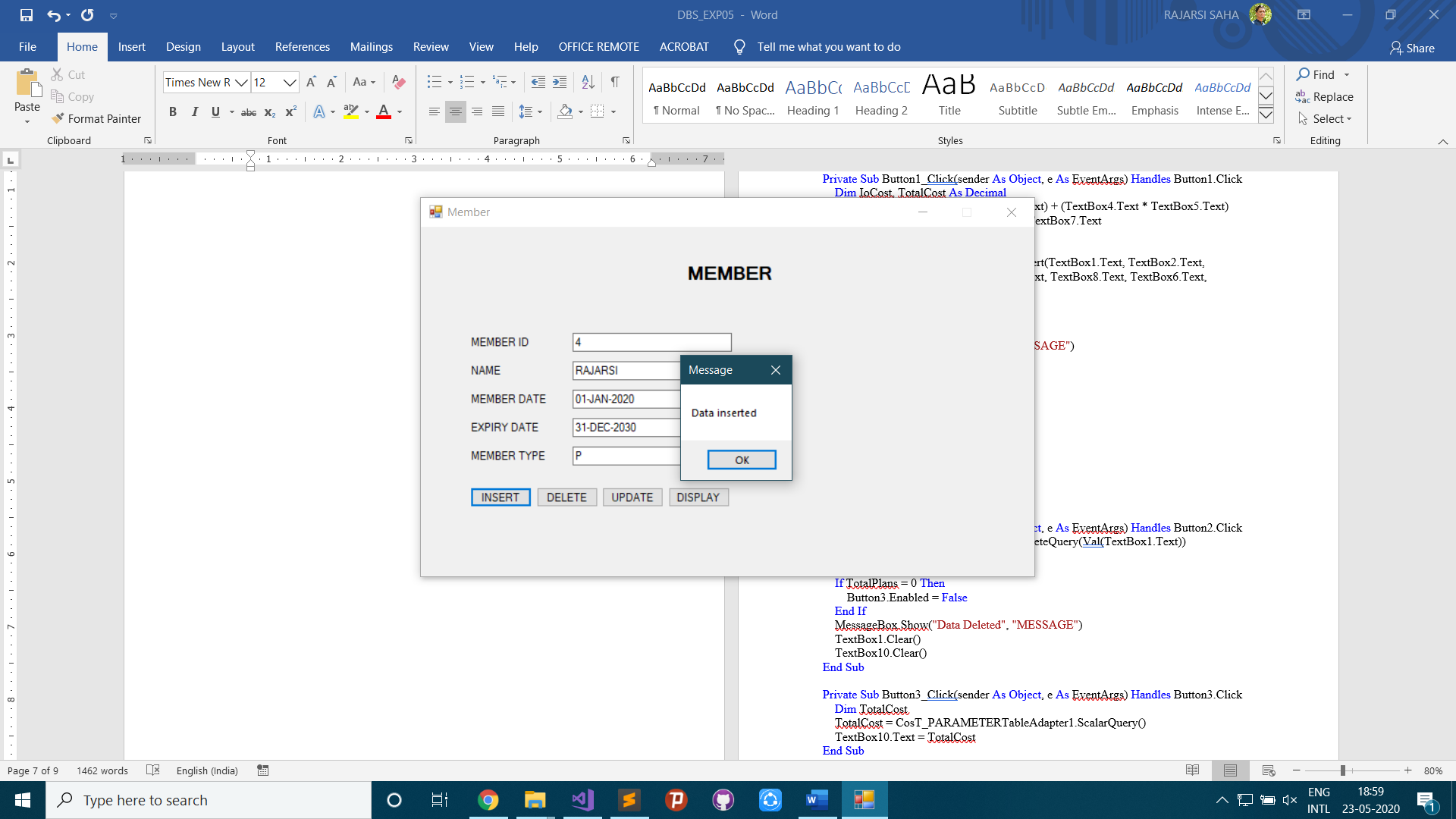
1. Insert Books



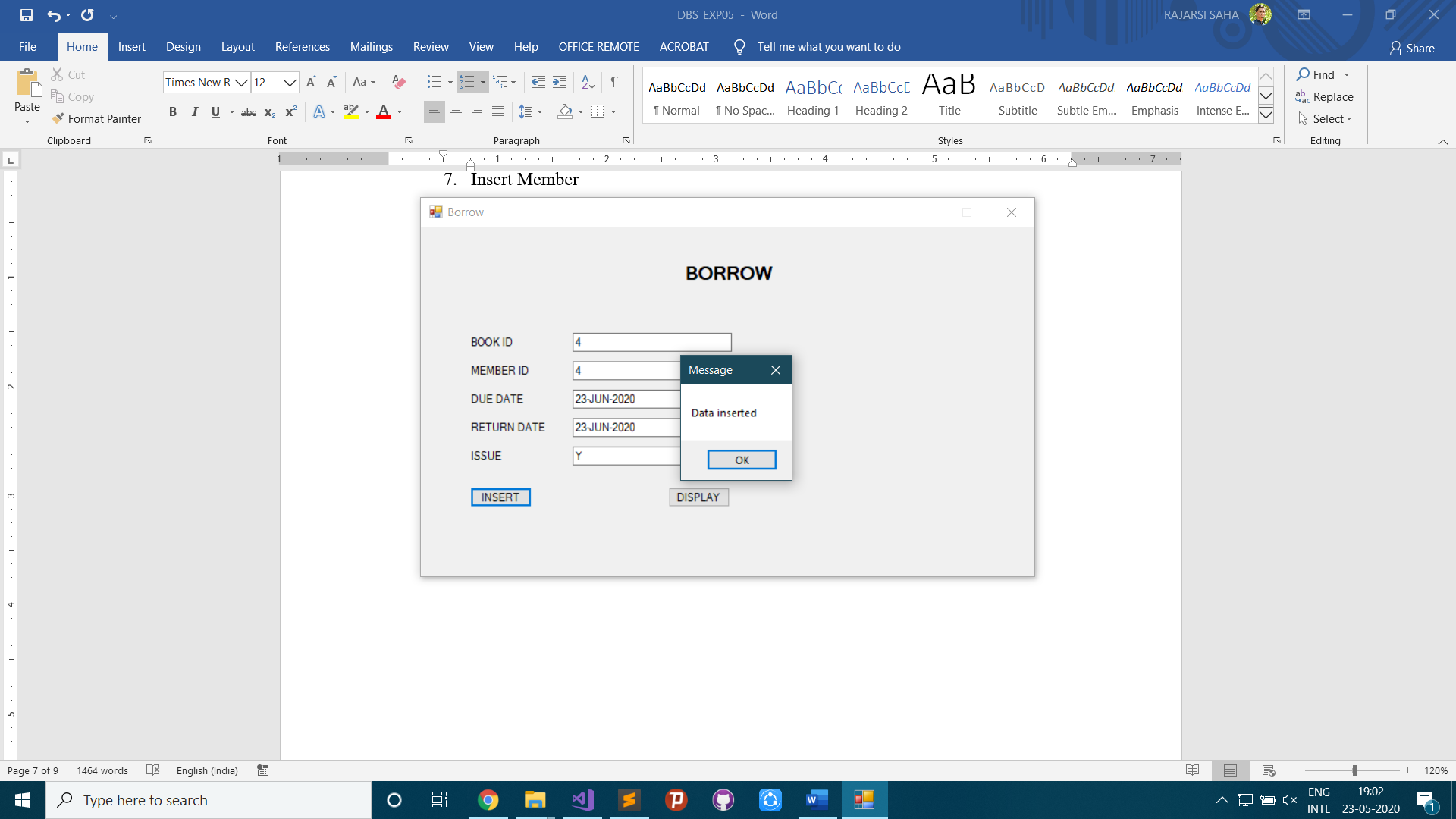
1. Insert Publisher



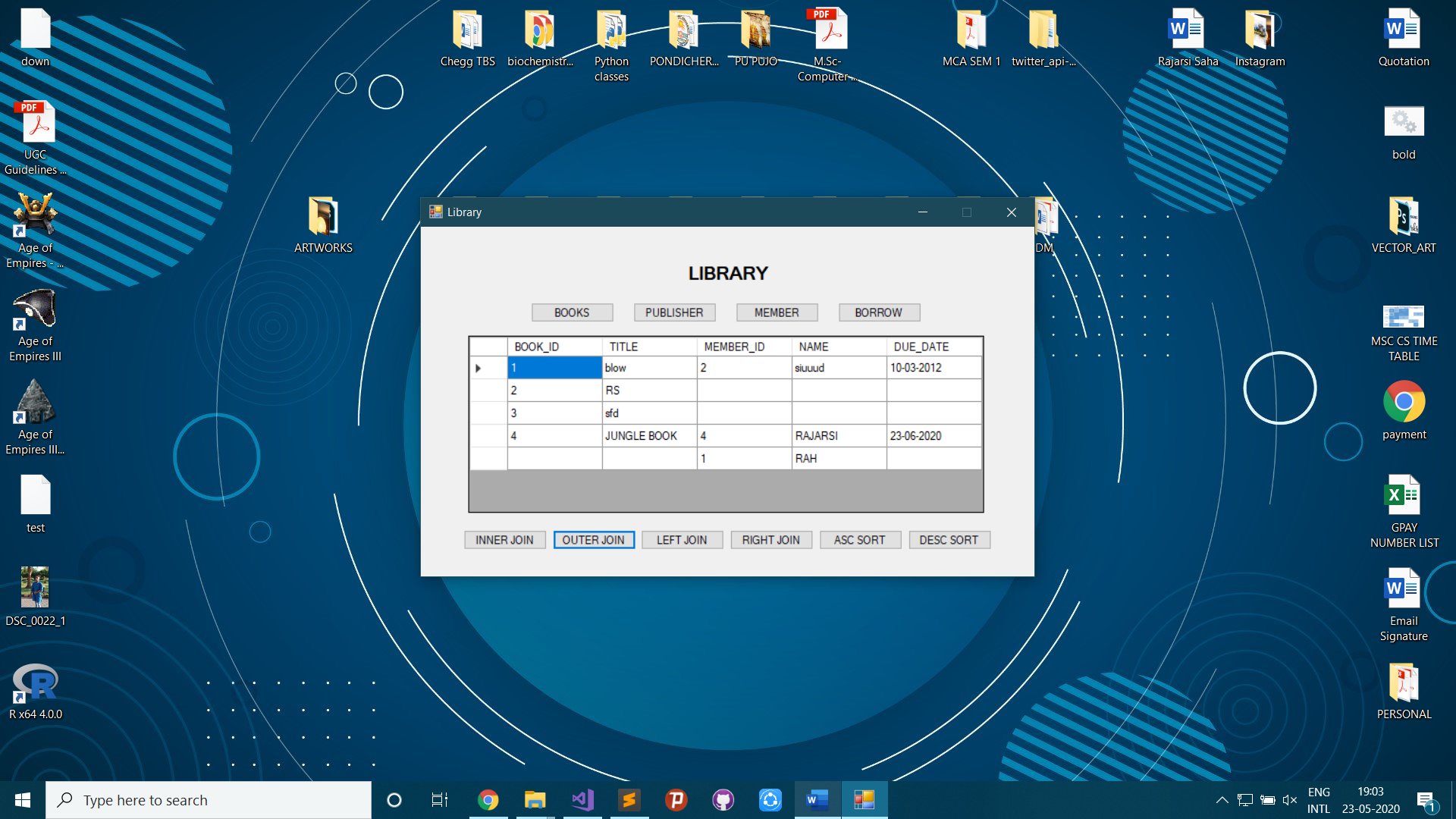
1. Insert Member



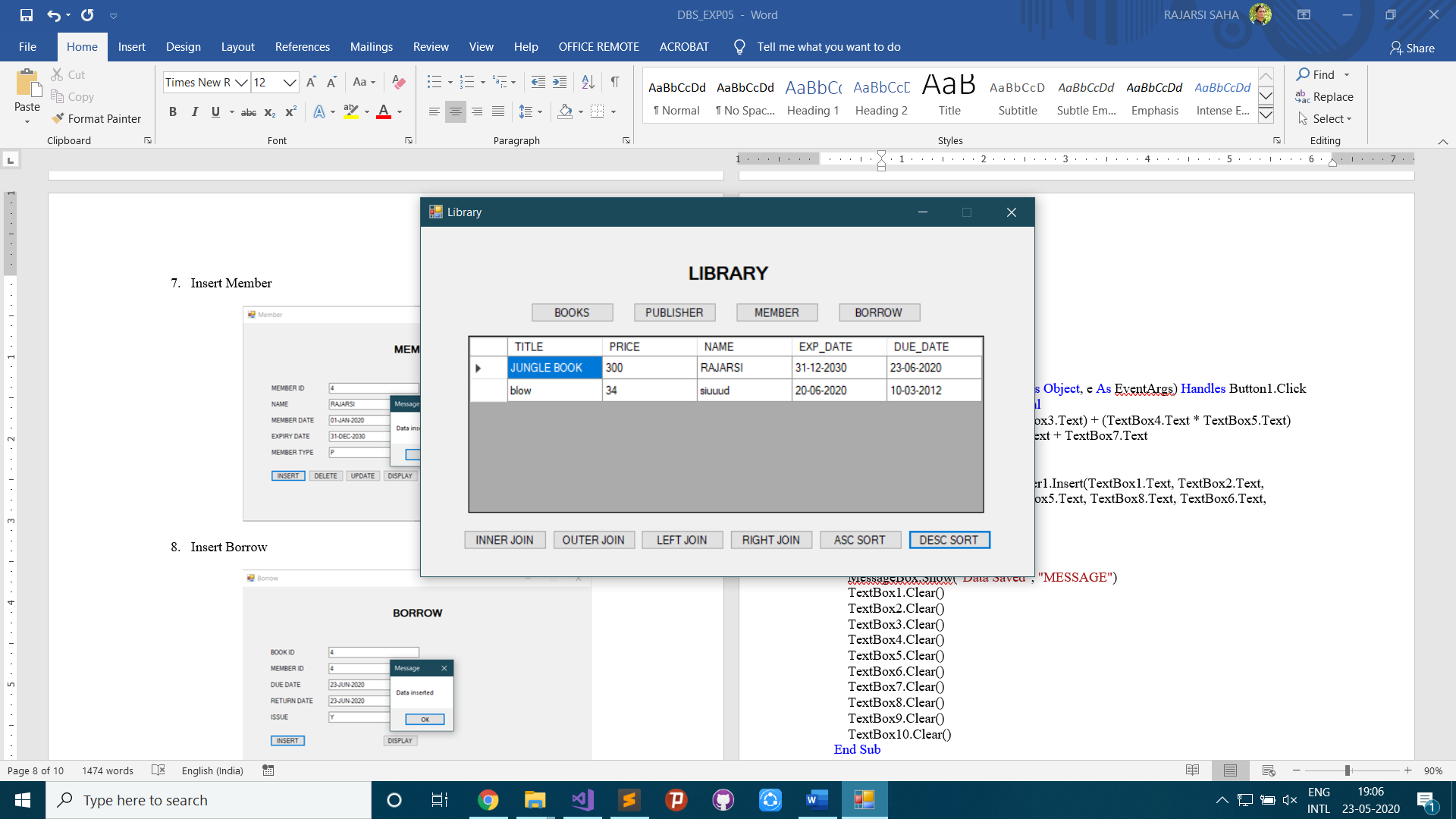
1. Insert Borrow



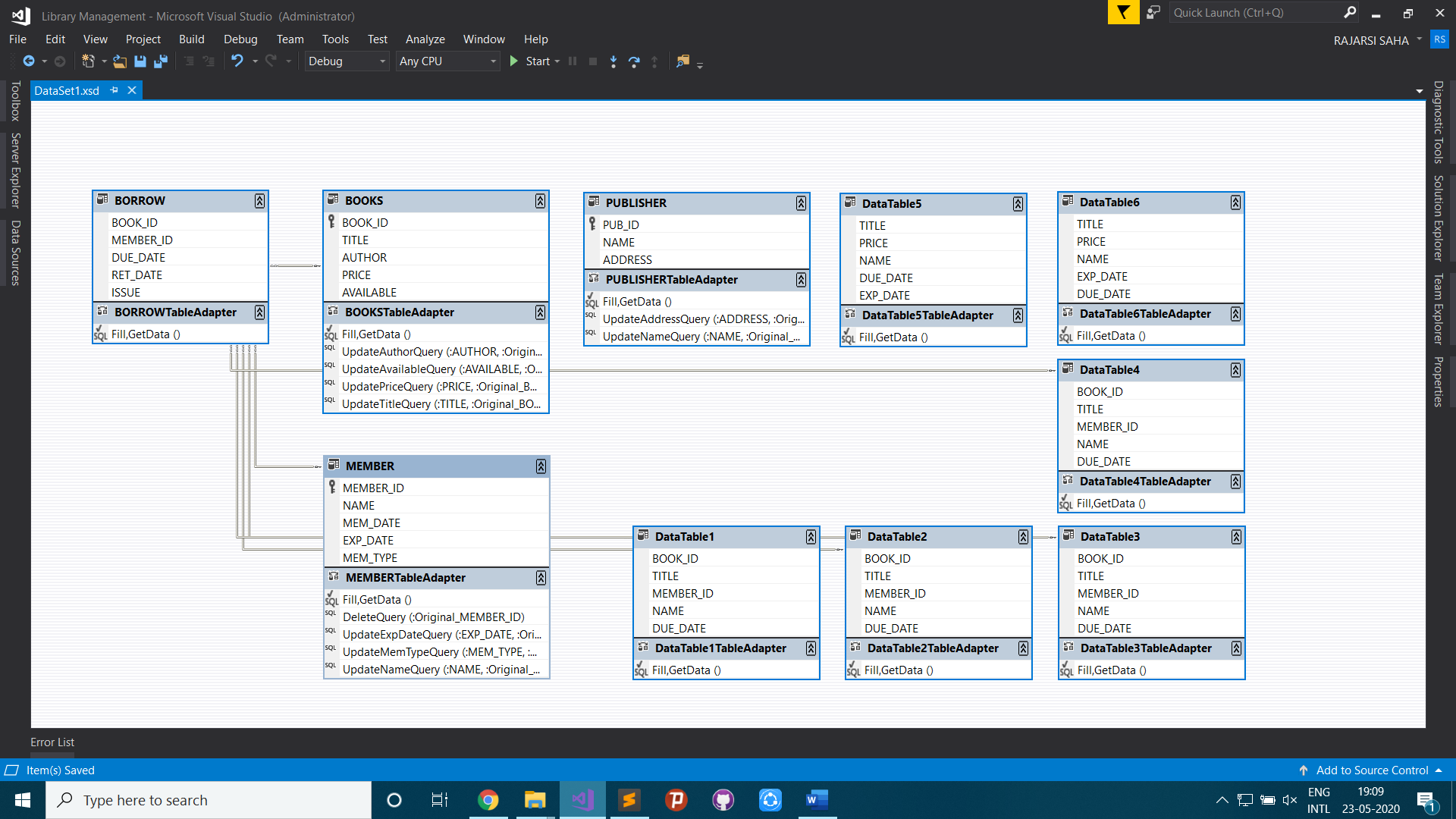
1. Use Join (Full Outer)



1. Use Sort (Desceding)



**DATSET IN VISUAL STUDIO**



**E-R DIAGRAM:**

An **entity–relationship model** (or **ER model**) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).

In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model, that defines a data or information structure which can be implemented in a database, typically a relational database.

Entity–relationship modeling was developed for database and design by Peter Chen and published in a 1976 paper.[1] However, variants of the idea existed previously.[2] Some ER models show super and subtype entities connected by generalization-specialization relationships,[3] and an ER model can be used also in the specification of domain-specific ontologies.

**VISUAL BASIC CODE:**

Books Form:

Public Class Form1

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'TODO: This line of code loads data into the 'DataSet11.BOOKS' table. You can move, or remove it, as needed.

Me.BOOKSTableAdapter.Fill(Me.DataSet11.BOOKS)

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

DataGridView1.Visible = False

Try

BOOKSTableAdapter.Insert(TextBox1.Text, TextBox2.Text, TextBox3.Text, TextBox4.Text, TextBox5.Text)

MessageBox.Show("Data inserted", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

TextBox4.Clear()

TextBox5.Clear()

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

DataGridView1.Visible = False

If TextBox1.Text = "" Then

Exit Sub

End If

If TextBox3.Text = "" And TextBox4.Text - "" And TextBox5.Text = "" Then

Try

BOOKSTableAdapter.UpdateTitleQuery(TextBox1.Text, TextBox2.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

ElseIf TextBox2.Text = "" And TextBox4.Text - "" And TextBox5.Text = "" Then

Try

BOOKSTableAdapter.UpdateTitleQuery(TextBox1.Text, TextBox3.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

ElseIf TextBox2.Text = "" And TextBox3.Text - "" And TextBox5.Text = "" Then

Try

BOOKSTableAdapter.UpdateTitleQuery(TextBox1.Text, TextBox4.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

ElseIf TextBox2.Text = "" And TextBox3.Text - "" And TextBox4.Text = "" Then

Try

BOOKSTableAdapter.UpdateTitleQuery(TextBox1.Text, TextBox5.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

End If

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

TextBox4.Clear()

TextBox5.Clear()

End Sub

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

BOOKSTableAdapter.Fill(DataSet11.BOOKS)

DataGridView1.DataSource = BOOKSBindingSource

DataGridView1.Visible = True

End Sub

Private Sub Form1\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

Me.Hide()

Form5.Show()

End Sub

End Class

Publiher Form:

Public Class Form2

Private Sub Form2\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'TODO: This line of code loads data into the 'DataSet11.PUBLISHER' table. You can move, or remove it, as needed.

Me.PUBLISHERTableAdapter.Fill(Me.DataSet11.PUBLISHER)

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

DataGridView1.Visible = False

Try

PUBLISHERTableAdapter.Insert(TextBox1.Text, TextBox2.Text, TextBox3.Text)

MessageBox.Show("Data inserted", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

DataGridView1.Visible = False

If TextBox1.Text = "" Then

Exit Sub

End If

If TextBox3.Text = "" Then

Try

PUBLISHERTableAdapter.UpdateNameQuery(TextBox1.Text, TextBox2.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

ElseIf TextBox2.Text = "" Then

Try

PUBLISHERTableAdapter.UpdateAddressQuery(TextBox1.Text, TextBox3.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

End If

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

End Sub

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

PUBLISHERTableAdapter.Fill(DataSet11.PUBLISHER)

DataGridView1.DataSource = PUBLISHERBindingSource

DataGridView1.Visible = True

End Sub

Private Sub Form2\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

Me.Hide()

Form5.Show()

End Sub

End Class

Member Form:

Public Class Form3

Private Sub Form3\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'TODO: This line of code loads data into the 'DataSet11.MEMBER' table. You can move, or remove it, as needed.

Me.MEMBERTableAdapter.Fill(Me.DataSet11.MEMBER)

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

DataGridView1.Visible = False

Try

MEMBERTableAdapter.Insert(TextBox1.Text, TextBox2.Text, TextBox3.Text, TextBox4.Text, TextBox5.Text)

MessageBox.Show("Data inserted", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

TextBox4.Clear()

TextBox5.Clear()

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

DataGridView1.Visible = False

Try

MEMBERTableAdapter.DeleteQuery(TextBox1.Text)

MessageBox.Show("Data deleted", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

TextBox4.Clear()

TextBox5.Clear()

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

DataGridView1.Visible = False

If TextBox1.Text = "" Then

Exit Sub

End If

If TextBox3.Text = "" And TextBox4.Text - "" And TextBox5.Text = "" Then

Try

MEMBERTableAdapter.UpdateNameQuery(TextBox1.Text, TextBox2.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

ElseIf TextBox2.Text = "" And TextBox4.Text - "" And TextBox5.Text = "" Then

MessageBox.Show("Cannot edit Member Date!", "Message")

ElseIf TextBox2.Text = "" And TextBox3.Text - "" And TextBox5.Text = "" Then

Try

MEMBERTableAdapter.UpdateExpDateQuery(TextBox1.Text, TextBox4.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

ElseIf TextBox2.Text = "" And TextBox3.Text - "" And TextBox4.Text = "" Then

Try

MEMBERTableAdapter.UpdateMemTypeQuery(TextBox1.Text, TextBox5.Text)

MessageBox.Show("Data updated", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

End If

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

TextBox4.Clear()

TextBox5.Clear()

End Sub

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

MEMBERTableAdapter.Fill(DataSet11.MEMBER)

DataGridView1.DataSource = MEMBERBindingSource

DataGridView1.Visible = True

End Sub

Private Sub Form3\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

Me.Hide()

Form5.Show()

End Sub

End Class

Borrow Form:

Public Class Form4

Private Sub Form4\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'TODO: This line of code loads data into the 'DataSet11.BORROW' table. You can move, or remove it, as needed.

Me.BORROWTableAdapter.Fill(Me.DataSet11.BORROW)

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

DataGridView1.Visible = False

If TextBox1.Text = "" Then

Exit Sub

End If

Try

BORROWTableAdapter.Insert(TextBox1.Text, TextBox2.Text, TextBox3.Text, TextBox4.Text, TextBox5.Text)

MessageBox.Show("Data inserted", "Message")

Catch ex As Exception

MessageBox.Show(ex.Message, "Error")

End Try

TextBox1.Clear()

TextBox2.Clear()

TextBox3.Clear()

End Sub

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

BORROWTableAdapter.Fill(DataSet11.BORROW)

DataGridView1.DataSource = BORROWBindingSource

DataGridView1.Visible = True

End Sub

Private Sub Form4\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

Me.Hide()

Form5.Show()

End Sub

End Class

**FEW RELATED DEFINITIONS:**

**Database:** A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modelling techniques.

**Database Management System (DBMS):**

**The database management system (DBMS)** is the software that interacts with end users, applications, and the database itself to capture and analyse the data. The DBMS software additionally encompasses the core facilities provided to administer the database.

**Oracle 12c database**

The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

**Windows Forms:**

**Windows Forms (WinForms)** is a graphical (GUI) class library included as a part of Microsoft .NET Framework or Mono Framework, providing a platform to write rich client applications for desktop, laptop, and tablet PCs. While it is seen as a replacement for the earlier and more complex C++ based Microsoft Foundation Class Library, it does not offer a comparable paradigm and only acts as a platform for the user interface tier in a multi-tier solution.

**Microsoft .NET Windows form**

At the Microsoft Connect event on December 4, 2018, Microsoft announced releasing Windows Forms as an open source project on GitHub. It is released under the MIT License. With this release, Windows Forms has become available for projects targeting the .NET Core framework. However, the framework is still available only on the Windows platform, and Mono's incomplete implementation of WinForms remains the only cross-platform implementation.