МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

спеціальність 121 – Програмна інженерія

на тему: **Система контролю успішності студентів навчального закладу**
(назва теми)

Студентка групи КП-02 **Гаценко Марія Андріївна**
(ПІБ) _____
(підпис)

Викладач к.т.н **Радченко К. О.** _____
(підпис)

Захищено з оцінкою _____

Київ – 2021

АНОТАЦІЯ

Ця курсова робота є додатком, написаним на мові програмування C# 10, який створений для керування базою даних для моніторингу успішності студентів навчального закладу.

Додаток поділяється на дві частини: основний модуль консольного додатку для керування запитами до бази даних через консольний інтерфейс та допоміжний модуль, який направлений на статистичний аналіз даних за допомогою алгоритмів машинного навчання.

Як середовище розробки були використані Visual Studio та Visual Studio Code. В якості СУБД було вибрано PostgreSQL.

ЗМІСТ

ВСТУП

Ця курсова робота направлена на допомогу у взаємодії з базою даних для контролю успішності студентів навчального закладу. Мета роботи полягає у створенні додатку, який забезпечує створення, збір, маніпуляцію, валідацію та фільтрацію великих об’ємів інформації та допомагає користуватися нею.

Дана курсова робота є актуальною на момент її написання, адже наразі майже всі заклади вищої освіти користуються цифровим збереженням даних. Враховуючи, що ми живемо в цифровій епосі, можна з впевненістю сказати, що діджиталізація буде тільки зростати. Та й не можна забувати про епідеміологічну ситуацію у світі, через яку більшість закладів працюють в дистанційному режимі, через що можливість зберігати та маніпулювати даними в цифровому вигляді становить ще актуальнішою.

Для забезпечення цілісності бази даних було використано великий інструментарій, який включає в себе, як вбудовані засоби СУБД PostgreSQL, сторонні бібліотеки коду, сторонні програми для автоматичного резервування даних, так і перевірки всередині самого додатку. Це все було зроблено для можливості програми працювати з великими об’ємами даних, забезпечення можливості відновити базу у разі втрати даних та підвищення швидкодії бази даних.

1. Аналіз інструментарію для використання курсової роботи

Для даної курсової роботи була обрана реляційна СУБД PostgreSQL. На доцільність даного даної СУБД вказує велика кількість нормалізованих даних та відношень, невеликий пріоритет швидкості виконання запитів та потреба в аналізі даних, що потребує зручну та наочну маніпуляцією даними.
Мова програмування була вибрана C# на основі фреймворку .NET, так як вона має достатньо широкий інструментарій для виконання задач, які пов'язані з роботою з базами даних, машинним навчанням, візуалізації результатів тощо. Зокрема, були використані наступні бібліотеки:

- NpgSQL для взаємодії з PostgreSQL
- ScottPlot для створення візуалізації результатів, зокрема, графіків
- Microsoft.Ml та Microsoft.ML.FastTree для машинного навчання

2. Аналіз функціонування засобів резервування/відновлення бази даних

Як засіб резервування та відновлення бази даних був використаний додаток Effector Saver. Він дозволяє автоматично робити back-up бази даних у різні місця. Для даної курсової роботи, як сховище, було вибрано хмарове сховище Google Drive. База даних зберігається щодня о 1:00.

3. Структура бази даних

База даних, що була створена для оперування даними тематики курсової, складається з чотирьох таблиць:

1. Students – таблиця для зберігання інформації про студентів. Має наступні атрибути:
    - Name – ім'я студента
    - Surname – прізвище студента
    - Id – ідентифікаційний номер студента, який виступає Primary key та автоінкрементується.
2. Teachers – таблиця для зберігання інформації про викладачів. Має наступні атрибути:
    - Name – ім'я викладача
    - Surname – прізвище викладача
    - Id – ідентифікаційний номер викладача, який виступає Primary key та автоінкрементується.

3. Subjects – таблиця для зберігання інформації про предмети. Має наступні атрибути:
    - Id – ідентифікаційний номер предмета, який виступає Primary key та автоінкрементується.
    - Name – назва предмета
    - Teacher_id – Foreign key, який посилається на поле id таблиці Teachers
4. Grades – таблиця для зберігання оцінок студентів по тим чи іншим предметам. Має наступні атрибути:
    - Student_id - Foreign key, який посилається на поле id таблиці Students
    - Subject_id - Foreign key, який посилається на поле id таблиці Subjects
    - Grade – числове значення, яке виступає оцінкою студента, у межах від 1 (включно) до 100 (включно)

У додатку наведена ER-діаграма бази даних.

3. Опис програмного забезпечення

3.1 Загальна структура програмного забезпечення

Програмне забезпечення складається з трьох модулів: ConsoleApp для обробки запитів користувача та доступу до бази даних, бібліотека класів Entities: студентів, викладачів, оцінок та предметів, та модуль статистичного аналізу Analysis.

3.2 Опис модулів програмного забезпечення

1. ConsoleApp:
   Програма складається з наступних частин:
    - GradeRepo – клас для передачі запитів до таблиці grades бази даних та отримання від неї результатів
    - StudentRepo - клас для передачі запитів до таблиці students бази даних та отримання від неї результатів
    - TeacherRepo - клас для передачі запитів до таблиці teachers бази даних та отримання від неї результатів
    - SubjectRepo - клас для передачі запитів до таблиці subjects бази даних та отримання від неї результатів
    - Program – основний вхід в програму, представляє собою консольний інтерфейс та обробник запитів користувача
    - Graph – клас для створення графіку оцінок певного студента

- StudentLogin – клас, який дозволяє студенту зробити наступні запити до бази даних: подивитися всі свої оцінки, подивитися оцінки по конкретному предмету, подивитися всі предмети, які студент вивчає, дізнатися середній бал.
- TecherLogin - клас, який дозволяє викладачу зробити наступні запити до бази даних: поставити оцінку студенту, подивитися всі предмети, які він викладає.

2. Entities

   Бібліотека складається з наступних частин:

   - Grade – клас, який представляє собою об’єкт таблиці grades
   - Person – абстрактний клас, від якого наслідуються класи Teacher та Student
   - Teacher - клас, який представляє собою об’єкт таблиці teachers
   - Student - клас, який представляє собою об’єкт таблиці students
   - Subject - клас, який представляє собою об’єкт таблиці subjects

3. Analysis

   Програма складається з наступних частин:

   - Grade – клас, в який зчитується csv файл для тренування компьютеру для подальшого статистичного аналізу
   - GradePrediction – допоміжний клас для проведення статистичного аналізу
   - AnalysisProgram – вхідна точка програми, яка проводить аналіз

4. Аналіз функціонування засобів реплікації

У процесі виконання курсової роботи мала бути використана можливість реплікації даних. Була обрана логічна реплікація в межах однієї локальної мережі. З технічних причин, даний функціонал не був реалізований. Нижче приведений алгоритм дій.

«ALTER SYSTEM SET wal_level = logical;»

1. Перезавантажити сервер.
2. «SHOW wal_level»
3. «SELECT * FROM pg_create_logical_replication_slot('slotName', 'pgoutput');»
4. Створення публікації в цій базі даних
5. Створення БД для реплікації, створення відповідних таблицб з відповідними колонками (копії основних таблиць)
6. У новій БД створюємо підписку на публікацію, як слот вказуємо

'slotName'.

5. Аналіз функціонування засобів резервування/відновлення бази даних

Як засіб резервування та відновлення бази даних був використаний додаток Effector Saver. Він дозволяє автоматично робити back-up бази даних у різні місця. Для даної курсової роботи, як сховище, було вибрано хмарове сховище Google Drive. База даних зберігається щодня о 1:00.

6. Аналіз результатів підвищення швидкодії виконання запитів

Підвищення швидкодії досягалось використанням індексів для наступних таблиць:

- Grades: індекс для ідентифікаторів студентів та індекс для ідентифікаторів предметів
- Students: індекс для імен та індекс для прізвищ
- Teachers: індекс для імен та індекс для прізвищ

Результати підвищення швидкодії наведені у додатку.

7. Опис результатів аналізу предметної галузі

Аналіз проводився щодо оцінок, які, скоріш за все, отримає студент з того чи іншого предмету. Для цього використовувались бібліотеки Microsoft.ML та Microsoft.ML.FastTree. Як алгоритм аналізу біла використана лінійна регресія. Результати аналізу у вигляді графіка наведені у додатку.

ВИСНОВКИ

Виконуючи дану курсову роботу, було створено базу даних та програмне забезпечення для її використання та контролю. При створенні бази було проаналізовано, які показники мають значення для системи контрою успішності студентів. Для симуляцію великої кількості таких даних було створено відповідні SQL-запити, які генерують рандомізовані дані.

Задля зручності керування СУБД було використано сторонній додаток, він спрощує резервацію даних, та забезпечує безпечне та автоматичне створення Back-up.

За для підвищення швидкодії, була використана індексація, що допомогло значно пришвидшити отримання відповідей СУБД на запити.

Отже, по завершенню виконання роботи, можна сказати, що початкова мета проєкту – створення системи для контролю успішності студентів вищого навчального закладу, була досягнута з використанням СУБД PostgreSQL та мови програмування C# 10 у середовищі Visual Studio та Visual Studio Code, що дозволяє роботі використовуватися за призначенням у своїй предметній галузі.

ДОДАТКИ

Додатки А.

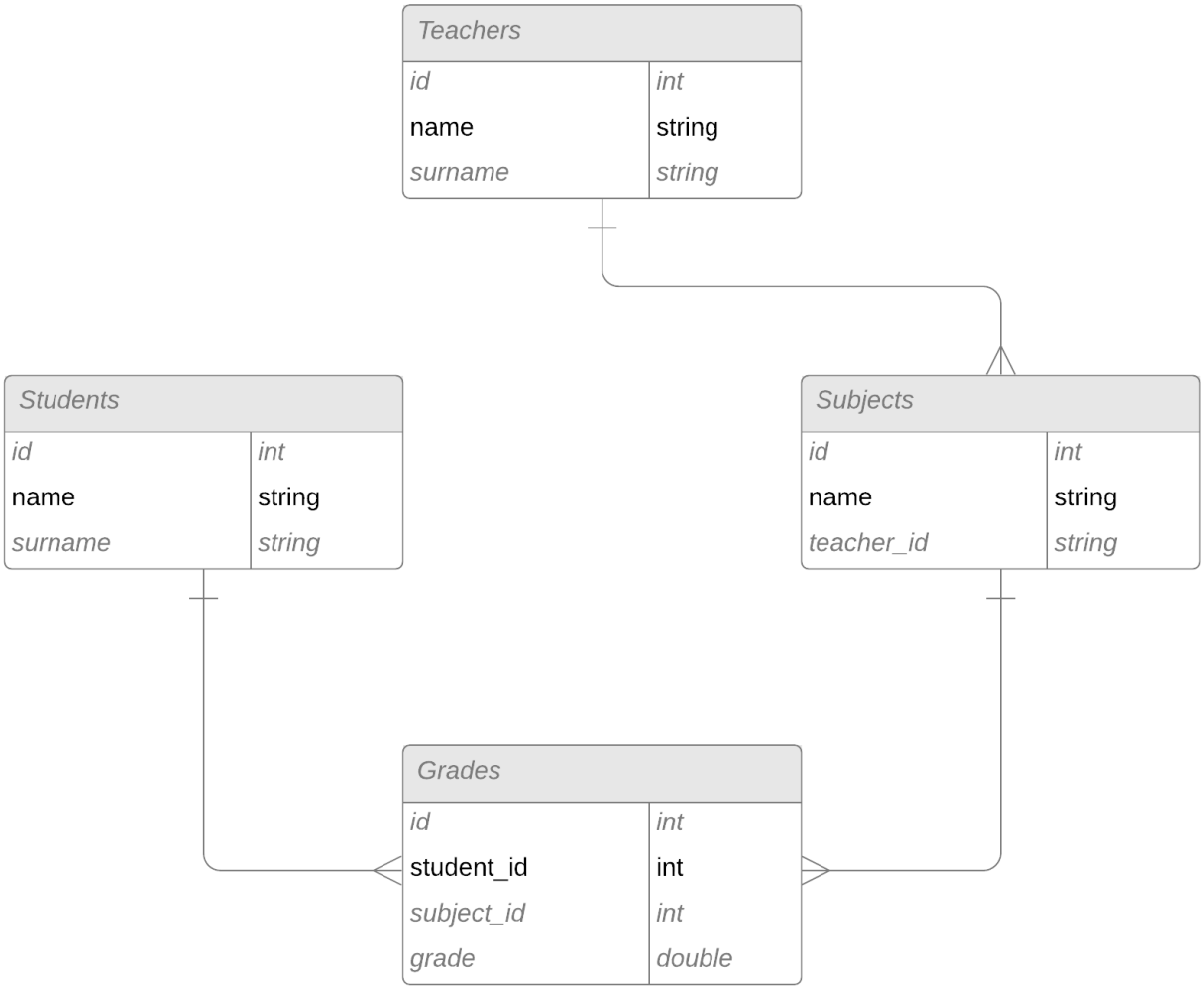

Рис 1. ER-діаграма бази даних

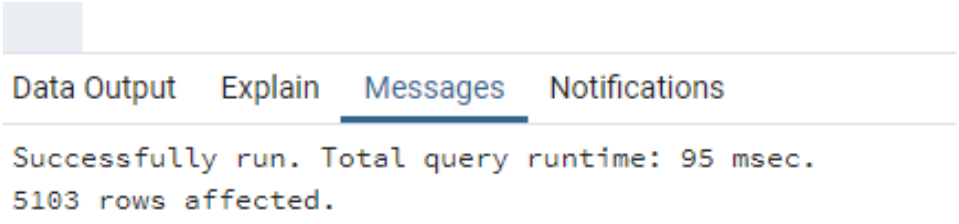

Рис 2. Час виконання запиту 1 до таблиці grades до індексів

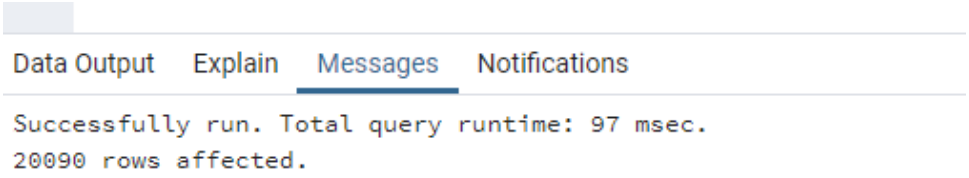

Рис 3. Час виконання запиту 2 до таблиці grades до індексів

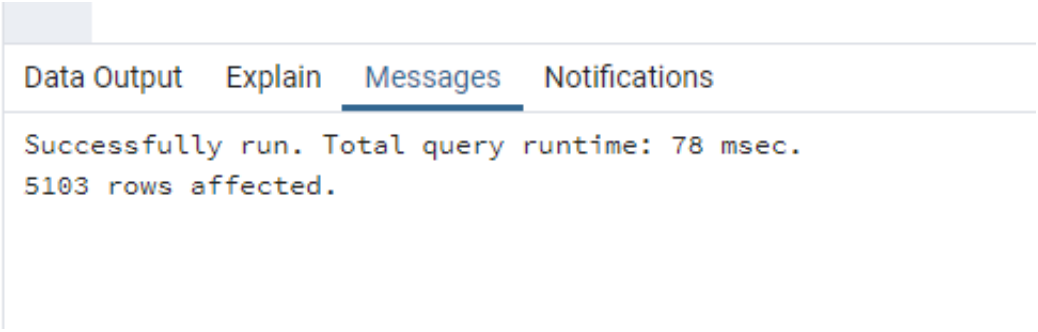

Рис 4. Час виконання запиту 1 до таблиці grades після індексів

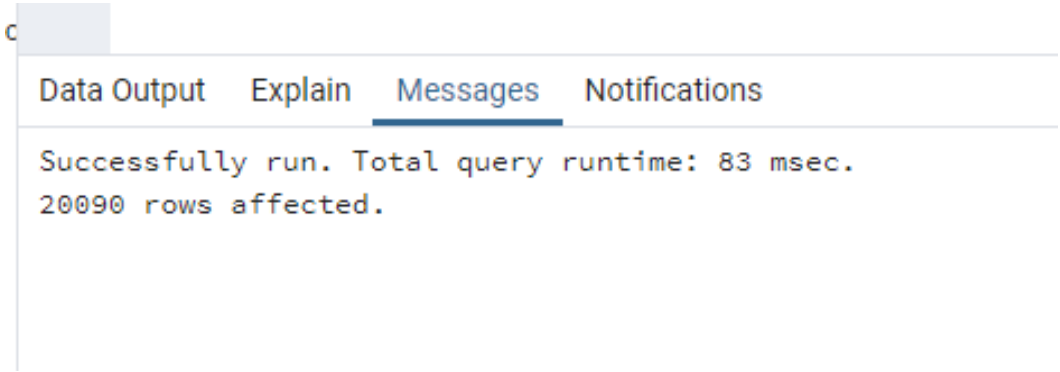

Рис 5. Час виконання запиту 2 до таблиці grades після індексів

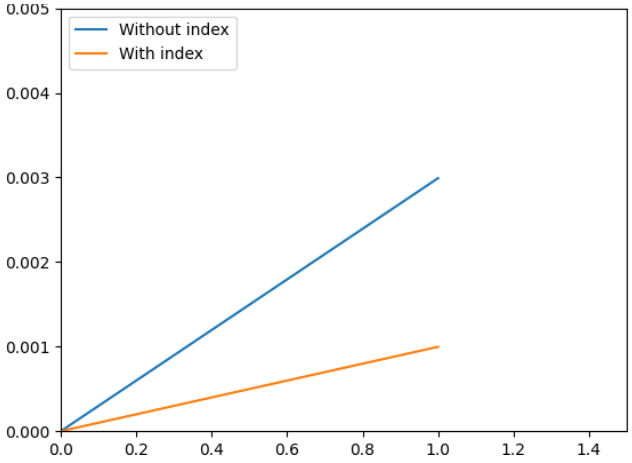

Рис 6. Графік порівння часу виконання запиту з індексами

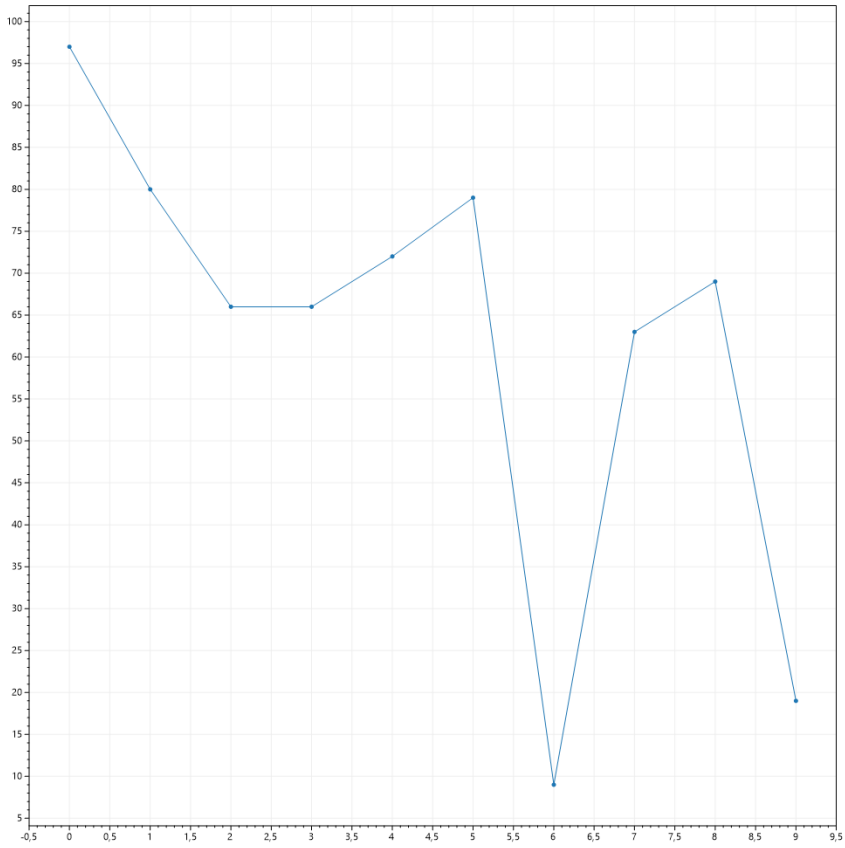

Рис 7. Графік оцінок студента до статистичного аналізу

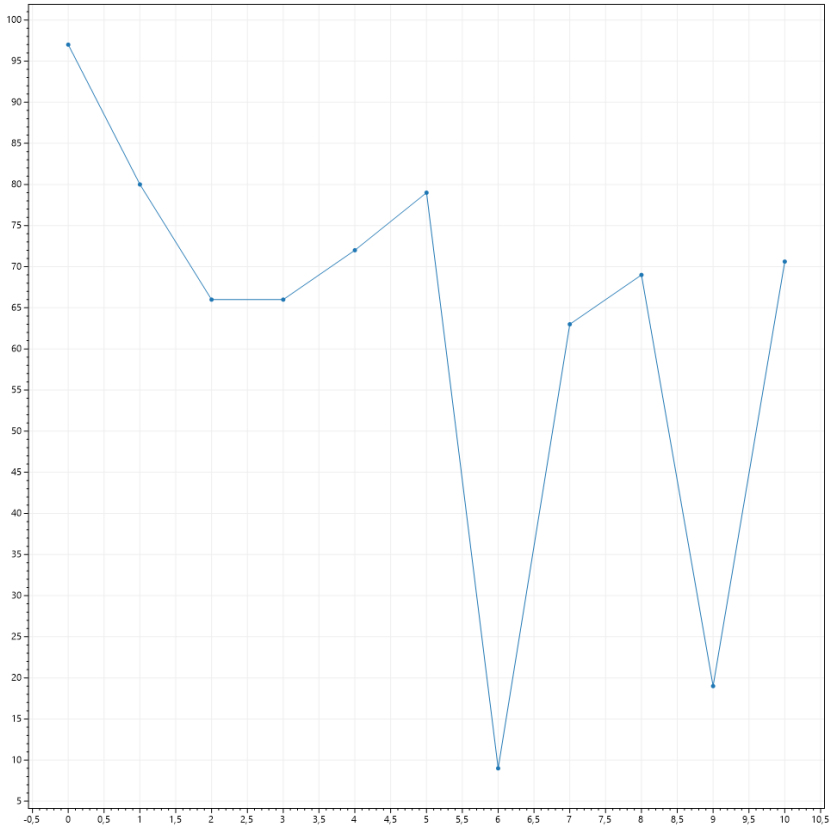

Рис 8. Графік оцінок студента після статистичного аналізу та передбачення

Додатки Б.

Приклади SQL запитів:

```
public Student GetById(int id)
        {
            string command = $"SELECT * FROM students WHERE id = {id}";
            NpgsqlCommand cmd = new NpgsqlCommand(command, connection);

            NpgsqlDataReader reader = cmd.ExecuteReader();
            Student st = new Student();

            if (reader.Read())
            {
                st.id = reader.GetInt32(2);
                st.name = reader.GetString(0);
                st.surname = reader.GetString(1);
            }
            reader.Close();
            return st;
        }

public bool Insert(Student st)
        {
            string command = $"INSERT INTO students (name, surname) VALUES ('{st.name}',
'{st.surname}');";

            NpgsqlCommand cmd = new NpgsqlCommand(command, connection);
            if (cmd.ExecuteScalar() == null) return false;
            return true;
        }

public bool DeleteById(int id)
        {
            string command = $"DELETE FROM students WHERE id = {id};";

            NpgsqlCommand cmd = new NpgsqlCommand(command, connection);

            GradeRepo gradeRepo = new GradeRepo(connection);
            gradeRepo.DeleteByStudentId(id);

            int res = cmd.ExecuteNonQuery();

            if (res == -1) return false;
            return true;
```

```
        }
```

Класи сутностей:

```
 public class Grade
    {
        public double grade;
        public int studentId;
        public int subjectId;
        public Grade() : base() { }
        public Grade(int grade, int studentId, int subjectId)
        {
            this.subjectId = subjectId;
            this.studentId = studentId;
            this.grade = grade;
        }
    }
public abstract class Person
    {
        public int id;
        public string name;
        public string surname;
    }
public class Student : Person
    {
        public Student() : base() { }
        public Student(int id, string name, string surname)
        {
            this.id = id;
            this.name = name;
            this.surname = surname;
        }
    }
public class Subject
    {
        public int id;
        public string name;
        public int teacherId;
        public Subject() : base() { }
        public Subject(int id, string name, int teacherId)
        {
            this.id = id;
            this.name = name;
            this.teacherId = teacherId;
        }
```

```
    }
public class Teacher : Person
    {
        public Teacher() : base() { }
        public Teacher(int id, string name, string surname)
        {
            this.id = id;
            this.name = name;
            this.surname = surname;
        }
    }
```

Модуль аналізу даних:

```
public class AnalysisProgram
    {
        static readonly string _trainDataPath = "./TrainingData/GradesTableTrain.csv";
        static readonly string _testDataPath = "./TrainingData/GradesTableTest.csv";
        static readonly string _modelPath = "./Analysis/TrainingData/Model.zip";
        static void Main(string[] args)
        {

        }
        public static float Predict(Grade gradeSample)
        {
            MLContext mlContext = new(seed: 0);

            var model = Train(mlContext, _trainDataPath);

            Evaluate(mlContext, model);

            return TestSinglePrediction(mlContext, model, gradeSample);
        }
        public static ITransformer Train(MLContext mlContext, string dataPath)
        {
            IDataView dataView = mlContext.Data.LoadFromTextFile<Grade>(dataPath,
hasHeader: true, separatorChar: ',');

            var pipeline = mlContext.Transforms.CopyColumns(outputColumnName: "Label",
inputColumnName: "grade")
                .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName
: "studentIdEncoded", inputColumnName: "studentId"))
                .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName:
"subjectIdEncoded", inputColumnName: "subjectId"))
```

```
                .Append(mlContext.Transforms.Concatenate("Features", "studentIdEncoded",
"subjectIdEncoded"))
                .Append(mlContext.Regression.Trainers.FastTree());

            var model = pipeline.Fit(dataView);

            return model;
        }
        private static void Evaluate(MLContext mlContext, ITransformer model)
        {
            IDataView dataView = mlContext.Data.LoadFromTextFile<Grade>(_testDataPath,
hasHeader: true, separatorChar: ',');
            var predictions = model.Transform(dataView);
            var metrics = mlContext.Regression.Evaluate(predictions, "Label", "Score");
        }
        private static float TestSinglePrediction(MLContext mlContext, ITransformer
model, Grade gradeSample)
        {
            var predictionFunction = mlContext.Model.CreatePredictionEngine<Grade,
GradePrediction>(model);

            var prediction = predictionFunction.Predict(gradeSample);

            Console.WriteLine($"Predicted grade is: {prediction.grade:0.####}");
            return prediction.grade;
        }
    }
```

Генерація графіку:

```
internal class Graph
    {
        public static void CreateGradeGraph(double[] grades, string filePath)
        {
            List<double> list = new List<double>();
            for(int i = 0; i < grades.Length; i++)
            {
                list.Add(i);
            }

            double[] dataX = list.ToArray();
            double[] dataY = grades;

            var plt = new Plot(1000, 1000);
            plt.AddScatter(dataX, dataY);
```

```
            plt.SaveFig(filePath);
        }
    }
```

Псевдовипадкова генерація даних:

```
static void DataGen(int number)
        {
            WriteLine("What entities would you like to generate?");
            string command = ReadLine();

            if (command == "students")
            {
                for (int i = 0; i < number; i++)
                {
                    string allNames =
File.ReadAllText("C:/Users/User/projects/databaseCW/EntitiesData/names.txt");
                    string allSurnames =
File.ReadAllText("C:/Users/User/projects/databaseCW/EntitiesData/surnames.txt");

                    string[] arrNames = allNames.Split("\r\n");
                    string[] arrSurnames = allSurnames.Split("\r\n");

                    Random r = new Random();

                    string name = arrNames[r.Next(0, arrNames.Length)];
                    string surname = arrSurnames[r.Next(0, arrSurnames.Length)];

                    studentRepo.Insert(new Student(1, name, surname));
                }
            }
            else if (command == "teachers")
            {
                for (int i = 0; i < number; i++)
                {
                    string allNames =
File.ReadAllText("C:/Users/User/projects/databaseCW//EntitiesData/names.txt");
                    string allSurnames =
File.ReadAllText("C:/Users/User/projects/databaseCW/EntitiesData/surnames.txt");

                    string[] arrNames = allNames.Split("\r\n");
                    string[] arrSurnames = allSurnames.Split("\r\n");

                    Random r = new Random();
```

```
                    string name = arrNames[r.Next(0, arrNames.Length)];
                    string surname = arrSurnames[r.Next(0, arrSurnames.Length)];

                    teacherRepo.Insert(new Teacher(1, name, surname));
                }
            }
            else if (command == "grades")
            {
                for (int i = 0; i < number; i++)
                {
                    Random r = new();

                    int perc = r.Next(0, 101);
                    int mark;

                    if (perc < 10)
                    {
                        mark = r.Next(0, 60);
                    }
                    else if (perc < 90)
                    {
                        mark = r.Next(60, 90);
                    }
                    else
                    {
                        mark = r.Next(90, 101);
                    }

                    gradeRepo.Insert(new Grade(mark, studentRepo.GetRandomStudent().id,
subjectRepo.GetRandomSubject().id));
                }
            }
            else if (command == "subjects")
            {
                for (int i = 0; i < number; i++)
                {
                    string allSubjects =
File.ReadAllText("C:/Users/User/projects/databaseCW/EntitiesData/subjects.txt");
                    string[] arrSubjects = allSubjects.Split("\r\n");

                    Random r = new();

                    string subject = arrSubjects[r.Next(0, arrSubjects.Length)];
```

```
                Teacher t = teacherRepo.GetRandomTeacher();

                subjectRepo.Insert(new Subject(0, subject, t.id));
            }
        }
        else
        {
            WriteLine("Invalid input");
            return;
        }
        WriteLine("Done!");
    }
```