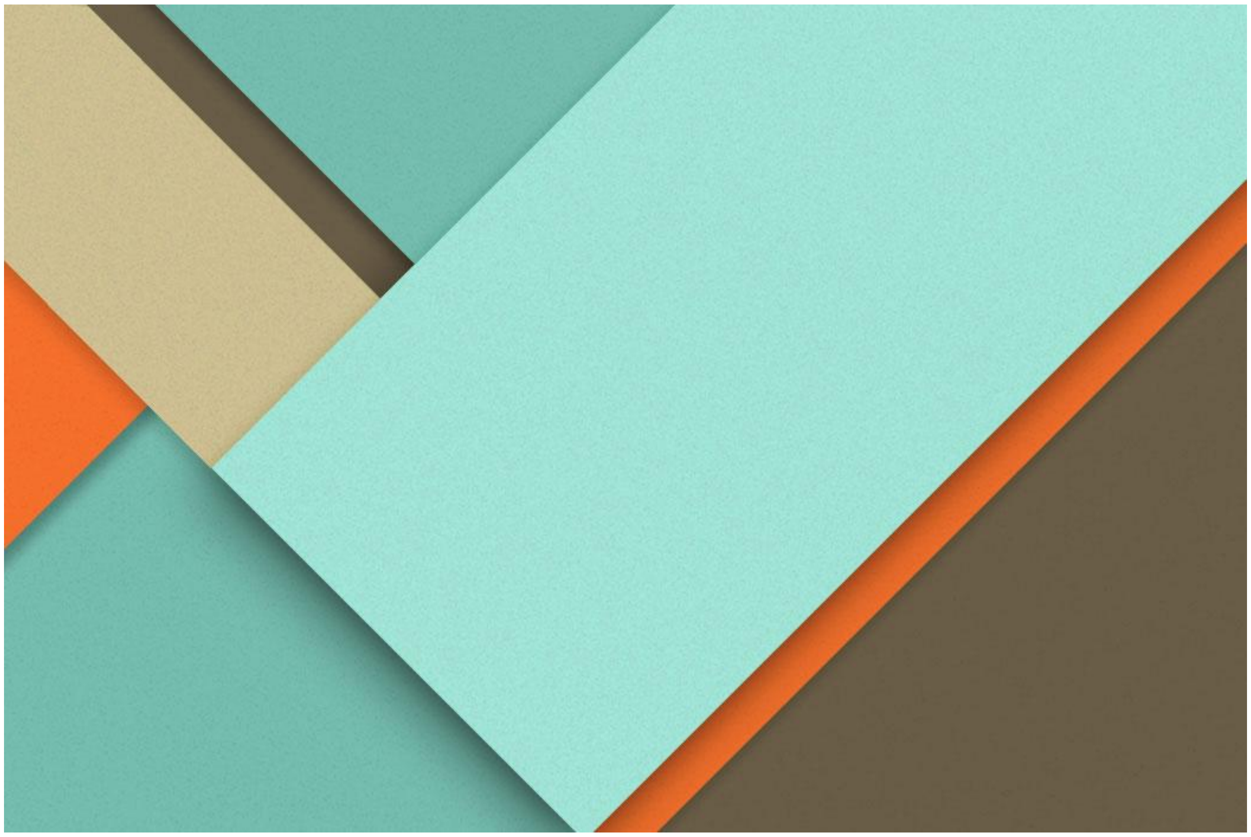


OneNote.

10.25.2022

The domain of Your Project

Note-taking and editing.

Motivation for the Project

The motivation for choosing this particular project is to apply and copy the masterly App OneNote of Microsoft. I personally am fascinated by the discreteness of the OneNote, Its interface is so simple that a kid might get used to it and its features are so vast that i yet didn't know about some. And furthermore I like the idea of how the notes are all interconnected through your microsoft account, you can use it on any device having a single account and can read and write from wherever you want.

Problems that will be Solved by this Project

By this project I will try to provide a single place for keeping all of your notes, research, plans, and information — everything you need to remember and manage in your life at home, at work, or at school. In OneNote, notebooks never run out of paper.

Database Design

No of Roles are 4

User	Role	Description
End User (Student, Person)	Uses the app.	Can note down important dates and messages. Can make his education notes. Can share his notes.
Database Administrator	Manages, Monitor	Review users and can delete them.
Database Analyst	Maintain, Accesses	Maintain data storage, Accesses database design.
Database Developer	Create, Develop	Ensure Performance, Security and Integrity of data.

Tables: No of Tables are 7 namely, User, Notebook, Section, Page, Note, NoteLayout, Category.

Database Overview

Table Name	Column Name	Datatype	Constraints

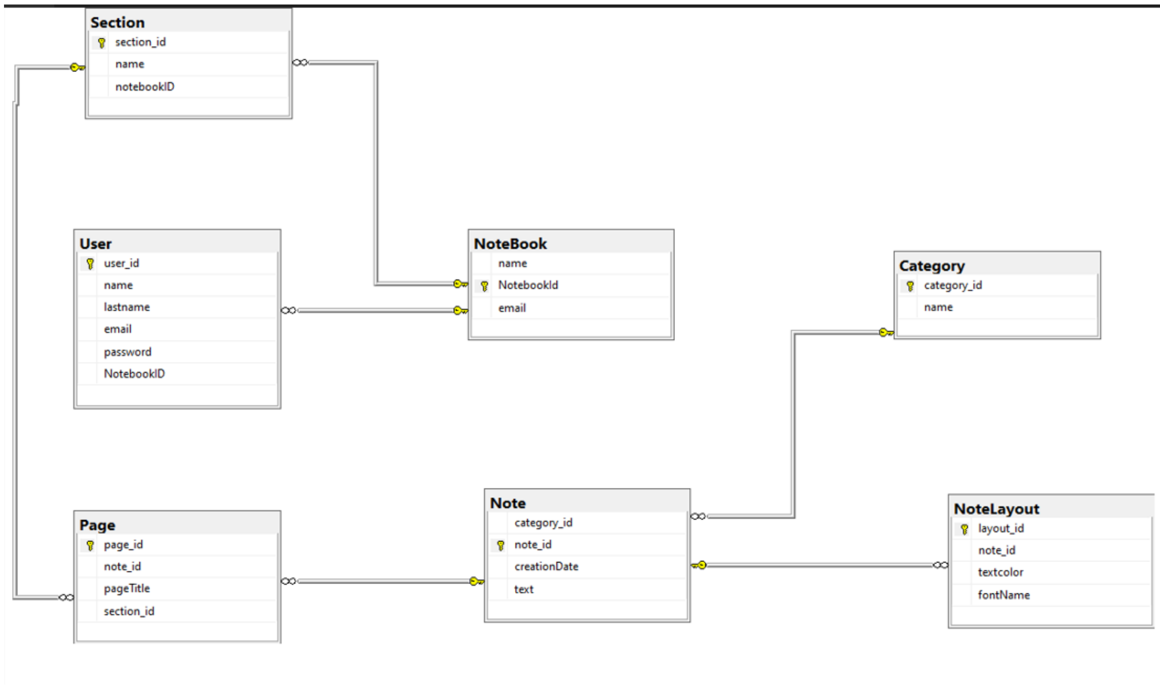
User	User_id
	firstname
	Lastname
	Email
	password.
	notebook_id
		Int
		varchar(45)
		varchar(45)
		varchar(max)
		varchar(max)
		int
			Primary Key
			
			Unique Key
			
			FK

Notebook	Name
	Notebook_id
	email
		varchar(45)
		Int
		varchar(max)
			Not Null
			PK
			UQ

Section	Section_id Name notebook_id	Int varchar(45) int	PK Not null FK
Page	Page_id Note_id Pagetitle Section_id	Int Int varchar(45) int	PK FK Not Null FK
Note	Category_id Note_id Creationdate	Int Int DateTime	FK PK

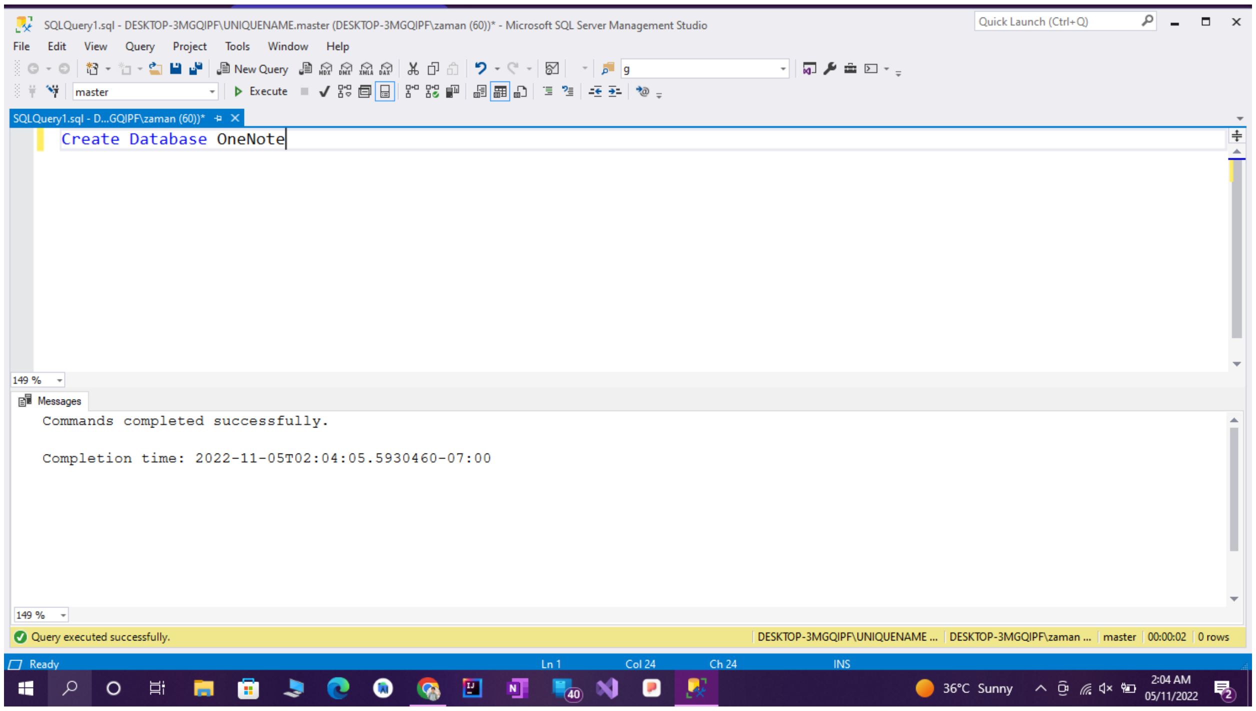
	Text	varchar(max)	
NoteLayout	layout_id Note_id Textcolor fontname	Int Int varchar(15) varchar(15)	PK FK
Category	Category_id Name	Int varchar(45)	PK

ER-Diagram



Database Creation

1. Create Database OneNote:

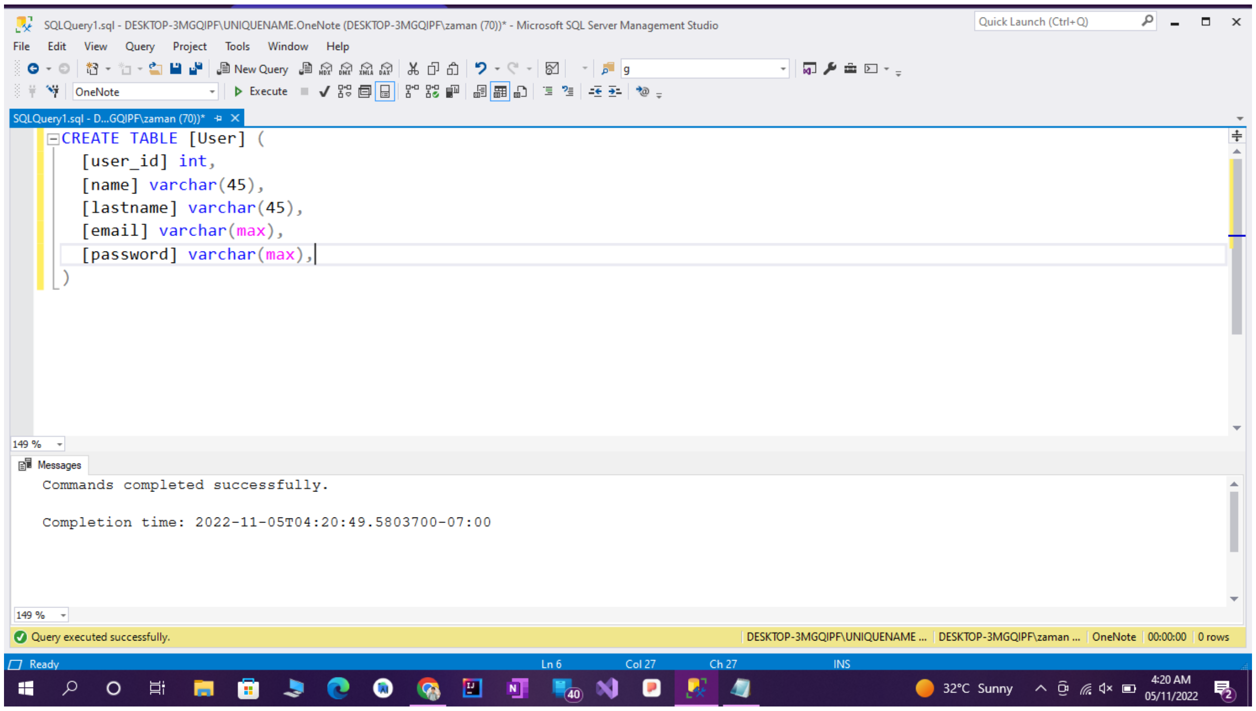


```
Create Database OneNote
```

```
Commands completed successfully.

Completion time: 2022-11-05T02:04:05.5930460-07:00
```

SQLQuery1.sql - DESKTOP-3MGQIPF\UNIQUENAME.master (DESKTOP-3MGQIPF\zaman (60))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
master
Execute
Messages
149 %
Query executed successfully.
DESKTOP-3MGQIPF\UNIQUENAME ... DESKTOP-3MGQIPF\zaman ... master 00:00:02 0 rows
Ready Ln 1 Col 24 Ch 24 INS
36°C Sunny 2:04 AM 05/11/2022

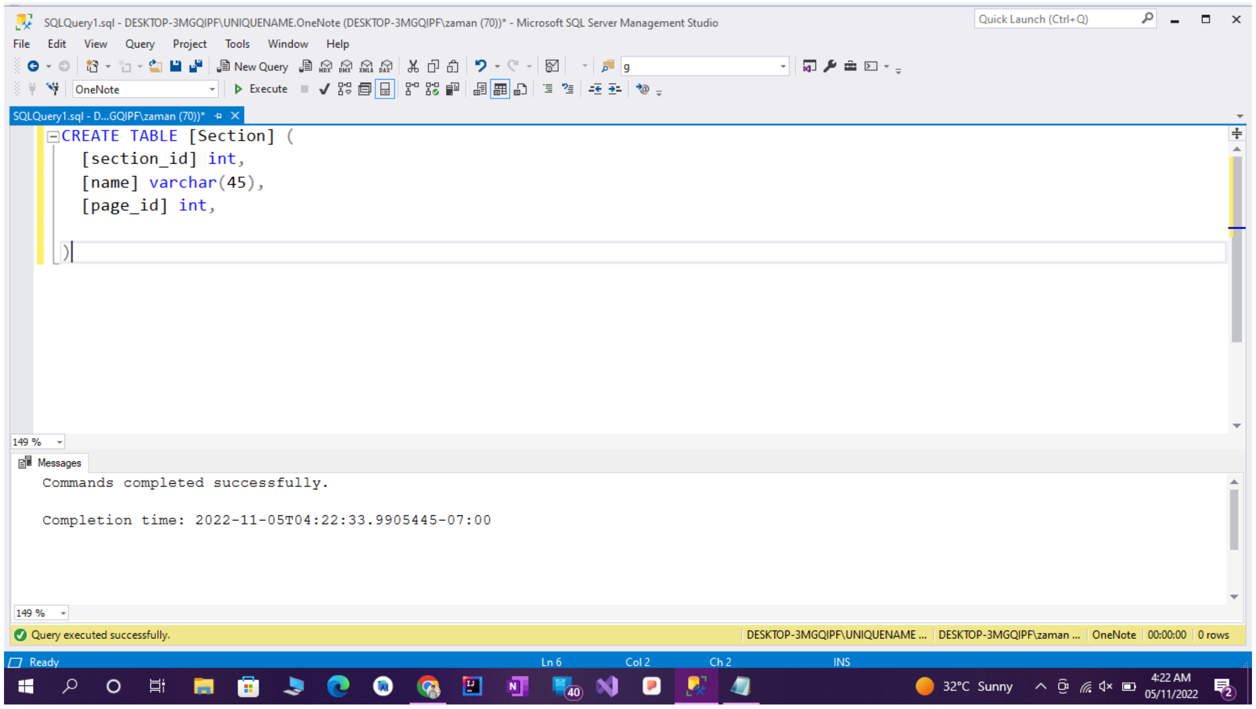
2. Table User:



```sql
CREATE TABLE [User] (
    [user_id] int,
    [name] varchar(45),
    [lastname] varchar(45),
    [email] varchar(max),
    [password] varchar(max),
)
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:20:49.5803700-07:00
```

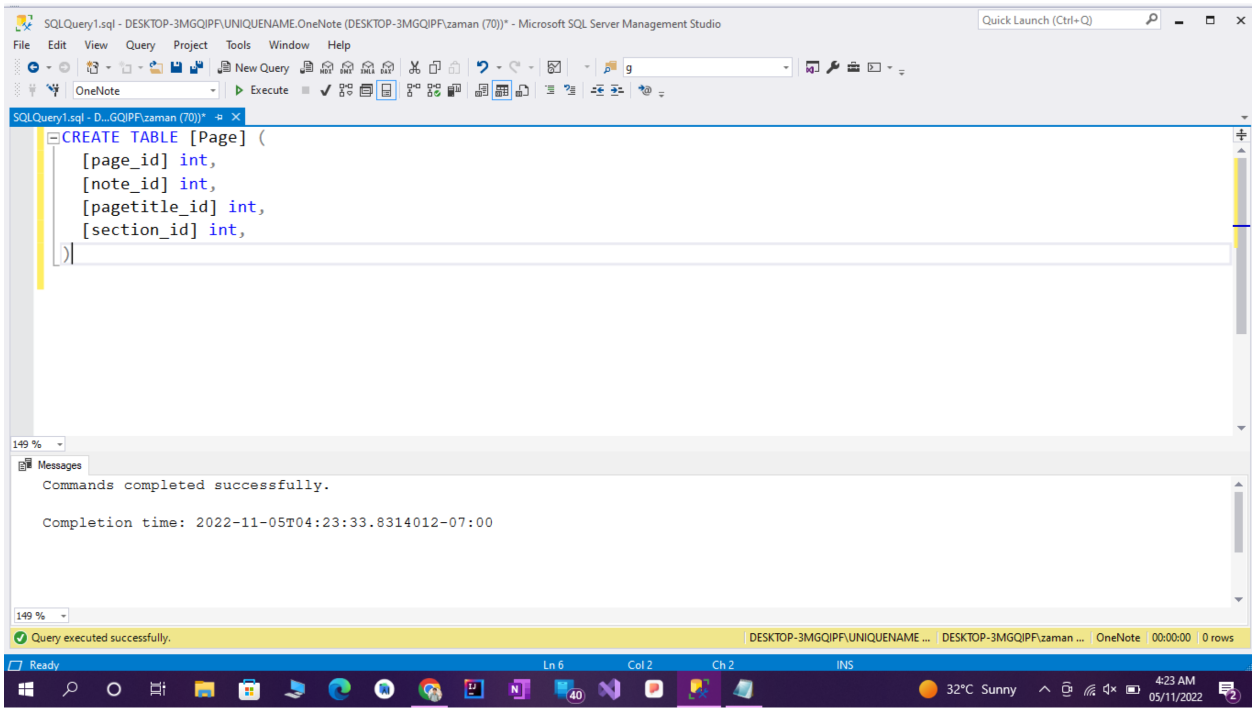
3. Table Section:



```sql
CREATE TABLE [Section] (
    [section_id] int,
    [name] varchar(45),
    [page_id] int,

)
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:22:33.9905445-07:00
```

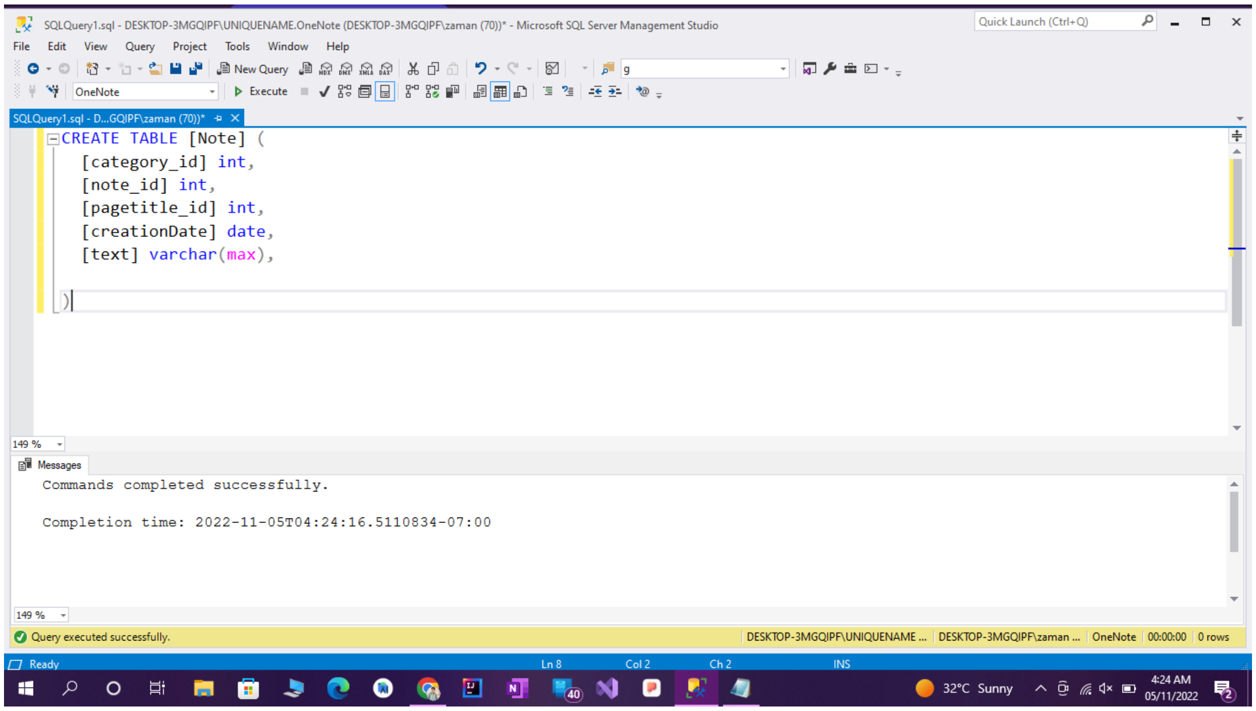
4. Table Page:



```sql
CREATE TABLE [Page] (
  [page_id] int,
  [note_id] int,
  [pagetitle_id] int,
  [section_id] int,
)
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:23:33.8314012-07:00
```

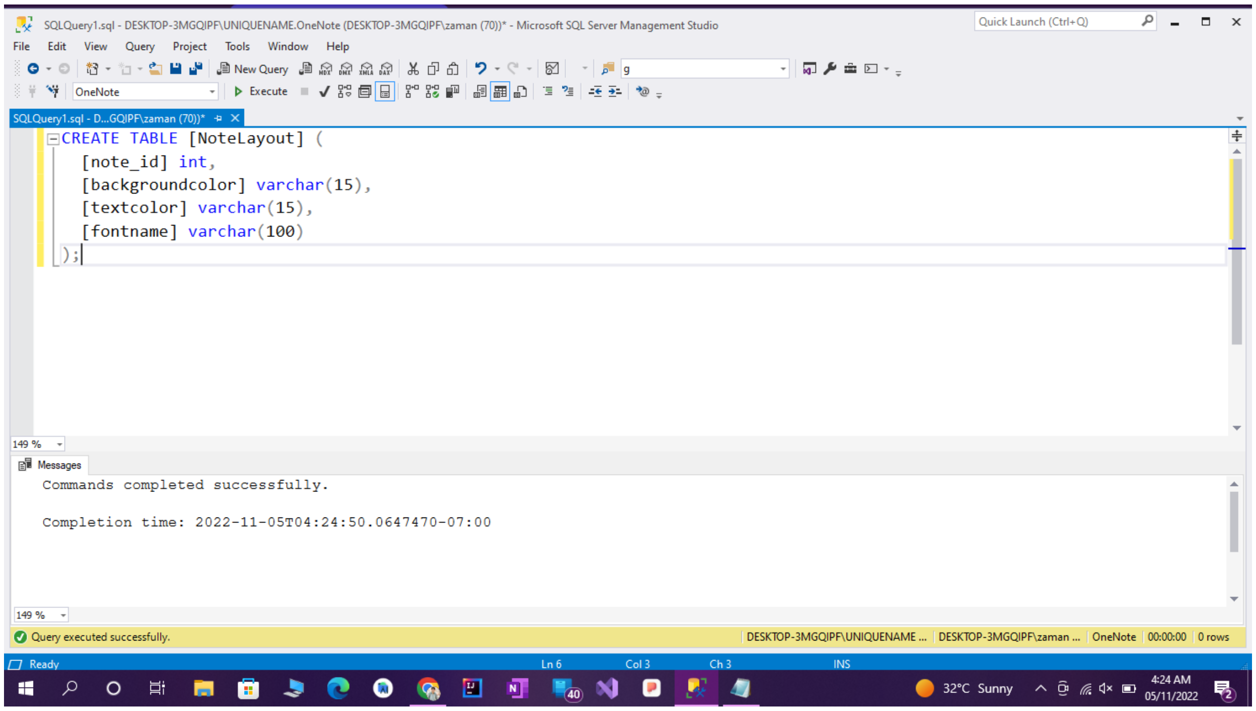
5. Table Note:



```sql
CREATE TABLE [Note] (
  [category_id] int,
  [note_id] int,
  [pagetitle_id] int,
  [creationDate] date,
  [text] varchar(max),

)
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:24:16.5110834-07:00
```

6. Table NoteLayout:



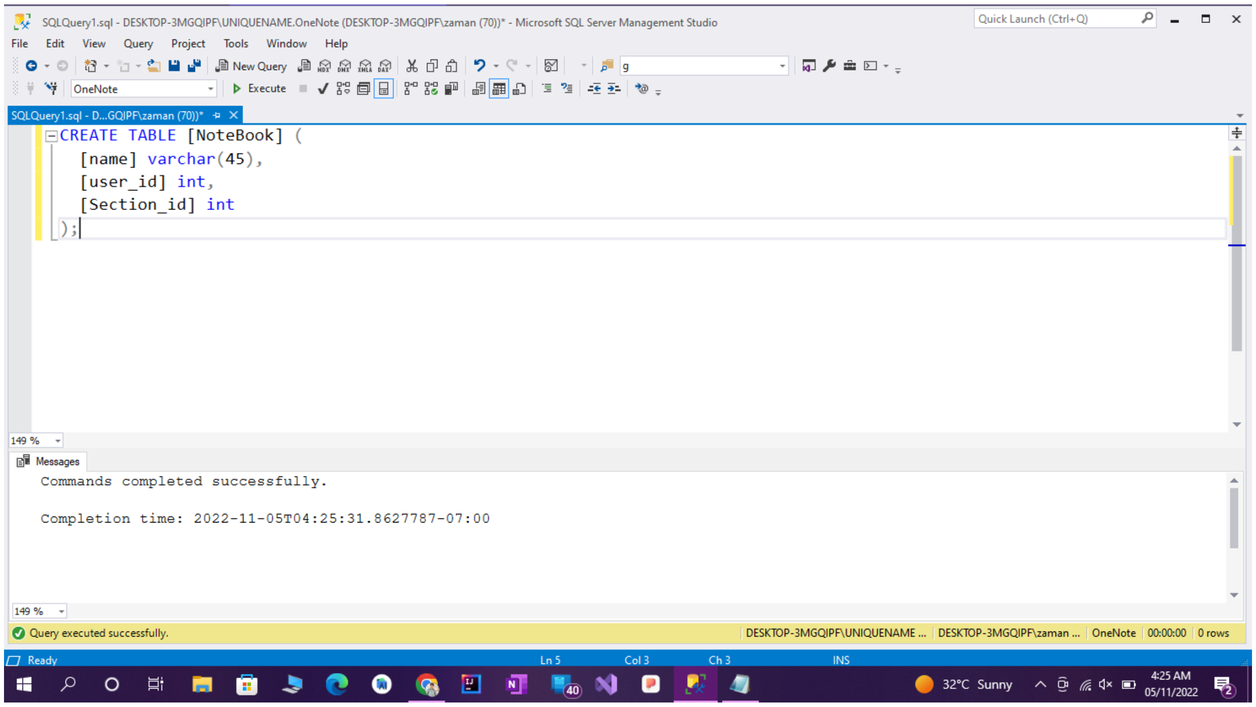
SQLQuery1.sql - DESKTOP-3MGQIPF\UNIQUENAME.OneNote (DESKTOP-3MGQIPF\zaman (70))* - Microsoft SQL Server Management Studio

```sql
CREATE TABLE [NoteLayout] (
  [note_id] int,
  [backgroundcolor] varchar(15),
  [textcolor] varchar(15),
  [fontname] varchar(100)
);
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:24:50.0647470-07:00
```

Query executed successfully.

7. Table NoteBook



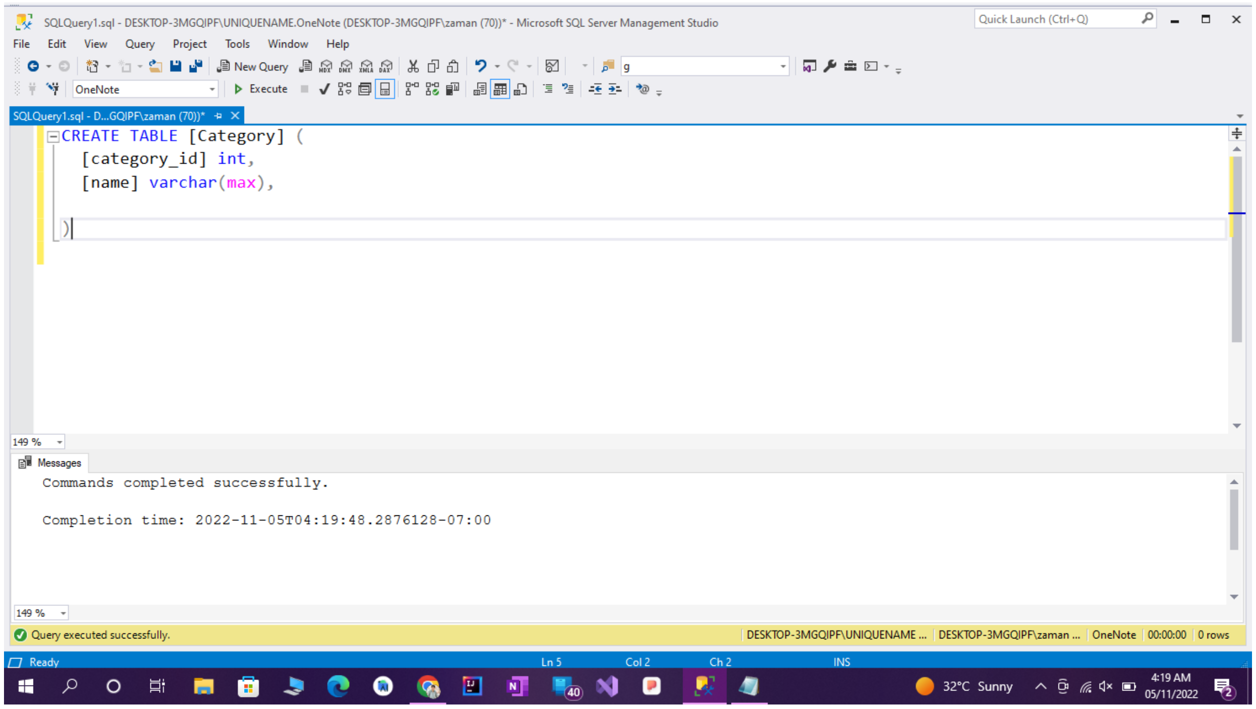
SQLQuery1.sql - DESKTOP-3MGQIPF\UNIQUENAME.OneNote (DESKTOP-3MGQIPF\zaman (70))* - Microsoft SQL Server Management Studio

```sql
CREATE TABLE [NoteBook] (
  [name] varchar(45),
  [user_id] int,
  [Section_id] int
);
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:25:31.8627787-07:00
```

Query executed successfully.

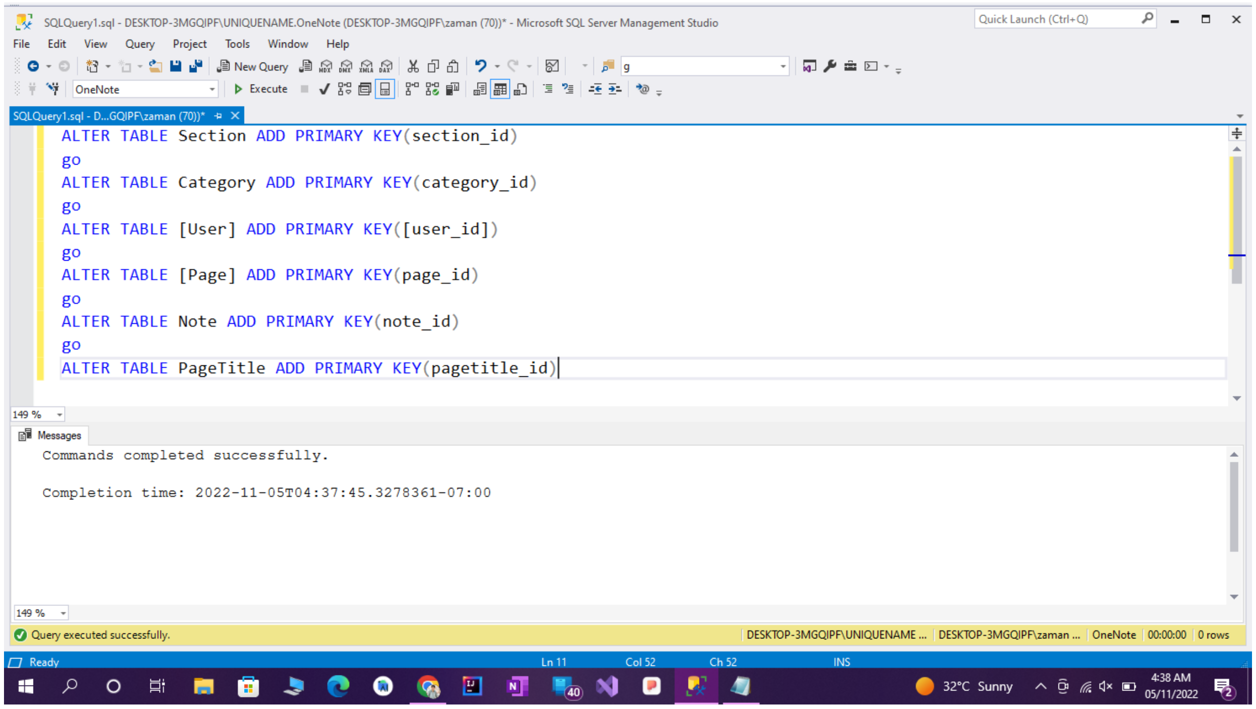
8. Table Category:



```
CREATE TABLE [Category] (
  [category_id] int,
  [name] varchar(max),

)
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:19:48.2876128-07:00
```

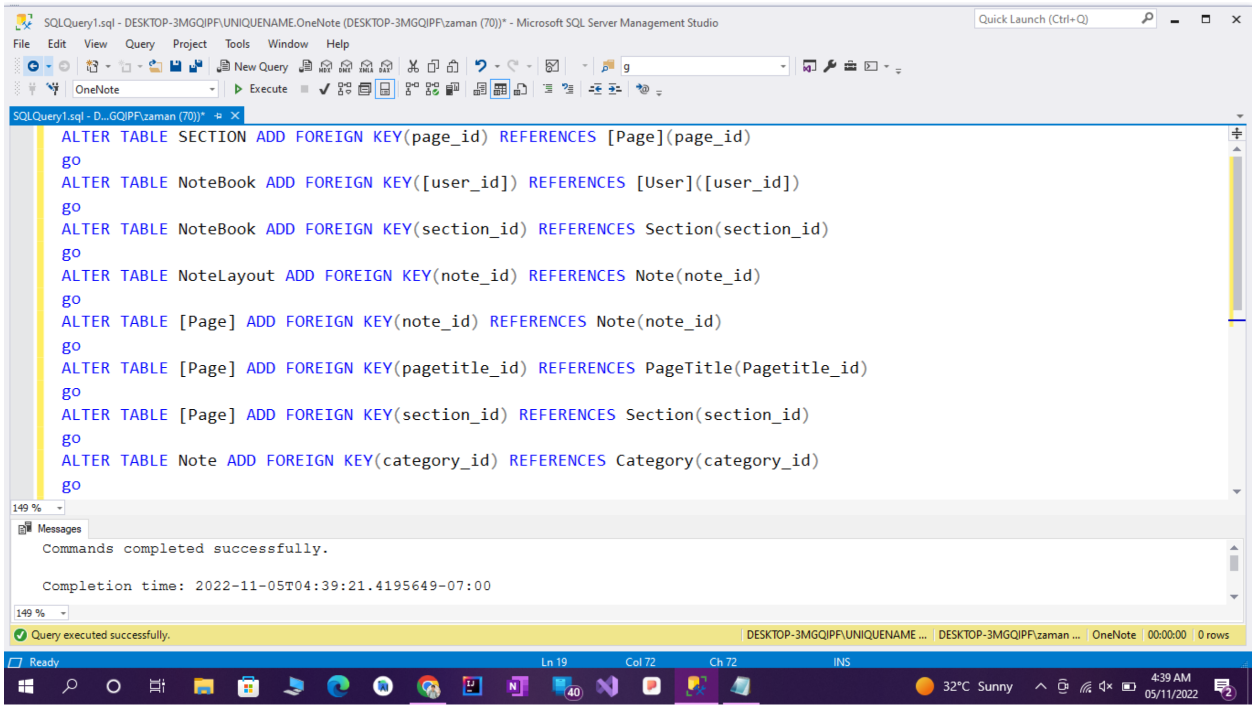
9. Adding Primary Keys:



```
ALTER TABLE Section ADD PRIMARY KEY(section_id)
go
ALTER TABLE Category ADD PRIMARY KEY(category_id)
go
ALTER TABLE [User] ADD PRIMARY KEY([user_id])
go
ALTER TABLE [Page] ADD PRIMARY KEY(page_id)
go
ALTER TABLE Note ADD PRIMARY KEY(note_id)
go
ALTER TABLE PageTitle ADD PRIMARY KEY(pagetitle_id)
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:37:45.3278361-07:00
```

10. Adding Foreign Keys:



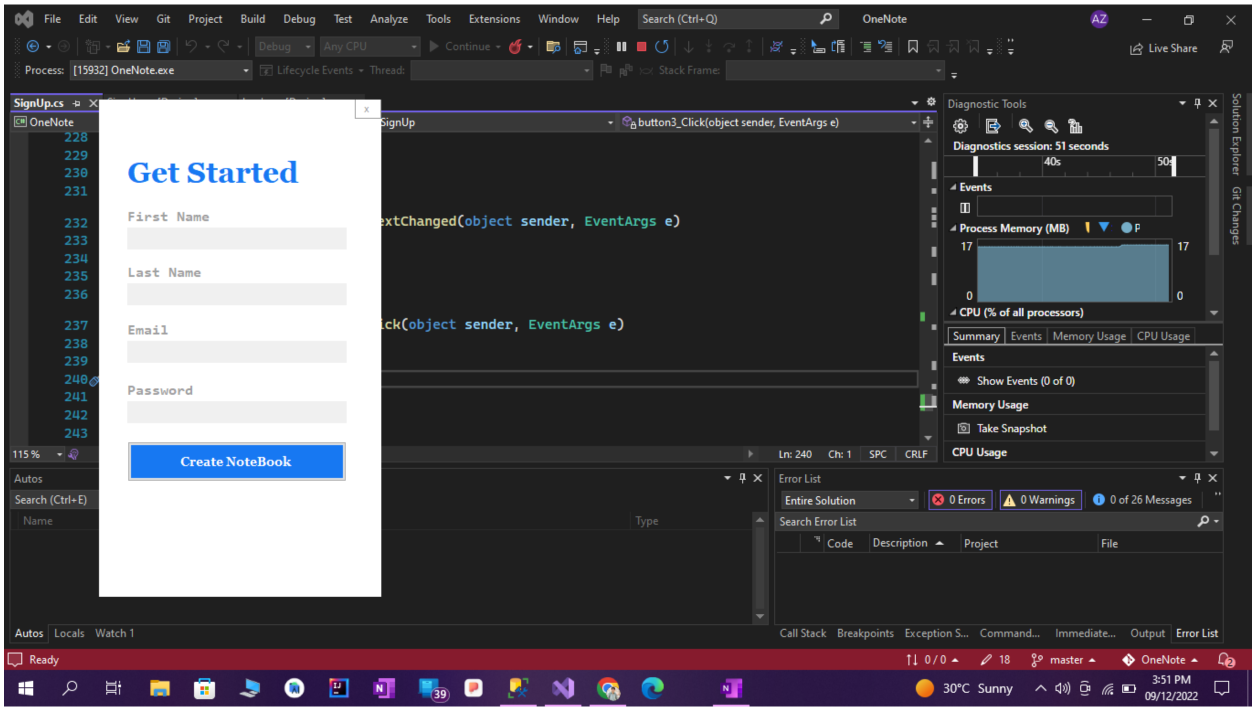
```
ALTER TABLE SECTION ADD FOREIGN KEY(page_id) REFERENCES [Page](page_id)
go
ALTER TABLE NoteBook ADD FOREIGN KEY([user_id]) REFERENCES [User]([user_id])
go
ALTER TABLE NoteBook ADD FOREIGN KEY(section_id) REFERENCES Section(section_id)
go
ALTER TABLE NoteLayout ADD FOREIGN KEY(note_id) REFERENCES Note(note_id)
go
ALTER TABLE [Page] ADD FOREIGN KEY(note_id) REFERENCES Note(note_id)
go
ALTER TABLE [Page] ADD FOREIGN KEY(pagetitle_id) REFERENCES PageTitle(Pagetitle_id)
go
ALTER TABLE [Page] ADD FOREIGN KEY(section_id) REFERENCES Section(section_id)
go
ALTER TABLE Note ADD FOREIGN KEY(category_id) REFERENCES Category(category_id)
go
```

```
Commands completed successfully.

Completion time: 2022-11-05T04:39:21.4195649-07:00
```

Query executed successfully.

User Interface

1. Login/SignUp:



Get Started

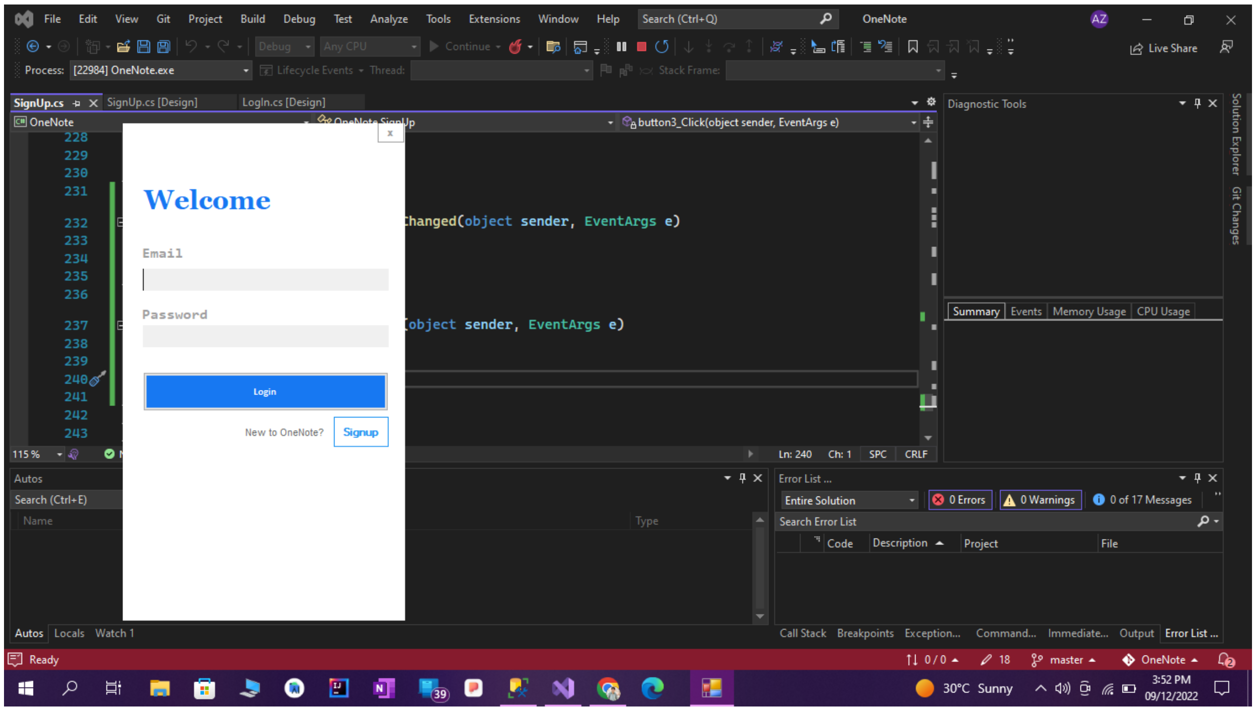
First Name

Last Name

Email

Password

Create NoteBook



Welcome

Email

Password

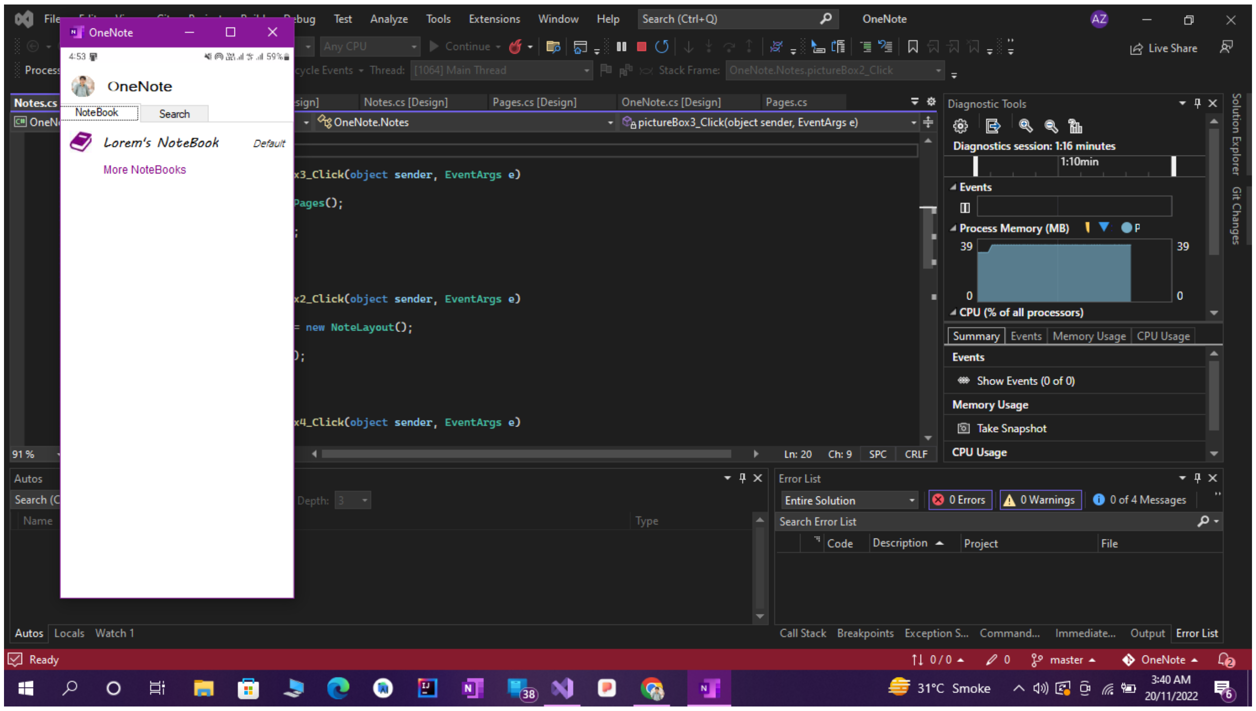
Login

New to OneNote? Signup

```
hanged(object sender, EventArgs e)

(object sender, EventArgs e)
```

Error List: Entire Solution — 0 Errors — 0 Warnings — 0 of 17 Messages

2. OneNote App Starting Page:(After Clicking Login)



OneNote

NoteBook Search

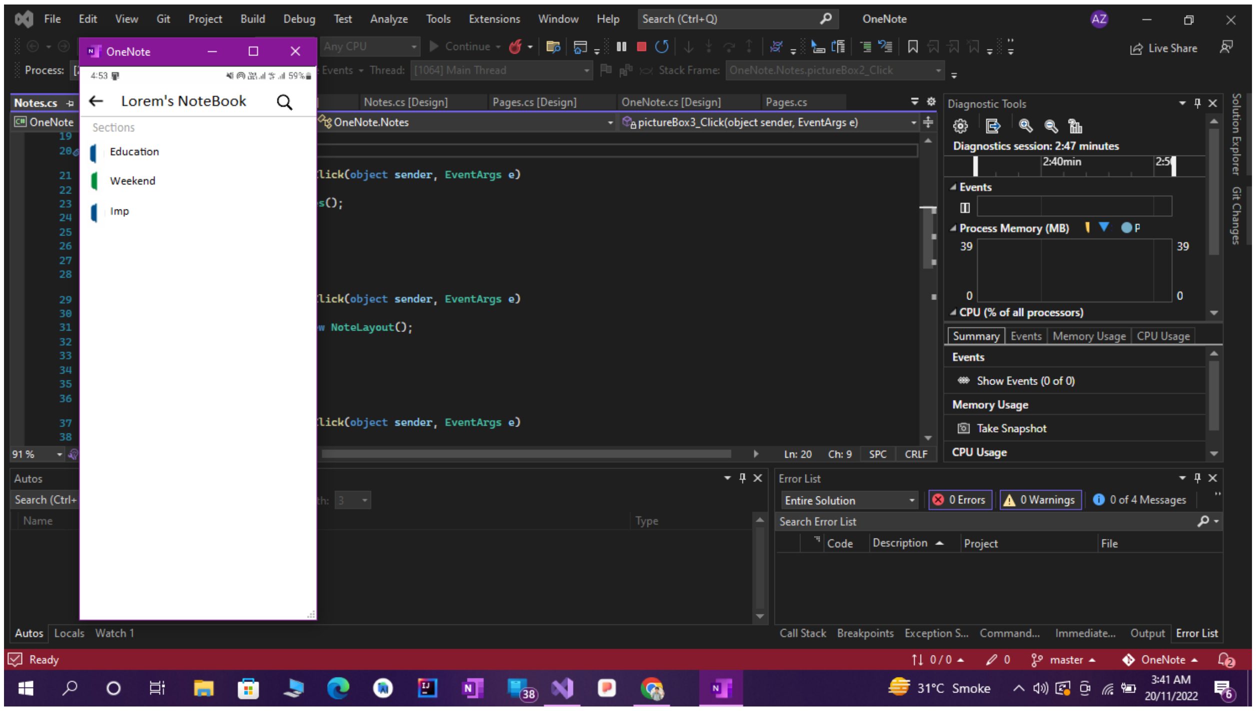
Lorem's NoteBook Default

More NoteBooks

```
x3_Click(object sender, EventArgs e)

Pages();

;

x2_Click(object sender, EventArgs e)

= new NoteLayout();

);

x4_Click(object sender, EventArgs e)
```

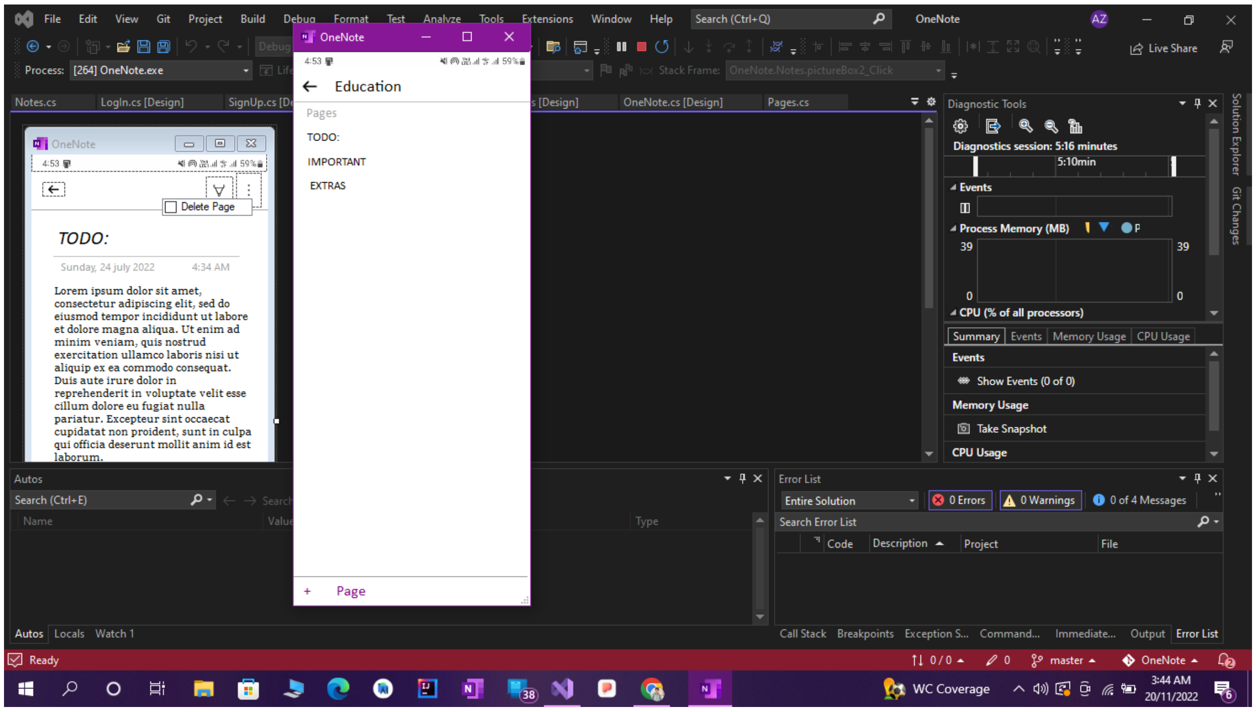
Error List: Entire Solution — 0 Errors — 0 Warnings — 0 of 4 Messages

3. After Clicking Lorem's NoteBook: (These Are Sections)



```
lick(object sender, EventArgs e)
s();
lick(object sender, EventArgs e)
NoteLayout();
lick(object sender, EventArgs e)
```

4. After Clicking Education(Section): (Now these are the pages which are inside the Sections)

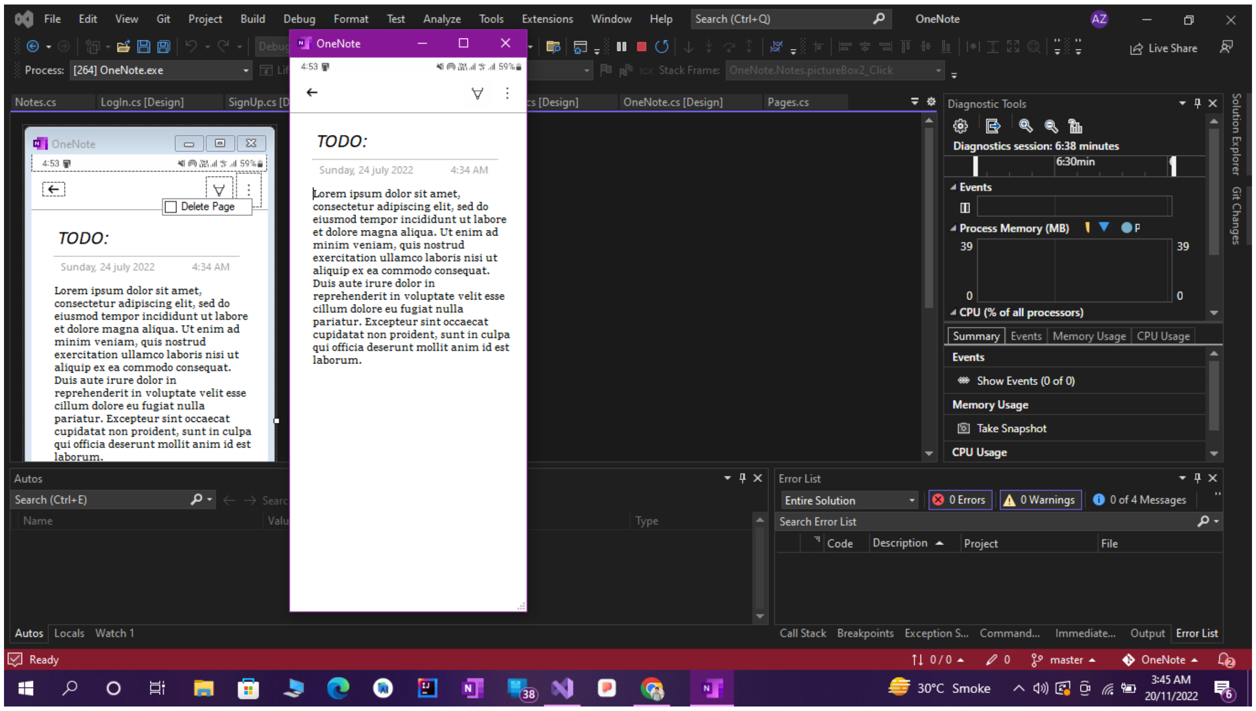


TODO:

Sunday, 24 july 2022 4:34 AM

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5. After Clicking TODO(Which is a Page) : (Now we are directed to Notes Screen)

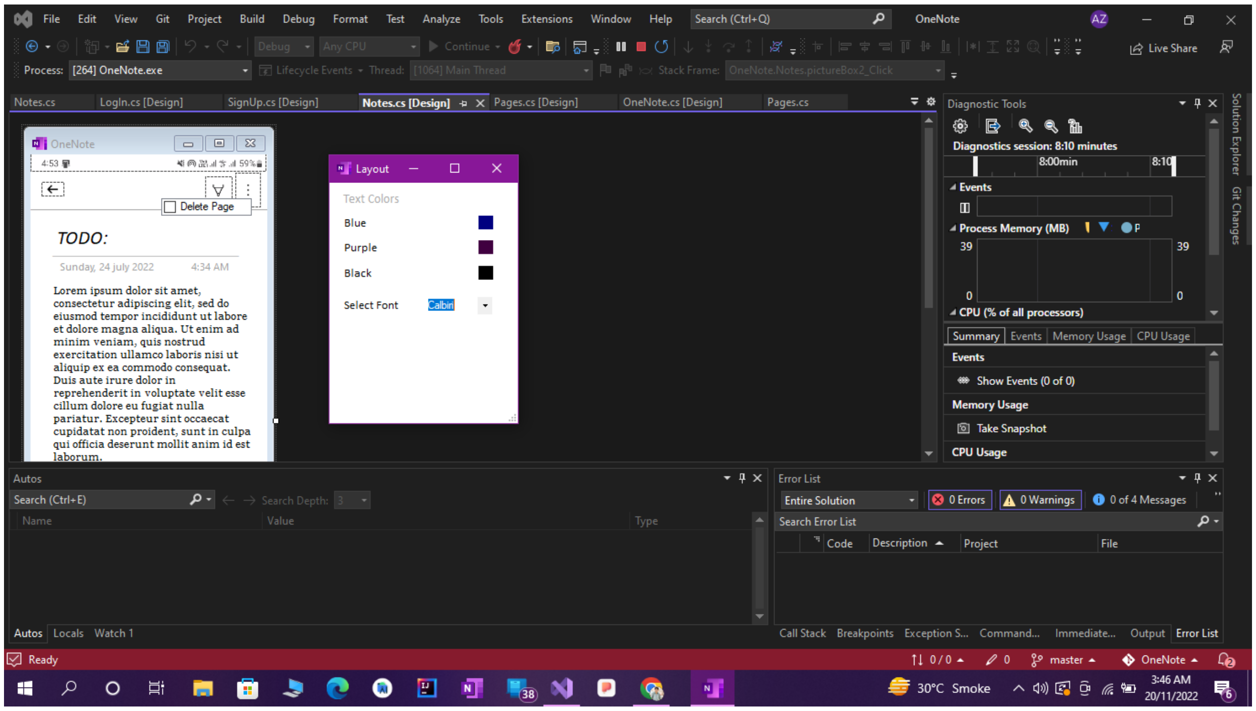


TODO:

Sunday, 24 july 2022 4:34 AM

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

6. You can Customize the Note by clicking on the Marker icon on top right.

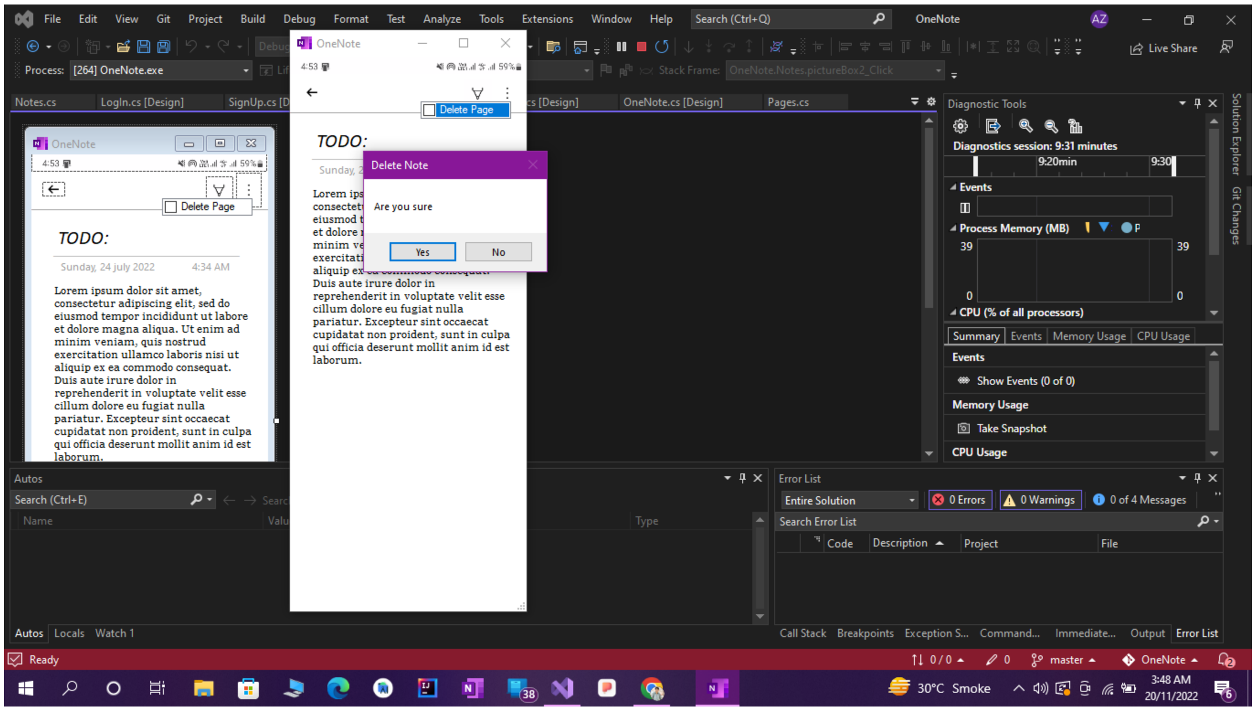


TODO:

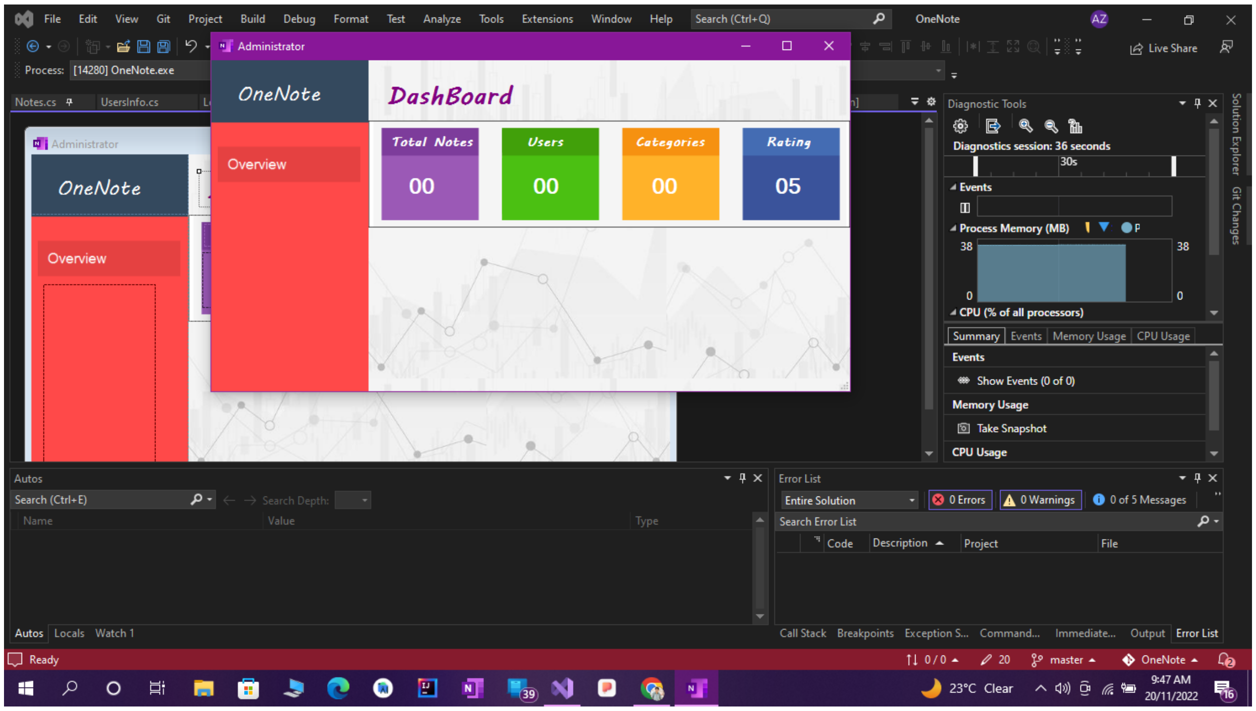
Sunday, 24 july 2022 4:34 AM

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

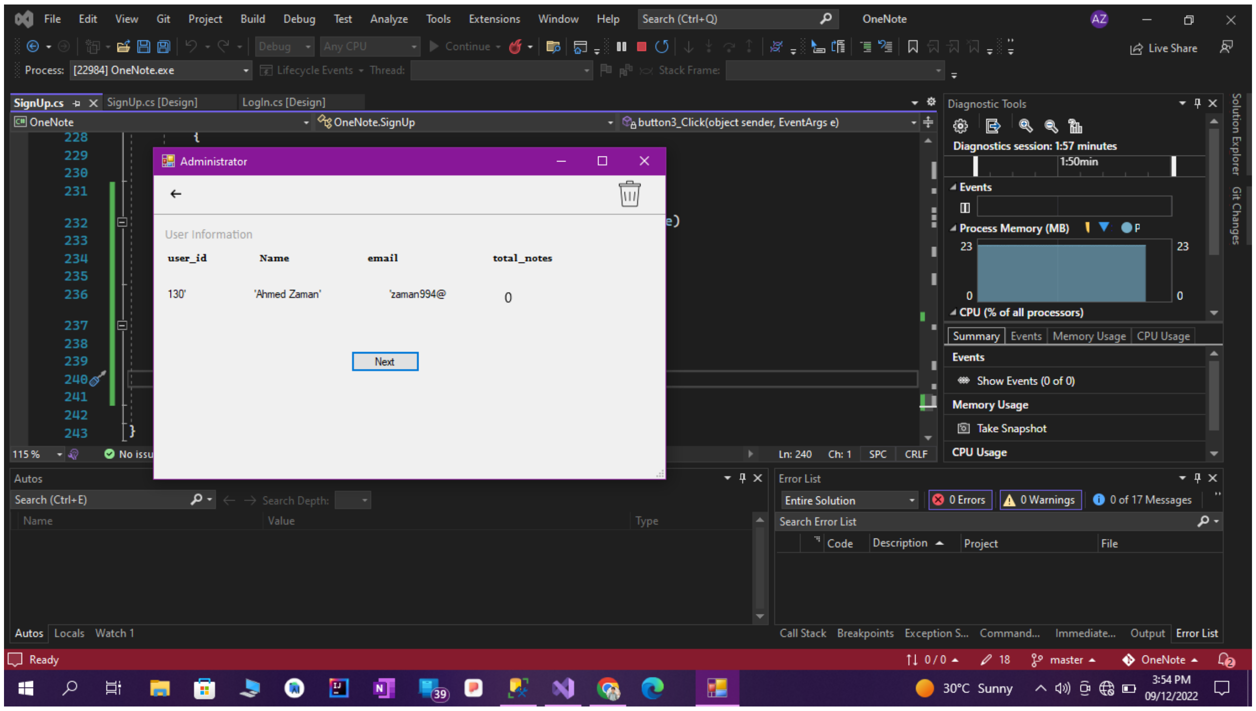
7. Or you can also delete this note by clicking on three dots on the top right corner of the screen.



8. Administrator Dashboard:



9. By clicking on the Users Dashboard Admin will get a new screen to overview the details of the user and also by clicking on the recycle bin admin can delete any user.



User Information

user_id	Name	email	total_notes
130'	'Ahmed Zaman'	'zaman994@	0

Code

1. Code For SignUp Form:

When the user clicks Create Notebook Button.

```
private void button1_Click(object sender, EventArgs e)
    {


        // check if the email already exists or not

        SqlConnection sqlCon = new SqlConnection();

        sqlCon.ConnectionString = "Data
Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial Catalog=OneNote;Integrated
Security=True";

        sqlCon.Open();
```

```
String firstname = textBox1.Text.Trim();
String lastname = textBox2.Text.Trim();
String email = textBox3.Text.Trim();
String password = textBox4.Text.Trim();

if (textBox3.Text.Contains(" "))
{
    // for the email
    MessageBox.Show("No Spaces allowed in email");
    textBox3.Text = null;
    email = "";
}

if (firstname == "" || lastname == "" || email == "" || password == "")
{
    MessageBox.Show("Plz enter all the details correctly:");
    this.Hide();
    this.Close();
    SignUp signUp = new SignUp();
    signUp.ShowDialog();
}
else
{
    if (check_existing_email(email) == true)
    {
        MessageBox.Show("Sorry: Entered email already exists:");
        this.Hide();
        this.Close();
        SignUp signUp = new SignUp();
        signUp.ShowDialog();
    }
    else
```

```
        {

            String query = "insert into [User](name,lastname,email,password)" +
           " values('" + firstname + "','" + lastname + "','" + email + "', '" + password +
"')";

            SqlCommand sqlCommand = new SqlCommand(query, sqlCon);
            sqlCommand.Connection = sqlCon;
            sqlCommand.CommandType = CommandType.Text;
            int a = sqlCommand.ExecuteNonQuery();
            if (a != -1)
            {
                add_data_to_NoteBookTable(firstname, email, sqlCon);
                MessageBox.Show("Account Created Successfully:");
                LogIn login = new LogIn();
                this.Hide();
                login.ShowDialog();
                this.Close();

            }
            else
            {
                MessageBox.Show("Error: Account can not be created");
            }


        }


    }
```

```
                    }
```

2. **For Login Form:**
    a. **When User Clicks Login Button:**

```
    private void button1_Click(object sender, EventArgs e)
{



  String email = textBox1.Text.TrimEnd();
  String pass = textBox2.Text.TrimEnd();

  //first varify if the entered acccount exists or not
  if (email == "" || pass == "")
  {
    MessageBox.Show("Plz enter all the details correctly:");
  }
  else if (isAdmin(email, pass) == true)
  {
    Administrator administrator = new Administrator();
    this.Hide();
    administrator.ShowDialog();
    this.Close();
  }
  else
  {

    //now check the account exists or not
    SqlConnection sqlCon = new SqlConnection();
    sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
    sqlCon.Open();
```

```
if (check_Account_existance(email, sqlCon) == true)
{
    //now check if the passwor entered for this email is correct.
    if (check_password(email, pass, sqlCon) == true)
    {
        //done
        //now user will be directed to the main page of application
        //and lorem's NoteBook Text should change into usrs's Notebook on the OneNote
Form

        String throw_email = email;

        throw_NoteBookName = getUserName(throw_email, sqlCon);
        throw_NoteBooKID = throwNOteBookID_to_PAges(email, sqlCon);

        OneNote one = new OneNote();
        this.Hide();
        one.notebookName = throw_NoteBookName;
        one.notebookID = throw_NoteBooKID;
        one.ShowDialog();
        this.Close();

    }

}
else
{
    sqlCon.Close();
    MessageBox.Show("No Account found with this email");
}
```

```
    }

}
```

3. **Code For OneNote Form (Front Page):**
    a. **When User clicks on his Notebook .**

```
     private void label1_Click(object sender, EventArgs e)
{
    var sections = new Sections();
    sections.heading = notebookName;
    sections.NOTEBOOKID = notebookID;
    sections.ShowDialog();

}


private void OneNote_Load(object sender, EventArgs e)
{
    //here u will do this
    // lorem's NoteBook Text should change into usrs's name Notebook on the OneNote Form:

    labelNoteBook.Text = notebookName + "'s NoteBook";

}
```

4. **Code For Section Form**
    a. **When User clicks on any of his/her Sections inside the Notebook:**

```
private void pictureBox3_Click(object sender, EventArgs e)
```

```
{
   //so that back icon is clicked the user go back to the same OneNote Form:
   this.Close();
}

private void label4_Click(object sender, EventArgs e)
{
   var OFedu = new Pages();
   OFedu.PagesFormHeading = "Education/Work";
   OFedu.NOTEBOOKid = NOTEBOOKID;
   OFedu.ShowDialog();

}

private void label5_Click(object sender, EventArgs e)
{
   var OFweekend = new Pages();
   OFweekend.PagesFormHeading = "Weekend";
   OFweekend.NOTEBOOKid = NOTEBOOKID;
   OFweekend.ShowDialog();

}

private void label6_Click(object sender, EventArgs e)
{
   var OFimp = new Pages();
   OFimp.PagesFormHeading = "Imp";
   OFimp.NOTEBOOKid = NOTEBOOKID;
   OFimp.ShowDialog();

}
public String heading = "";
```

```
private void Sections_Load(object sender, EventArgs e)
{
   //first display the name above
   label1heading.Text = heading + "'s NoteBook";
}
```

5. Code For Pages Form which are inside the Sections :

a.When The Page Form Loads:

```
 private void Page_Load(object sender, EventArgs e)
      {
         label1.Text = PagesFormHeading;


         //depending on sectionID and name make pages visible if user already have pages.
         //first through notebookid save the sectionid.
         int sectionID = -1;
         int notebookId = NOTEBOOKid;



         SqlConnection sqlCon = new SqlConnection();
         sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
         sqlCon.Open();
         if (notebookId != -1)
         {
            //MessageBox.Show(""+notebookId);
            SqlCommand sqlCommand = new SqlCommand("Select section_id from [Section]
where notebookID= '" + notebookId + "' and name = '" + PagesFormHeading + "' ", sqlCon);
            sectionID = (int)sqlCommand.ExecuteScalar();
            if (sectionID != -1)
            {
               // now u have sectionid inside which the user is currently in.
               //so save the count of the pages inside that section.
```

```
                //i am saving the count of noteid not the pageid becouse it will help later.

                countOfPage = getPageCount(sectionID, sqlCon);
                displaypages(countOfPage, sectionID, sqlCon);
            }
            else
            {
                MessageBox.Show("sectionid couldnot be found: pagesload()");
            }



        }
        else
        {
            MessageBox.Show("NotebookID could not be found in Pages.cs");
        } }
```

B. **When user clicks on Add Page:**

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

            //this not working.........> it worked

            int notebookId = NOTEBOOKid;
            //////////////////////
            int sectionID = -1;
            //to know in which section user is making pages:
            //first save the section name inside which user is
            //then save the notebookid for that section
            SqlConnection sqlCon = new SqlConnection();
            sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
            sqlCon.Open();
```

```
if (notebookId != -1)
{
    //MessageBox.Show(""+notebookId);
    SqlCommand sqlCommand = new SqlCommand("Select section_id from [Section] where notebookID= '" + notebookId + "' and name = '" + PagesFormHeading + "' ", sqlCon);
    sectionID = (int)sqlCommand.ExecuteScalar();
}
else
{
    MessageBox.Show("NotebookID could not be found in Pages.cs");
}

// make pages visibile- i have placed 10 pages so after each click make 1 visible
//each page will have  a note
//insert data into Table--> Page.




pagename = textBox1.Text.Trim();
if (pagename.Equals(""))
{
    MessageBox.Show("Plz enter the Name For your Note");
}
else if (checkPageExistance(sectionID, pagename, sqlCon))
{
    //add a method so that no page with same name can be created
    MessageBox.Show("Page with this name already exists in " + PagesFormHeading);
}
else
{
```

```
//first set visibility to false.
textBox1.Visible = false;
label111.Visible = false;
buttonAddPage.Visible = false;
button1.Visible = false;

//depending on count make each page visible



if (sectionID != -1)
{

    count = getPageCount(sectionID, sqlCon) + 1;
    if (count == 1)
    {






        labelpage1.Text = pagename.ToUpper();
        labelpage1.Visible = true;
        addData_to_Pages(pagename, sectionID, sqlCon);



        this.Close();


    }
    if (count == 2)
    {
```

```
    labelpage2.Text = pagename.ToUpper();
    labelpage2.Visible = true;
    addData_to_Pages(pagename, sectionID, sqlCon);
    this.Close();

}
if (count == 3)
{


    labelpage3.Text = pagename.ToUpper();
    labelpage3.Visible = true;
    addData_to_Pages(pagename, sectionID, sqlCon);
    this.Close();




}
if (count == 4)
{


    labelpage4.Text = pagename.ToUpper();
    labelpage4.Visible = true;
    addData_to_Pages(pagename, sectionID, sqlCon);
    this.Close();



}
if (count == 5)
{
    labelpage5.Text = pagename.ToUpper();
    labelpage5.Visible = true;
    addData_to_Pages(pagename, sectionID, sqlCon);
    this.Close();



}
if (count == 6)
{
    labelpage6.Text = pagename.ToUpper();
    labelpage6.Visible = true;
    addData_to_Pages(pagename, sectionID, sqlCon);
    this.Close();



}
if (count == 7)
{


    labelpage7.Text = pagename.ToUpper();
    labelpage7.Visible = true;
    addData_to_Pages(pagename, sectionID, sqlCon);
    this.Close();


}
if (count == 8)
{
    labelpage8.Text = pagename.ToUpper();
    labelpage8.Visible = true;
    addData_to_Pages(pagename, sectionID, sqlCon);
    this.Close();
    }
    if (count == 9)
    {
        labelpage9.Text = pagename.ToUpper();
        labelpage9.Visible = true;
        addData_to_Pages(pagename, sectionID, sqlCon);
        this.Close();
    }
    if (count == 10)
    {

        labelpage10.Text = pagename.ToUpper();
        labelpage10.Visible = true;
        addData_to_Pages(pagename, sectionID, sqlCon);
        this.Close();


    }
    if (count > 10)
    {

        MessageBox.Show("Only 10 Pages Are Allowed In Each Section");

    }

}
else
{
    MessageBox.Show("Sorry: Section not Specified");
}
```

```
                    }

                }
```

6. **Code For Notes Form:**
    a. **When Notes Form Loads:**

```
        private void Notes_Load(object sender, EventArgs e)
    {
            SqlConnection sqlCon = new SqlConnection();
            sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
            sqlCon.Open();

            SqlCommand sqlCommand = new SqlCommand("select text from Note where note_id = '" +
NOTEid + "'", sqlCon);
            sqlCommand.Connection = sqlCon;
            sqlCommand.CommandType = CommandType.Text;
            richTextBox1.Text = sqlCommand.ExecuteScalar().ToString();



            //to get NoteName \(noteName = paganame)
            SqlCommand sqlCommand1 = new SqlCommand("select pageTitle from [Page] where note_id
= '" + NOTEid + "'", sqlCon);
            sqlCommand1.Connection = sqlCon;
            sqlCommand1.CommandType = CommandType.Text;
            label1.Text = sqlCommand1.ExecuteScalar().ToString();

            //to getDate of Page Creation
            SqlCommand sqlCommand2 = new SqlCommand("select creationDate from [Note] where
note_id = '" + NOTEid + "'", sqlCon);
            sqlCommand2.Connection = sqlCon;
            sqlCommand2.CommandType = CommandType.Text;
            label4.Text = sqlCommand2.ExecuteScalar().ToString();


            setFont(NOTEid, sqlCon);
            setColor(NOTEid, sqlCon);
        }
```

B. **When User wants to delete his Note:**

```
 private void deleteNote()
        {
            SqlConnection sqlCon = new SqlConnection();
            sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
            sqlCon.Open();


            SqlCommand sqlCommand = new SqlCommand("delete from Page  where note_id = '" +
NOTEid + "'", sqlCon);
            sqlCommand.Connection = sqlCon;
            sqlCommand.CommandType = CommandType.Text;
            int b = (int)sqlCommand.ExecuteNonQuery();
            if (b != -1)
            {
                SqlCommand sqlCommand11 = new SqlCommand("delete from NoteLayout  where note_id
= '" + NOTEid + "'", sqlCon);
                sqlCommand11.Connection = sqlCon;
                sqlCommand11.CommandType = CommandType.Text;
                int c = (int)sqlCommand11.ExecuteNonQuery();


                if (c != -1)
                {
                    SqlCommand sqlCommand1 = new SqlCommand("delete from Note  where note_id = '" +
NOTEid + "'", sqlCon);
                    sqlCommand1.Connection = sqlCon;
                    sqlCommand1.CommandType = CommandType.Text;
                    int a = (int)sqlCommand1.ExecuteNonQuery();
                    if (a != -1)
                    {
                        //  MessageBox.Show("Deleted");
                    }
                    else
                    {
                        MessageBox.Show("Not Deleted from Note");
                    }
                }
                else
                {
                    MessageBox.Show("Not Deleted from Note");
                }



            }
            else
            {
                MessageBox.Show("Error");
            }
        }
```

7. Code For NoteLayout:

```
            {
        SqlConnection sqlCon = new SqlConnection();
        sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
        sqlCon.Open();



        SqlCommand sqlCommand = new SqlCommand("update NoteLayout set textcolor= '" + color
+ "' where note_id = '" + NOTEID + "'", sqlCon);
```

```
            sqlCommand.Connection = sqlCon;
            sqlCommand.CommandType = CommandType.Text;
            int b = (int)sqlCommand.ExecuteNonQuery();
            if (b != -1)
            {

                MessageBox.Show("color added");
            }
            else
            {
                MessageBox.Show("color not set");
            }

        }

        public void selectFont(String fontName)
        {
            SqlConnection sqlCon = new SqlConnection();
            sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
            sqlCon.Open();



            SqlCommand sqlCommand = new SqlCommand("update NoteLayout set fontname= '" +
fontName + "' where note_id = '" + NOTEID + "'", sqlCon);
            sqlCommand.Connection = sqlCon;
            sqlCommand.CommandType = CommandType.Text;
            int b = (int)sqlCommand.ExecuteNonQuery();
            if (b != -1)
            {

                MessageBox.Show("Font Applied");
            }
            else
            {
                MessageBox.Show("font not set");
            }


        }
```

8. Code For Administrator Form

```
  private void Administrator_Load(object sender, EventArgs e)
        {
            SqlConnection sqlCon = new SqlConnection();
            sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True";
            sqlCon.Open();


            totalUserLabel.Text = totalUsers(sqlCon).ToString();
            totalNotesLabel.Text = totalNotes(sqlCon).ToString();
            categoryLabel.Text = totalCategories(sqlCon).ToString();
        }
```

9. Code For User Info Form (Admin second Form);

A.

```
          private void UsersInfo_Load(object sender, EventArgs e)
                {
                    buttonClickCOunt = 0;
                    SqlConnection sqlCon = new SqlConnection();
                    sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
        Catalog=OneNote;Integrated Security=True";
                    sqlCon.Open();
                    int user_id = getFirstUserid();
                    getColumns(user_id, sqlCon);
                }

                private void getColumns(int user_id, SqlConnection sqlCon)
                {
                    try
                    {

                        String query = "select user_id,CONCAT([User].name,'" + " " + "'
        ,lastname),email,count(page_id) from [User]" +
                            "\r\njoin Section on Section.notebookID=[User].NotebookID\r\n" +
                            "join Page on Page.section_id = Section.section_id \r\ngroup by
        user_id,[User].name," +
                            "lastname,email\r\nhaving user_id ='" + user_id + "'";
                        SqlCommand cmd = new SqlCommand(query, sqlCon);

                        SqlDataReader reader = cmd.ExecuteReader();

                        if (reader.HasRows == true)
                        {
                            reader.Read();
                            String space = new String(' ', 25);

                            firstcolumn.Text = (string)reader[0].ToString() + "" + space + "" +
        (string)reader[1].ToString() + "" + space + ""
                                + (string)reader[2].ToString() + "" + space + 10 + "" +
        (string)reader[3].ToString();
                        }
                        else
                        {
                            reader.Close();
                            String query2 = "select user_id,CONCAT([User].name,'" + " " + "' ,lastname),email
        from [User]" +
                                "where user_id= '" + user_id + "'";
                            SqlCommand cmd2 = new SqlCommand(query2, sqlCon);

                            SqlDataReader reader2 = cmd2.ExecuteReader();

                            reader2.Read();
                            String space = new String(' ', 25);

                            firstcolumn.Text = (string)reader2[0].ToString() + "" + space + "" +
                  (string)reader2[1].ToString() + "" + space + ""
                              + (string)reader2[2].ToString();
                          }
                        }
                        catch (Exception e)
                        {
                          return;
                        }
                      }
```

B. When Admin Wants to Remove The User:

```
     private void show3_Click(object sender, EventArgs e)
 {
   if (show2.Text.Equals(""))
   {
     MessageBox.Show("Enter User_id");
   }
   else
   {
     String user_id = show2.Text.ToString();
     for (int i = 0; i < user_id.Length; i++)
     {
       if (!char.IsNumber(user_id[i]))
       {
```

```
                MessageBox.Show("Please enter a valid number");
                show2.Text = "";
                user_id = "-1";
            }

        }

        if (!user_id.Equals("-1"))
        {
            SqlConnection sqlCon = new SqlConnection();
            sqlCon.ConnectionString = "Data Source=DESKTOP-3MGQIPF\\UNIQUENAME;Initial
Catalog=OneNote;Integrated Security=True ;MultipleActiveResultSets=true";
            sqlCon.Open();



            SqlCommand sqlCommand = new SqlCommand("IF exists (Select user_id from [User]
where user_id= '" + user_id + "') select 1 else select -1 ", sqlCon);
            int a = (int)sqlCommand.ExecuteScalar();

            if (a != -1)
            {
                int USER_ID = int.Parse(user_id);

                //delete user
                //  deleteUser(USER_ID, sqlCon);
                deleteAllDataofThisUser(USER_ID, sqlCon);



            }
            else
            {
                    MessageBox.Show("Userid '" + user_id + "' not exists");

                }

            }

        }
    }

    private int getNotebookId(int user_id, SqlConnection sqlCon)
    {
        SqlCommand sqlCommand = new SqlCommand("Select notebookid from Notebook where
email=" +
            "(select email from [User] where user_id = '" + user_id + "') ", sqlCon);
        int notebookId = (int)sqlCommand.ExecuteScalar();
        return notebookId;
    }

    private void deleteNotebookId(int NoteBookid, SqlConnection sqlCon)
    {
        SqlCommand sqlCommand = new SqlCommand("Delete from Notebook where notebookid ='"
+ NoteBookid + "'", sqlCon);
        sqlCommand.ExecuteNonQuery();
    }
```

```
        private int getSectionId(int user_id, int noteBookid, SqlConnection sqlCon)
        {

            String query = "select section_id from Section where notebookID ='" + noteBookid + "' and
name='Education/Work'";
            SqlCommand sqlCommand = new SqlCommand(query, sqlCon);
            return (int)sqlCommand.ExecuteScalar();
        }

        private int PageCountOfthisUser(int user_id, int sectionId, SqlConnection sqlCon)
        {
            String query = "if exists (select page_id from [Page] where Section_id ='" + sectionId + "' or
Section_id = '" + (sectionId + 1) + "'  or Section_id = '" + (sectionId + 2) + "') " +
            "Select count(page_id) from [Page] where Section_id ='" + sectionId + "' or Section_id = '" +
(sectionId + 1) + "'  or Section_id = '" + (sectionId + 2) + "' ELSE Select 0";
            SqlCommand sqlCommand = new SqlCommand(query, sqlCon);
            return (int)sqlCommand.ExecuteScalar();



        }



        private int[] getPageId(int user_id, int sectionId, int pageCount, SqlConnection sqlCon)
        {

            int[] page_id_arr = new int[pageCount];
            if (pageCount > 0)
            {

                String query = "select page_id from [Page] where Section_id ='" + sectionId + "' or Section_id
= '" + (sectionId + 1) + "'  or Section_id = '" + (sectionId + 2) + "' ";
                SqlCommand sqlCommand = new SqlCommand(query, sqlCon);
                SqlDataReader reader = sqlCommand.ExecuteReader();



                for (int i = 0; i < pageCount; i++)
                {
                    reader.Read();
                    page_id_arr[i] = (int)reader[0];

                }

                return page_id_arr;

            }
            else
            {

                return page_id_arr;
            }
        }

        private int[] getNoteid(int[] pageids, SqlConnection sqlCon)
        {
            int[] noteids = new int[pageids.Length];

            for (int i = 0; i < pageids.Length; i++)
            {
                SqlCommand sqlCommand = new SqlCommand("select note_id from Page where page_id=
'" + pageids[i] + "'", sqlCon);
                noteids[i] = (int)sqlCommand.ExecuteScalar();
            }



            return noteids;
        }

        private int[] getNoteLayoutids(int[] noteids, SqlConnection sqlCon)
        {
            int[] notelayoutIds = new int[noteids.Length];
            for (int i = 0; i < noteids.Length; i++)
            {
                SqlCommand sqlCommand = new SqlCommand("select layout_id from NoteLayout where
note_id='" + noteids[i] + "'", sqlCon);
                notelayoutIds[i] = (int)sqlCommand.ExecuteScalar();

            }
            return notelayoutIds;

        }

        private void deleteAllDataofThisUser(int userid, SqlConnection sqlCon)
        {
            int Notebookid = getNotebookId(userid, sqlCon);
            int Sectionid = getSectionId(userid, Notebookid, sqlCon);
            int pageCount = PageCountOfthisUser(userid, Sectionid, sqlCon);
            int[] pageid = getPageId(userid, Sectionid, pageCount, sqlCon);
            int[] Noteid = getNoteid(pageid, sqlCon);
            int[] NoteLayoutid = getNoteLayoutids(Noteid, sqlCon);



            deleteNoteLayout(NoteLayoutid, sqlCon);
            deletePage(Sectionid, userid, sqlCon);
            deleteNotes(Noteid, sqlCon);
            deleteSection(Sectionid, sqlCon);
            deleteUser(userid, sqlCon);
            deleteNotebookId(Notebookid, sqlCon);


        }

        private void deleteNotes(int[] noteids, SqlConnection sqlCon)
        {

            for (int i = 0; i < noteids.Length; i++)
            {
                SqlCommand sqlCommand = new SqlCommand("delete from Note where note_id='" +
noteids[i] + "'", sqlCon);
                sqlCommand.ExecuteNonQuery();
            }
        }



        private void deleteNoteLayout(int[] notelayoutids, SqlConnection sqlCon)
        {
            for (int i = 0; i < notelayoutids.Length; i++)
            {
                SqlCommand sqlCommand = new SqlCommand("delete from NoteLayout where
layout_id='" + notelayoutids[i] + "'", sqlCon);
                sqlCommand.ExecuteNonQuery();


            }
        }

        private void deleteSection(int sectionid, SqlConnection sqlCon)
        {
            SqlCommand sqlCommand = new SqlCommand("delete from Section where section_id='" +
sectionid + "' or section_id='" + (sectionid + 1) + "' or section_id='" + (sectionid + 2) + "'  ", sqlCon);
            sqlCommand.ExecuteNonQuery();



        }




        private void deleteUser(int user_id, SqlConnection sqlCon)
        {
            SqlCommand sqlCommand = new SqlCommand("Delete  from [User] where user_id= '" +
user_id + "' ", sqlCon);
            int a = (int)sqlCommand.ExecuteNonQuery();
            if (a != -1)
            {

                MessageBox.Show("User Deleted");
            }
            else
            {
                MessageBox.Show("Error while deleting user");
            }
        }

        private void deletePage(int Sectionid, int user_id, SqlConnection sqlCon)
        {

            SqlCommand sqlCommand = new SqlCommand("delete from Page where section_id = '" +
Sectionid + "' Or  section_id = '" + (Sectionid + 1) + "' Or  section_id = '" + (Sectionid + 2) + "'  ", sqlCon);
            sqlCommand.ExecuteNonQuery();
```

```
}
```

Conclusion

1. **The group's experience with the project: which steps were the most difficult?**

   The most difficult parts of the project were:

   a. **Database:** One of the challenges I faced during this project was the need to change database design as I progressed through the project. I initially designed m ydatabase in a certain way, but as I worked on the project and encountered new challenges and requirements, I realized that my original design was not sufficient and had to be revised. This is a difficult and time-consuming process, as it involves making changes to the underlying structure of the database. However, by adapting to these challenges and updating my database design, I was able to overcome this difficulty.

   b. **Page Form:** Another challenge I faced during this project was implementing the ability for users to add pages to the app and have the form automatically update to display the new pages. This was a difficult task because it involved dynamically modifying the user interface of the app at runtime, which can be complex and require careful planning and implementation. In order to successfully implement this feature, I need to design the user interface in a flexible way that allows for changes to be made to it while the app is running.. Overall, this was a challenging but important part of the project, and by overcoming this difficulty I was able to add an important feature to the app.

c. **Admin ability to delete:**A third challenge   faced during the project was implementing the ability for the admin user to delete other users and their associated data from the app. This was a complex task because it involves not only removing the user's account from the app, but also all of the data that the user has created, such as sections, pages, notes, and layouts. This requires careful planning and implementation, as I need to ensure that all of the user's data is properly deleted from the database, without affecting the data of other users or causing any errors or inconsistencies in the app. In order to implement this feature, you would need to design the database and the app's code in a way that allows for the efficient and reliable deletion of user data, including handling any dependencies or relationships between different pieces of data.

2. **Which were the easiest?**

   The easiest part was to design the project as it only required drag and drop.

3. **What did you learn that you did not imagine you would have?**

   By working on a project that involved creating a clone of Microsoft OneNote using C# and .NET, likely gained experience with technologies and programming languages that are commonly used in software engineering. Additionally, the project is likely required to work with database management systems, which can be an important skill for software engineers. By implementing features such as the ability to add pages and delete users,   have also learned about design and implementation strategies for complex software systems, and gained experience with the challenges and rewards of working on a real-world software project. Overall, by completing this project, I likely gained a wide range of valuable skills and knowledge that will be useful in the future career as a software engineer.

4. **If you had to do it all over again, what would you have done differently?**

   There are likely several things I would have done differently. For example, I may have approached the design of the database in a different way, or implemented certain features in a different way. Additionally, may have encountered new challenges or requirements during the project that didn't anticipate the first time around, and would need to adapt the approach in order to successfully complete the project. Overall, if had to do the project again, would likely make different decisions and take different approaches based on the new experiences and knowledge gained from completing the project the first time.