Learning Objectives

- Understand the key differences between C++ and C# syntax and convert basic programs from C++ to C#.
- Establish a secure connection between a MySQL database and a C# application.
- Perform CRUD (Create, Read, Update, Delete) operations in C# using MySQL.
- Learn how to set up and configure a Google Cloud Platform (GCP) account for database hosting.

LO1: Understand the key differences between C++ and C# syntax and convert basic programs from C++ to C#.

Basic Program Structure

- C# uses **using** instead of **#include**
- C# code can execute without **Main()** function
- No **explicit return** statement needed in C#

C++	C#
`#include <iostream>` `int main() {` `    // code here` `    return 0;` `}`	`Console.WriteLine("Hello, World");`

Hello world

- C# uses Console.WriteLine() or Console.Write () to print any line on console.
- Console.WriteLine() display line on the console and move the cursor to next line.
- Console.Write() display line on the console and the cursor will remain on the same line.

C++	C#
#include <iostream> using namespace std; int main() { **cout << "Hello, World!" << endl; // move to next line** **cout << "Hello, World!"; // stay on the same line** return 0; }	using System; class Program { static void Main() { **Console.WriteLine("Hello, World!"); // move to next line** **Console.Write("Hello, World!"); // stay on the same line** } }

Reading Input

- C# uses Console.ReadLine() or Console.Read() to take input from user.
- Console.ReadLine() read the string and Console.Read() just read one character
- You need to convert the input into specific datatype as input of C# is always in string datatype.

C++	C#
#include <iostream> using namespace std; int main() { int number; cout << "Enter a number: "; **cin >> number;** cout << "You entered: " << number << endl; return 0; }	using System; class Program { static void Main() { Console.Write("Enter a number: "); **int number = Convert.ToInt32(Console.ReadLine());** Console.WriteLine("You entered: " + number); } }

Conditional Statements

- The Conditional Statements are same in C#

C++	C#
if (x > 0) { // code }	if (x > 0) { *// code* }

Loops

- The Loops are same in C#

For Loop:

C++	C#
`for (int i = 0; i < 5; i++) {` `    // code` `}`	`for (int i = 0; i < 5; i++) {` `    // code` `}`

While Loop:

C++	C#
`while (i < 5) {` `    // code` `}`	`while (i < 5) {` `    // code` `}`

Arrays

- C# arrays are objects with properties like Length (to calculate length of array)
- Array size can be determined at runtime

C++	C#
`int arr[5];`	`int[] arr = new int[5];`

Practice Tasks:

1. **P1Task1.cs:** Convert the following C++ program to C#:

```
#include <iostream>
using namespace std;
int main() {
    cout << "Enter your name: ";
    string name;
    cin >> name;
    cout << "Hello, " << name << "!" << endl;
    return 0;
}
```

2. **P1Task2.cs:** Convert the following C++ program to C#:

```
for (int i = 10; i > 0; i--) {
    cout << i << " ";
}
```

3. **P1Task3.cs:** Implement a C# program to calculate the sum of numbers from 1 to n. Take the value of n as input from the user.
4. **P1Task4.cs:** Write a C# program to check if a given number is even or odd.
5. **P1Task5.cs:** Write a C# program that takes a string input from the user and prints each character on a new line.
6. **P1Task6.cs:** Compare the syntax for declaring arrays in C++ and Python with that in C#. Write a C# program to store and display 5 integers.

Submission Requirements

- Submit the following files:
  - P1Task1.cs
  - P1Task2.cs
  - P1Task3.cs
  - P1Task4.cs
  - P1Task5.cs
  - P1Task6.cs

LO2: Establish a secure connection between a MySQL database and a C# application.

Understanding Connection Strings

A connection string is a string that specifies information about a data source and the means of connecting to it. Here's the anatomy of a MySQL connection string:

```
server=127.0.0.1;          // Database server address
port=3306;                 // Port on which server is running
user=your_username;        // Username by default it is “root”
database=database_name;    // Database name
password=your_password;    // Password
SslMode=Required;          // Security setting
```

Example:

```
server=127.0.0.1;port=3306;user=root;database=Lab2;password=
yourdatabasePassword;SslMode=Required;
```

Connection Building Steps:

- Install the MySql.Data NuGet package.
- Use MySqlConnection for establishing a connection.

We will perform these steps in the next Learning objective

LO3: Perform CRUD (Create, Read, Update, Delete) operations in C# using MySQL.

CRUD Operations in MySQL

- **Insert Data into Table:**

```
INSERT INTO table_name (column1, column2, column3,…)
VALUES (value1, value2, value3, ...);
```

- **Update Data into Table:**

```
UPDATE table_name
SET column1 = value1, column2 = value2,…
WHERE condition;
```

- **Retrieve Data from Table:**

```
SELECT column1, column2,…
FROM table_name;
```

- **Delete Data from a Table:**

```
DELETE FROM table_name WHERE condition;
```

Step 1: Create a New Database

1. Open MySQL Workbench and connect to your MySQL server.
2. Create a new database:
   ```
   CREATE DATABASE Lab2;
   ```
3. Verify the database creation:
   ```
   SHOW DATABASES;
   ```

Step 2: Create a Table

Note: There is no need to understand all these commands, these commands are used to create tables, we will explore these commands in more detail in upcoming weeks.

1. Use the Lab2 database:

```
USE Lab2;
```

2. Create a table named Student with the following schema:

```
CREATE TABLE Student (
    RegistrationNumber VARCHAR(15) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Department VARCHAR(50) NOT NULL,
    Session INT NOT NULL,
    CGPA DECIMAL(3, 2) NOT NULL,
    Address VARCHAR(255)
);
```

3. Verify the table creation:

```
SHOW TABLES;
```

Now, we will implement CRUD operations using a CLI-based C# project. Follow the steps below:

Step 3: Setup the Project

- Open Visual Studio and create a new C# Console Application (.NET Framework) project.
- Add the MySql.Data package via NuGet for MySQL database connectivity.

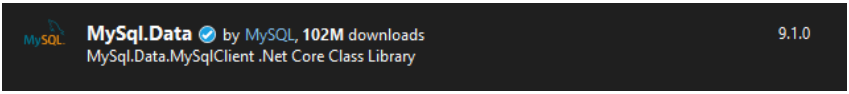

Figure 1 : MySQL.Data NuGet Package

(Tools -> NuGet Package Manager -> Manage NuGet Packages for Solution -> Search MySQL.Data -> Checkbox your project -> Install)

You have to do this step for every project.

- **DatabaseHelper Class:**
  Add following code in the class name Helper of your project and replace the “yourpassword” and “yourusername” in connection string with your actual password and username of MySQL

```
using MySql.Data.MySqlClient;
using Org.BouncyCastle.Tls;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab2
{
    public class DatabaseHelper
    {
        private String serverName = "127.0.0.1";
        private String port = "3306";
        private String databaseName = "Lab2";
        private String databaseUser = "root";
        private String databasePassword = "2002";

        private DatabaseHelper() { }

        private static DatabaseHelper _instance;
        public static DatabaseHelper Instance
        {
            get
            {
                if (_instance == null)
                    _instance = new DatabaseHelper();
                return _instance;
            }
        }
        public MySqlConnection getConnection()
        {
            string connectionString =
$"server={serverName};port={port};user={databaseUser};databa
se={databaseName};password={databasePassword};SslMode=Requir
ed;";
            var connection = new
MySqlConnection(connectionString);
            connection.Open();
```

```
            return connection;
        }

        public MySqlDataReader getData(string query)
        {
            using (var connection = getConnection())
            {
                using (var command = new MySqlCommand(query,
getConnection()))
                {
                    return command.ExecuteReader();
                }
            }

        }

        public int Update(string query)
        {
            using (var connection = getConnection())
            {
                using (var command = new MySqlCommand(query,
getConnection()))
                {
                    return command.ExecuteNonQuery();
                }
            }

        }
    }
}
```

You have to use this in every function which is accessing database to get data or updating the data in the database.

- **Student Class:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab2
{
```

```
    public class Student
    {
        public string RegNo { get; set; }
        public string Name { get; set; }
        public string Department { get; set; }
        public int Session { get; set; }
        public float Cgpa { get; set; }
        public string Address { get; set; }

        public Student(string regNo, string name = null,
string department = null, int session = 0, float cgpa = 0.0f,
string address = null)
        {
            RegNo = regNo;
            Name = name;
            Department = department;
            Session = session;
            Cgpa = cgpa;
            Address = address;
        }

        public void AddStudent()
        {
            string query = $"INSERT INTO Student VALUES
('{RegNo}', '{Name}', '{Department}', {Session}, {Cgpa},
'{Address}')";
            DatabaseHelper.Instance.Update(query);
        }

        public void EditStudent()
        {
            string query = $"UPDATE Student SET Name =
'{Name}', CGPA = {Cgpa} WHERE RegNo = '{RegNo}'";
            DatabaseHelper.Instance.Update(query);
        }

        public void DeleteStudent()
        {
            string query = $"DELETE FROM Student WHERE RegNo =
'{RegNo}'";
            DatabaseHelper.Instance.Update(query);
        }
```

```
        public void SearchStudent()
        {
            string query = $"SELECT * FROM Student WHERE RegNo
= '{RegNo}'";
            var reader =
DatabaseHelper.Instance.getData(query);
            if (reader.Read())
            {
                Console.WriteLine($"{reader["RegNo"]} -
{reader["Name"]} - {reader["Department"]} -
{reader["Session"]} - {reader["Cgpa"]} -
{reader["Address"]}");
            }
            else
            {
                Console.WriteLine("Student not found.");
            }
        }

        public void ShowStudents()
        {
            string query = "SELECT * FROM Student";
            var reader =
DatabaseHelper.Instance.getData(query);
            while (reader.Read())
            {
                Console.WriteLine($"{reader["RegNo"]} -
{reader["Name"]} - {reader["Department"]} -
{reader["Session"]} - {reader["Cgpa"]} -
{reader["Address"]}");
            }
        }
    }
}
```

- **Driver Code:**

```
using MySql.Data.MySqlClient;
using MySqlX.XDevAPI;
using System;
using System.Collections.Generic;
using System.ComponentModel.Design;
using System.Linq;
using System.Text;
```

```
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace Lab2
{
    public class Program
    {
        static void Main(string[] args)
        {
            string name, regNo, department, address;
            int session;
            float cgpa;
            Student student;

            Console.WriteLine("Welcome to Student Management
    System");
            int choice = Menu();
            while (choice != 6)
            {
                switch (choice)
                {
                    case 1:
                        Console.WriteLine("Enter RegNo: ");
                        regNo = Console.ReadLine();
                        Console.WriteLine("Enter Name: ");
                        name = Console.ReadLine();
                        Console.WriteLine("Enter Department:
    ");
                        department = Console.ReadLine();
                        Console.WriteLine("Enter Session: ");
                        session =
    Convert.ToInt32(Console.ReadLine());
                        Console.WriteLine("Enter CGPA: ");
                        cgpa =
    Convert.ToSingle(Console.ReadLine());
                        Console.WriteLine("Enter Address: ");
                        address = Console.ReadLine();

                        student = new Student(regNo, name,
    department, session, cgpa, address);
                        student.AddStudent();
                        break;
                    case 2:
                        Console.WriteLine("Enter RegNo: ");
                        regNo = Console.ReadLine();
```

```
                        Console.WriteLine("Enter Name: ");
                        name = Console.ReadLine();
                        Console.WriteLine("Enter CGPA: ");
                        cgpa =
    Convert.ToSingle(Console.ReadLine());

                        student = new Student(regNo, name,
    cgpa: cgpa);
                        student.EditStudent();
                        break;
                    case 3:
                        Console.WriteLine("Enter RegNo: ");
                        regNo = Console.ReadLine();

                        student = new Student(regNo);
                        student.DeleteStudent();
                        break;
                    case 4:
                        Console.WriteLine("Enter RegNo: ");
                        regNo = Console.ReadLine();

                        student = new Student(regNo);
                        student.SearchStudent();
                        break;
                    case 5:
                        student = new Student(null);
                        student.ShowStudents();
                        break;
                    default:
                        Console.WriteLine("Invalid choice");
                        break;
                }
                Console.ReadKey();
                choice = Menu();
            }

        }
        public static int Menu()
        {
            Console.Clear();
            Console.WriteLine("1. Add a New Student");
            Console.WriteLine("2. Edit a Student");
            Console.WriteLine("3. Delete a Student");
            Console.WriteLine("4. Search a Student");
            Console.WriteLine("5. Show all Students");
            Console.WriteLine("6. Exit");
```

```
            Console.WriteLine("Enter your choice: ");
            int choice = Convert.ToInt32(Console.ReadLine());
            return choice;
        }
    }
}
```

Tasks

P2Task1: Create Additional Tables

Create a database named Lab2_Task with the following schema:

1. **Student Table:**

```
CREATE TABLE Student (
    RegistrationNumber VARCHAR(15) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Department VARCHAR(50) NOT NULL,
    Session INT NOT NULL,
    Address VARCHAR(255)
);
```

2. **Course Table:**

```
CREATE TABLE Course (
    Course_ID VARCHAR(10) PRIMARY KEY,
    Course_Name VARCHAR(100) NOT NULL
);
```

3. **Enrollments Table:**

```
CREATE TABLE Enrollments (
    StudentRegNo VARCHAR(15),
    Course_ID VARCHAR(10),
    FOREIGN KEY (StudentRegNo) REFERENCES
Student(RegistrationNumber),
    FOREIGN KEY (Course_ID) REFERENCES
    Course(Course_ID)
);
```

4. **Attendance Table:**

```
CREATE TABLE Attendance (
    StudentRegNo VARCHAR(15),
    Course_ID VARCHAR(10),
    TimeStamp DATETIME NOT NULL,
    Status BOOLEAN NOT NULL,
```

```
    FOREIGN KEY (StudentRegNo) REFERENCES
Student(RegistrationNumber),
    FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID)
);
```

P2Task2: CRUD Operations

1. Perform CRUD operations on Student as you did in the first task
2. Perform CRUD operations on Courses
3. Create a UI to register a student and unregister from the course.
4. Create a page to mark and edit attendance for a particular course and list down attendance of a particular date.

User Interfaces:

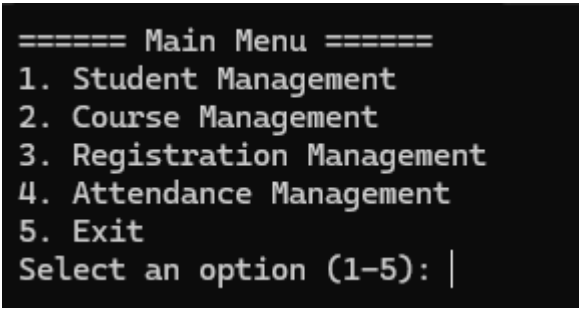

Figure 2 : Main Menu

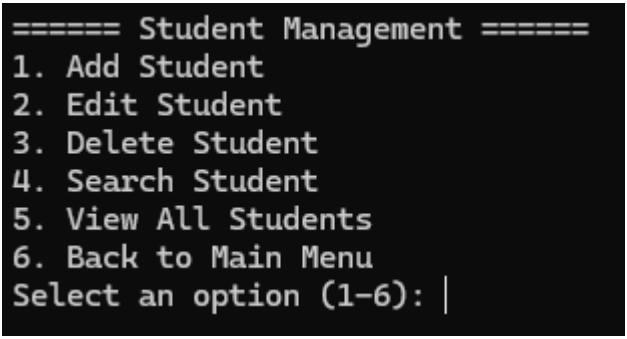

Figure 3 : Student Menu

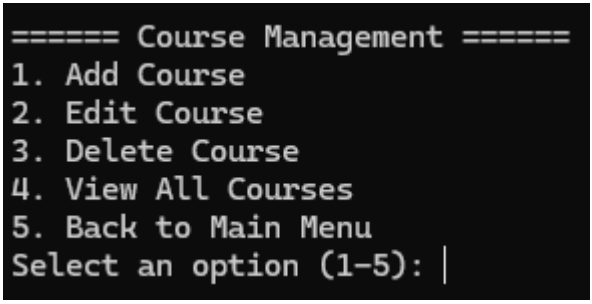

Figure 4 : Course Menu

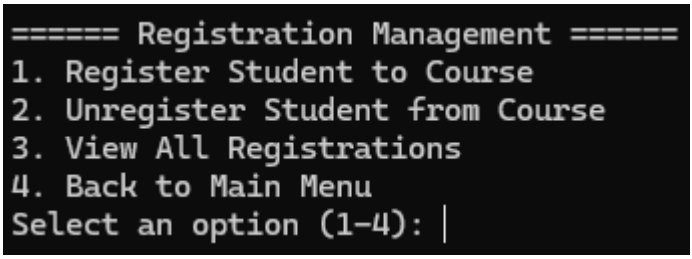

Figure 5 : Registration Menu

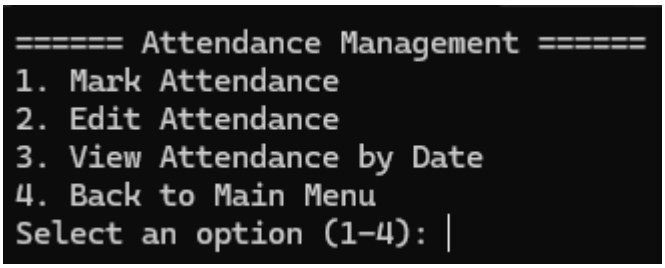

Figure 6 : Attendance Menu

P2Task3: Generate Scripts

1. Export the schema:
    - Use MySQL Workbench's **Data Export** tool to export the schema and save it as Lab2_Schema.sql.
2. Export the data:
    - Use the same tool to export the data and save it as Lab2_Data.sql.

Submission Requirements

- Submit the following files:
  - Dbscript.sql: Script for database schema and data.
  - .cs files for the CLI-based project.
  - No other files other than should be submitted

LO4: Learn how to set up and configure a Google Cloud Platform (GCP) account for database hosting.

1. Creating a GCP Account

1. Open your web browser and navigate to Google Cloud Platform.
2. Click on **"Sign In"** if you already have a Google account.

Figure 7 : Google Cloud Platform

3. Log in with your Gmail account credentials.
4. Again navigate to Google Cloud Platform. And click on Console

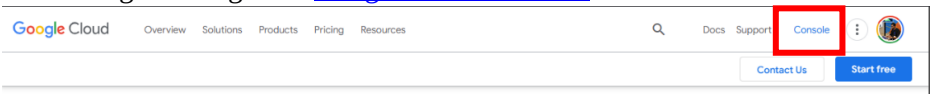

Figure 8 : GCP after user Login

5. This Window will open

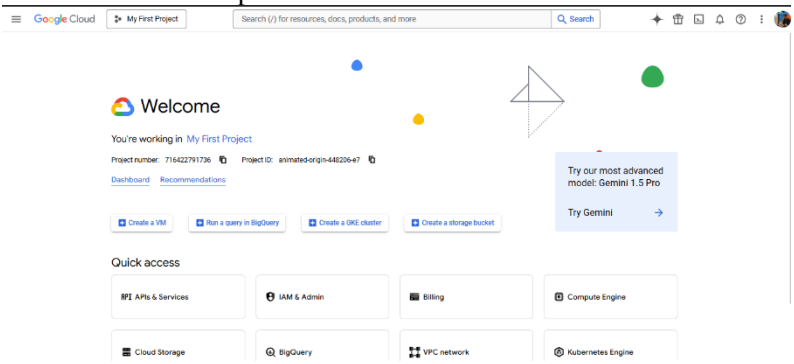

Figure 9 : GCP Console

2. Accessing BigQuery

1. In the search bar at the top, type **"BigQuery"** and click on the BigQuery option.

3. Exploring BigQuery Sandbox

1. BigQuery Sandbox allows you to use BigQuery without a billing account for free with certain limits.
2. Ensure that your project is selected in the top-left corner (create a new project if needed by clicking **"New Project"**).

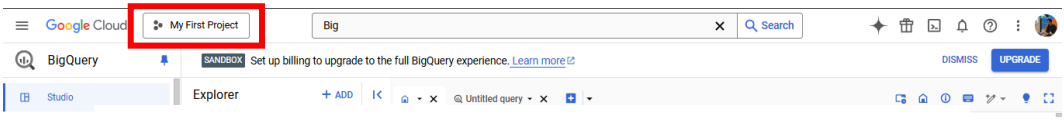

Figure 10 : Select Your Project

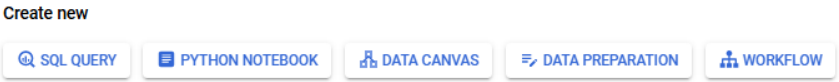

Figure 11 : BigQuery Studio

4. No setup is required to use the sandbox; you can start querying immediately.

4. Exploring Public Datasets

1. In the BigQuery console, go to the left-hand menu.

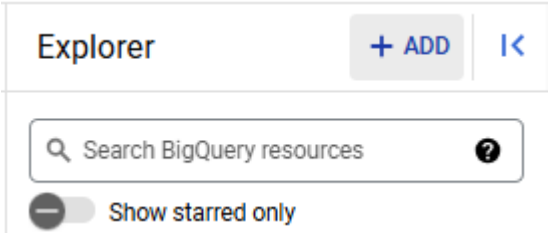

Figure 12 : BigQuery Explorer

2. Click on **"+ Add "** > Scroll down to Additional Sources and Click **"Public Datasets"**.
3. Browse through the available public datasets.

4. Search for "COVID-19 Public Datasets" and add them to your project.
5. After adding a dataset, click on it to explore the schema and table contents.
6. Write and run simple queries like:

```
SELECT *
FROM `bigquery-public-
data.covid19_open_data.covid19_open_data`
LIMIT 10;
```

5. Exploring the GitHub Dataset

1. In the public datasets, search for **"GitHub Repos"**.
2. Add the dataset to your project as you done before.
3. Open the dataset and explore its tables like contents, commits, or files.
4. Write queries to extract insights, for example:

```
SELECT author.name, COUNT(*) AS commit_count
FROM `bigquery-public-data.github_repos.commits`
GROUP BY author.name
ORDER BY commit_count DESC
LIMIT 10;
```

Submission Requirements

1. Word document named **"GCP.docx"** with your understanding of:
    - BigQuery
    - GitHub dataset insights
2. Screenshots of:
    - Your GCP account dashboard.
    - Queries written and their results.

Submission on Eduko

1. Zip all the files of your tasks:
    - LO1
    - LO3
    - LO4