НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

(назва дисципліни)

на тему: База даних з результатами зовнішнього незалежного оцінювання

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_\_2\_\_ курсу \_ІП-13\_\_\_\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_Шевцової Анастасії Андріївни\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_Ліщук Олександр Васильович\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс \_\_2\_\_\_ Група \_\_ІП-13\_\_ Семестр \_\_\_3\_\_

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шевцовій Анастасії Андріївні\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи: База даних з результатами зовнішнього незалежного оцінювання

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи Ліщук Олександр Васильович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи \_\_\_\_\_10.01.2023\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_завдання на розробку бази даних для зберігання результатів зовнішнього незалежного оцінювання\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у форматі обраної системи управління базою даних

5) Створення користувачів бази даних

6) Імпорт даних з використанням засобів СУБД в створену базу даних

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання\_\_\_\_08.11.2022**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища | 12.11.2022 |  |
| 2 | Побудова ER-моделі | 10.01.2023 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 10.01.2023 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 10.01.2023 |  |
| 5 | Створення користувачів бази даних | 10.01.2023 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 10.01.2023 |  |
| 7 | Створення мовою SQL запитів | 10.01.2023 |  |
| 8 | Оптимізація роботи запитів | 10.01.2023 |  |
| 9 | Оформлення пояснювальної записки | 10.01.2023 |  |
| 10 | Захист курсової роботи | 10.01.2022 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Шевцова А. А.\_\_\_\_\_\_\_

(підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Ліщук О.В.**\_\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**ЗМІСТ**

[ВСТУП 4](#_Toc124108007)

[1. ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА 5](#_Toc124108008)

[2. ПОСТАНОВКА ЗАВДАННЯ 6](#_Toc124108009)

[3. ПРОЕКТУВАННЯ БАЗИ ДАНИХ 7](#_Toc124108010)

[3.1. Побудова ER-моделі 7](#_Toc124108011)

[3.1.1. Бізнес-правила 7](#_Toc124108012)

[3.1.2. Вибір сутностей 7](#_Toc124108013)

[3.1.3. Набори атрибутів сутностей 8](#_Toc124108014)

[3.2. Даталогічна модель бази даних 11](#_Toc124108015)

[4. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ 13](#_Toc124108016)

[4.1. Створення бази даних у форматі СУБД MS SQL 13](#_Toc124108017)

[4.1.1. Створення таблиць 13](#_Toc124108018)

[4.1.2. Створення зав’язків 14](#_Toc124108019)

[4.1.3. Створення обмежень 15](#_Toc124108020)

[4.2. Імпортування даних у таблиці 15](#_Toc124108021)

[5. РОБОТА З БАЗОЮ ДАНИХ 17](#_Toc124108022)

[5.1. Створення користувачів 17](#_Toc124108023)

[5.1.1. Учасник ЗНО 17](#_Toc124108024)

[5.1.2. Перевіряючий 17](#_Toc124108025)

[5.2. SQL запити 17](#_Toc124108026)

[5.2.1. Генератори 17](#_Toc124108027)

[5.2.2. Збережені процедури та функції 18](#_Toc124108028)

[5.2.3. Тригери 19](#_Toc124108029)

[5.2.4. Представлення 20](#_Toc124108030)

[5.2.5. SQL-запити 20](#_Toc124108031)

[5.2.6. Індекси та їхній приклад роботи 29](#_Toc124108032)

[ВИСНОВКИ 30](#_Toc124108033)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 31](#_Toc124108034)

# ВСТУП

Для оптимальної організації даних використовують бази даних. Замість величезних архівів з купою паперів зараз використовують цифрові бази даних, які зберігають дані організовано в таблицях, що пришвидшує пошук потрібної інформації та її обробку.

Дана тема є актуальною, оскільки система зовнішнього незалежного оцінювання уніфіковує результати вступних випробувань, що створює рівні умови доступу до освіти для всіх, незалежно від матеріальних можливостей. А також запобігає корупції на локальному рівні та наближує українську систему освіти до європейських стандартів.

Метою курсової роботи є створення бази даних з результатами зовнішнього незалежного оцінювання, що спростить доступ до даних ЗНО, полегшить процес створення звітів та проведення аналізу.

1. ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Зовнішнє оцінювання проводиться з метою забезпечення прав осіб на рівний доступ до вищої освіти та оцінювання відповідності результатів навчання, здобутих на основі повної загальної середньої освіти, державним вимогам[1].

Кожен учасник ЗНО має інформаційну картку з результатами, яка має унікальний номер, PIN-код, рік отримання та номер Сертифіката зовнішнього незалежного оцінювання.

Дані в інформаційній картці завантажуються з відомості результатів зовнішнього незалежного оцінювання, яка включає номер облікового запису, предмет, кількість тестових балів, результат ЗНО за шкалою 100-200 балів та результат ДПА за шкалою 1-12 балів.

Кожен предмет має назву, максимальну кількість балів, яку можливо набрати, і пов’язаний з таблицею переведення тестових балів ЗНО.

В інформації про учасника ЗНО вказано його прізвище та ініціали, дані про те чи брав участь в тестуванні, місце проходження тестування та вказана сесія ЗНО(основна, додаткова, спеціальна). Місце проходження тестування включає назву області, району та населеного пункту.

За результатами проходження ЗНО формується регіональна статистика. Вона відображає кількість осіб, які зареєструвалися для проходження тестування та ти, які взяли участь у тестуванні. А також показує середній бал ЗНО в даному регіоні.

1. ПОСТАНОВКА ЗАВДАННЯ

Метою даної роботи є розробка бази даних з результатами зовнішнього незалежного оцінювання. Тобто організація даних таким чином, щоб доступ до результатів ЗНО та оперування даними було швидким ти ефективним. Отже основні задачі:

Учасник ЗНО повинен мати змогу:

* Зареєструватися до участі у ЗНО
* Переглядати результати ЗНО

Для отримання результатів перевіряючий повинен мати змогу:

* Виставляти бали
* Переглядати дані учасників

Результати іспитів мають автоматично переводитися у шкалу ЗНО 100-200 балів та шкалу ДПА 1-12 балів.

За результатами проведення ЗНО має формуватися загальнодоступна статистика за регіонами.

1. ПРОЕКТУВАННЯ БАЗИ ДАНИХ
   1. Побудова ER-моделі

Після аналізу було виділено такі сутності та зв’язки між ними:

* + 1. Бізнес-правила
* Бали ЗНО виставляються по шкалі 100-200
* Бали ДПА виставляються по шкалі 1-12
  + 1. Вибір сутностей
* Учасник ЗНО
* Предмет
* Відомість результатів ЗНО
* Інформаційна картка
* Шкала переведення балів ЗНО та ДПА
* Локація екзамену
* Сесія
* Сертифікат
* Статистика результатів
  + 1. Набори атрибутів сутностей

Таблиця 3.1 – Сутності та їхні атрибути

|  |  |
| --- | --- |
| Сутність | Атрибут |
| Student | student\_id  full\_name  info\_card\_id  participated  location\_id |
| InformationCard | info\_card\_id  pin\_code  year\_of\_receipt  certificate\_id |
| ResultInfo | res\_info\_id  subject\_id  test\_score  result  sfe  session\_name  info\_card\_id |
| Certificate | certificate\_id  subjects |
| Subject | subject\_id  subject\_name  max\_test\_score |
| ScoreScale | score\_scale\_id  subject\_id  test\_score  rating\_score  sfe |
| Session | session\_name |
| TestingLocation | location\_id  region  district  settlement |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| Stats | stats\_id  location\_id  registered\_num  participated\_num  average\_score |

Сутність Student пов’язана з сутністю InformationCard зв’язком один до одного, бо в кожного учасника є одна інформаційна картка.

Сутність Student пов’язана з сутністю TestingLocation зв’язком багато до одного, бо в одному місці можуть складати ЗНО декілька учасників.

Сутність InformationCard пов’язана з сутністю Certificate зв’язком один до одного, бо до кожної інформаційної картки прикріплений сертифікат з відповідним номером.

Сутність ResultInfo пов’язана з сутністю Subject зв’язком один до багатьох, бо по кожному предмету може бути багато результатів.

Сутність ResultInfo пов’язана з сутністю Session зв’язком один до багатьох, бо по кожній сесії може бути багато результатів.

Сутність ResultInfo пов’язана з сутністю InformationCard зв’язком один до багатьох, бо в кожній інформаційній картці може бути багато результатів.

Сутність ScoreScale пов’язана з сутністю Subject зв’язком багато до одного, бо до кожного предмета є відповідна шкала.

Сутність Stats пов’язана з сутністю TestingLocation зв’язком один до одного, бо статистика формується по кожній локації.

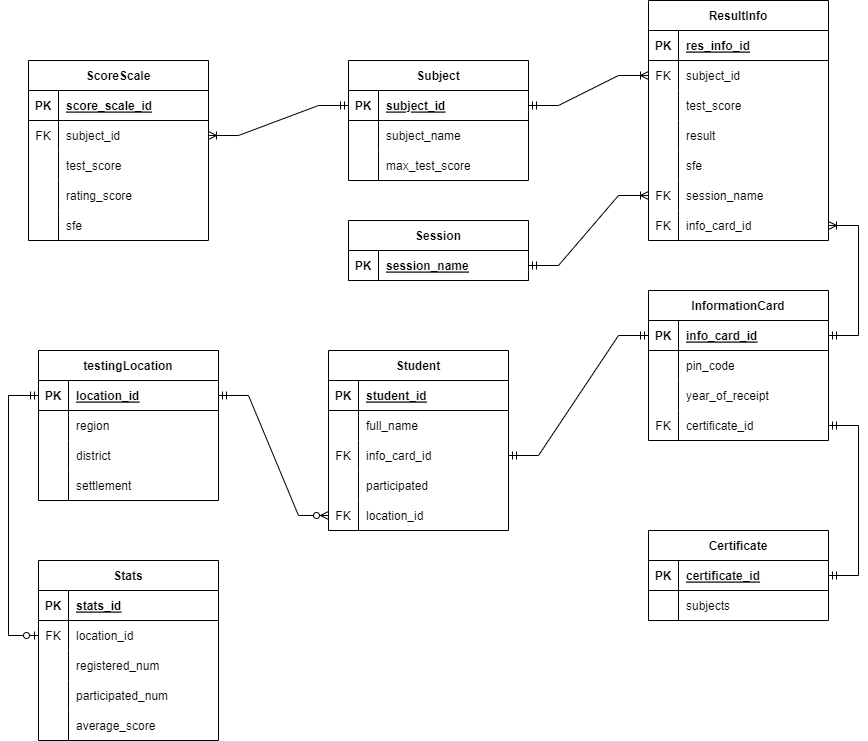


Рисунок 3.1 – ER-діаграма

* 1. Даталогічна модель бази даних
     1. Побудова необхідних відношень та визначення первинних і зовнішніх ключів

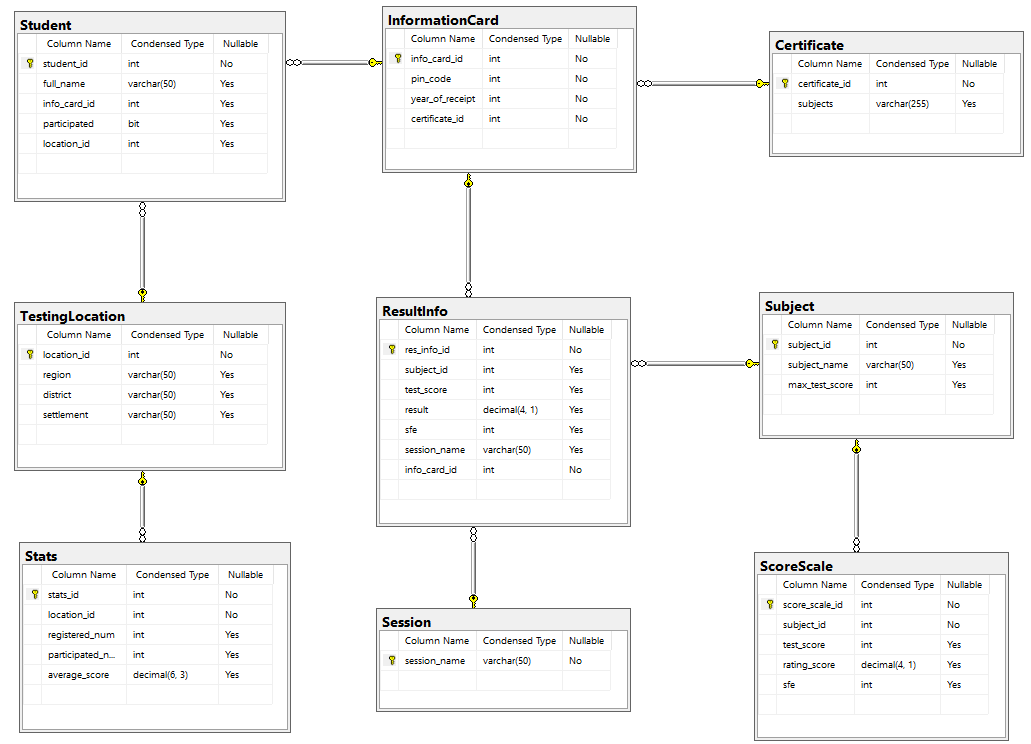


Рисунок 3.2 – Даталогічна модель бази даних

На даній схемі видно, що база даних знаходиться у третій нормальній формі, адже всі поля таблиць декомпозовані та всі атрибути таблиць функціонально повно залежать від первинного ключа, кожен неключовий атрибут не є транзитивно залежним від первинного ключа.

* Визначення обмежень цілісності для спроектованих відношень. Обмеження цілісності: Рядок батьківської таблиці може бути видалений лише у тому випадку, якщо немає зовнішніх ключів, що посилаються на значення преференційного ключа цього рядка. Реалізовано відсутністю параметра ON DELETE {CASCADE|SET NULL} при створенні таблиці, що за умовчанням значитиме ON DELETE RESTRICT;
* Обов’язкові атрибути таблиць мають обмеження NOT NULL, для запобіганню помилок при роботі з даними.

1. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ
   1. Створення бази даних у форматі СУБД MS SQL
      1. Створення таблиць

IF OBJECT\_ID ('dbo.Student', 'U') IS NOT NULL

DROP TABLE Student;

GO

CREATE TABLE Student

(

student\_id INT IDENTITY(1,1) NOT NULL,

full\_name VARCHAR(50),

info\_card\_id INT,

participated BIT,

location\_id INT,

CONSTRAINT PK\_Student\_student\_id PRIMARY KEY (student\_id)

);

IF OBJECT\_ID ('dbo.InformationCard', 'U') IS NOT NULL

DROP TABLE InformationCard;

GO

CREATE TABLE InformationCard

(

info\_card\_id INT IDENTITY(1,1) NOT NULL,

pin\_code INT NOT NULL,

year\_of\_receipt INT NOT NULL,

certificate\_id INT NOT NULL,

CONSTRAINT PK\_InformationCard\_info\_card\_id PRIMARY KEY (info\_card\_id)

);

IF OBJECT\_ID ('dbo.ResultInfo', 'U') IS NOT NULL

DROP TABLE ResultInfo;

GO

CREATE TABLE ResultInfo

(

res\_info\_id INT IDENTITY(1,1) NOT NULL,

subject\_id INT,

test\_score INT,

result DECIMAL(4,1),

sfe INT,

session\_name VARCHAR(50),

info\_card\_id INT NOT NULL,

CONSTRAINT PK\_ResultInfo\_res\_info\_id PRIMARY KEY (res\_info\_id)

);

IF OBJECT\_ID ('dbo.Certificate', 'U') IS NOT NULL

DROP TABLE "Certificate";

GO

CREATE TABLE "Certificate"

(

certificate\_id INT IDENTITY(1,1) NOT NULL,

subjects VARCHAR(255),

CONSTRAINT PK\_Certificate\_certificate\_id PRIMARY KEY (certificate\_id)

);

IF OBJECT\_ID ('dbo.Subject', 'U') IS NOT NULL

DROP TABLE "Subject";

GO

CREATE TABLE "Subject"

(

subject\_id INT IDENTITY(1,1) NOT NULL,

subject\_name VARCHAR(50),

max\_test\_score INT,

CONSTRAINT PK\_Subject\_subject\_id PRIMARY KEY (subject\_id)

);

IF OBJECT\_ID ('dbo.ScoreScale', 'U') IS NOT NULL

DROP TABLE ScoreScale;

GO

CREATE TABLE ScoreScale

(

score\_scale\_id INT IDENTITY(1,1) NOT NULL,

subject\_id INT NOT NULL,

test\_score INT,

rating\_score DECIMAL(4,1),

sfe INT,

CONSTRAINT PK\_ScoreScale\_score\_scale\_id PRIMARY KEY (score\_scale\_id)

);

IF OBJECT\_ID ('dbo.Session', 'U') IS NOT NULL

DROP TABLE "Session";

GO

CREATE TABLE "Session"

(

session\_name VARCHAR(50) NOT NULL,

CONSTRAINT PK\_Session\_session\_name PRIMARY KEY (session\_name)

);

IF OBJECT\_ID ('dbo.TestingLocation', 'U') IS NOT NULL

DROP TABLE TestingLocation;

GO

CREATE TABLE TestingLocation

(

location\_id INT IDENTITY(1,1) NOT NULL,

region VARCHAR(50),

district VARCHAR(50),

settlement VARCHAR(50),

CONSTRAINT PK\_TestingLocation\_location\_id PRIMARY KEY (location\_id)

);

IF OBJECT\_ID ('dbo.Stats', 'U') IS NOT NULL

DROP TABLE "Stats";

GO

CREATE TABLE "Stats"

(

stats\_id INT IDENTITY(1,1) NOT NULL,

location\_id INT NOT NULL,

registered\_num INT,

participated\_num INT,

average\_score DECIMAL(6,3),

CONSTRAINT PK\_Stats\_stats\_id PRIMARY KEY (stats\_id)

);

* + 1. Створення зав’язків

ALTER TABLE Student

ADD CONSTRAINT FK\_Student\_info\_card\_id FOREIGN KEY (info\_card\_id)

REFERENCES InformationCard (info\_card\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_Student\_location\_id FOREIGN KEY (location\_id)

REFERENCES TestingLocation (location\_id)

ON DELETE SET NULL

ON UPDATE CASCADE;

ALTER TABLE InformationCard

ADD CONSTRAINT FK\_InformationCard\_certificate\_id FOREIGN KEY (certificate\_id)

REFERENCES "Certificate" (certificate\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE ResultInfo

ADD CONSTRAINT FK\_ResultInfo\_subject\_id FOREIGN KEY (subject\_id)

REFERENCES "Subject" (subject\_id)

ON DELETE SET NULL

ON UPDATE CASCADE,

CONSTRAINT FK\_ResultInfo\_info\_card\_id FOREIGN KEY (info\_card\_id)

REFERENCES InformationCard (info\_card\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_ResultInfo\_session\_name FOREIGN KEY (session\_name)

REFERENCES "Session" (session\_name)

ON DELETE CASCADE

ON UPDATE CASCADE,

CHECK (result >= 100 AND result <= 200),

CHECK (sfe >= 1 AND sfe <= 12);

ALTER TABLE ScoreScale

ADD CONSTRAINT FK\_ScoreScale\_subject\_id FOREIGN KEY (subject\_id)

REFERENCES "Subject" (subject\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CHECK (rating\_score >= 100 AND rating\_score <= 200),

CHECK (sfe >= 1 AND sfe <= 12);

ALTER TABLE "Stats"

ADD CONSTRAINT FK\_Stats\_location\_id FOREIGN KEY (location\_id)

REFERENCES TestingLocation (location\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

* + 1. Створення обмежень

Обмеження для таблиці ResultInfo:

CHECK (result >= 100 AND result <= 200),

CHECK (sfe >= 1 AND sfe <= 12);

Обмеження для таблиці ScoreScale:

CHECK (rating\_score >= 100 AND rating\_score <= 200),

CHECK (sfe >= 1 AND sfe <= 12);

* 1. Імпортування даних у таблиці

Імпортувати дані у вибрану мною СУБД можна двома способами:

1. Імпортувати дані з файлу csv формату;
2. Імпортувати дані із файлу-скрипта sql формату.

Для першого методу створено .csv файли для кожної таблиці.

Для другого методу створено файл DownloadData.sql. Його вміст:

INSERT INTO [dbo].[Stats] (location\_id, [registered\_num])

SELECT [dbo].[TestingLocation].[location\_id], COUNT([dbo].[Student].[student\_id])

FROM [dbo].[TestingLocation] JOIN [dbo].[Student]

ON [dbo].[TestingLocation].location\_id = [dbo].[Student].location\_id

GROUP BY [dbo].[TestingLocation].[location\_id];

UPDATE [dbo].[Stats]

SET [participated\_num] = (SELECT COUNT([student\_id]) FROM [dbo].[Student]

WHERE ([dbo].[Stats].location\_id = [dbo].[Student].location\_id) AND [dbo].[Student].participated = 1);

UPDATE [dbo].[Stats]

SET [average\_score] = (SELECT AVG([result]) FROM

(SELECT [dbo].[Student].location\_id, [dbo].[ResultInfo].result

FROM [dbo].[Student] JOIN [dbo].[ResultInfo]

ON [dbo].[Student].info\_card\_id = [dbo].[ResultInfo].info\_card\_id)

AS ResByLocation

WHERE ResByLocation.location\_id = [dbo].[Stats].location\_id);

1. РОБОТА З БАЗОЮ ДАНИХ
   1. Створення користувачів
      1. Учасник ЗНО

CREATE LOGIN student WITH PASSWORD = '<password123>';

GO

CREATE ROLE student;

GRANT SELECT ON [dbo].[Certificate] TO student;

GRANT SELECT ON [dbo].[InformationCard] TO student;

GRANT SELECT ON [dbo].[ResultInfo] TO student;

GRANT SELECT ON [dbo].[Stats] TO student;

GRANT SELECT, INSERT ON [dbo].[Student] TO student;

GRANT INSERT ON [dbo].[TestingLocation] TO student;

CREATE USER new\_student FOR LOGIN student;

ALTER ROLE student ADD MEMBER new\_student;

* + 1. Перевіряючий

CREATE LOGIN examiner WITH PASSWORD = '<password123>';

GO

CREATE ROLE examiner;

GRANT SELECT ON [dbo].[Certificate] TO examiner;

GRANT SELECT ON [dbo].[InformationCard] TO examiner;

GRANT SELECT, INSERT, UPDATE, DELETE ON [dbo].[ResultInfo] TO examiner;

GRANT SELECT ON [dbo].[ScoreScale] TO examiner;

GRANT SELECT ON [dbo].[Session] TO examiner;

GRANT SELECT ON [dbo].[Stats] TO examiner;

GRANT SELECT, INSERT, UPDATE, DELETE ON [dbo].[Student] TO examiner;

GRANT SELECT ON [dbo].[Subject] TO examiner;

GRANT SELECT ON [dbo].[TestingLocation] TO examiner;

CREATE USER new\_examiner FOR LOGIN examiner;

ALTER ROLE examiner ADD MEMBER new\_examiner;

* 1. SQL запити
     1. Генератори

Під час створення таблиць були створені генератори типу IDENTITY(1,1) для primary key. Тому при додаванні нових значень у таблиці нам не потрібно вручну додавати значення.

Наприклад, для заповнення таблиці Stats використовується запит INSERT:

INSERT INTO [dbo].[Stats] (location\_id, [registered\_num])

SELECT [dbo].[TestingLocation].[location\_id], COUNT([dbo].[Student].[student\_id])

FROM [dbo].[TestingLocation] JOIN [dbo].[Student]

ON [dbo].[TestingLocation].location\_id = [dbo].[Student].location\_id

GROUP BY [dbo].[TestingLocation].[location\_id];

При виконанні даного запиту поле stats\_id заповнюється автоматично. В результаті маємо:

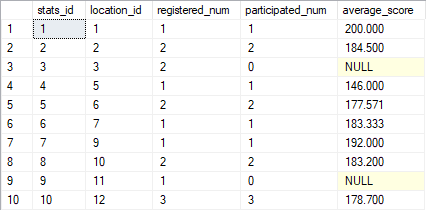


Рисунок 5.1 – Приклад роботи генераторів

* + 1. Збережені процедури та функції

1. Процедура для автоматичного розрахунку балів ЗНО та ДПА

CREATE PROC CountScore

AS

UPDATE [dbo].[ResultInfo]

SET

[result] = [dbo].[ScoreScale].rating\_score,

[sfe] = [dbo].[ScoreScale].[sfe]

FROM [dbo].[ScoreScale]

WHERE [dbo].[ResultInfo].test\_score = [dbo].[ScoreScale].test\_score AND [dbo].[ResultInfo].subject\_id = [dbo].[ScoreScale].subject\_id;

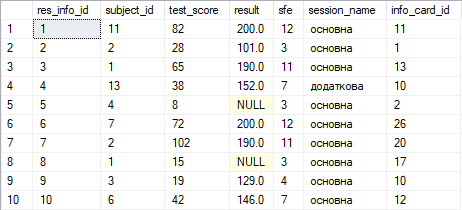


Рисунок 5.2 – Приклад роботи збереженої процедури

1. Функція для отримання середнього балу ЗНО за id учасника

CREATE FUNCTION dbo.AvgScoreById(@id INT)

RETURNS DECIMAL(6,3)

AS

BEGIN

DECLARE @avg\_score int;

SET @avg\_score = (SELECT AVG(result)

FROM [dbo].[ResultInfo] JOIN [dbo].[Student]

ON [dbo].[ResultInfo].info\_card\_id = [dbo].[Student].info\_card\_id

WHERE [dbo].[Student].student\_id = @id);

RETURN @avg\_score;

END;

За допомогою SQL-скрипту: SELECT dbo.AvgScoreById(3) AS average\_score перевіримо роботу функції



Рисунок 5.3 – Приклад роботи функції

* + 1. Тригери

1. Тригер на додавання нового учасника. При цьому в таблиці зі статистикою кількість зареєстрованих збільшується на 1

CREATE TRIGGER Student\_insert

ON [dbo].[Student]

AFTER INSERT

AS

BEGIN

UPDATE [dbo].[Stats]

SET [dbo].[Stats].registered\_num = [dbo].[Stats].registered\_num + 1

FROM [dbo].[Stats] JOIN [dbo].[Student]

ON [dbo].[Student].location\_id = [dbo].[Stats].location\_id

END;



Рис 5.4 – Поле таблиці Stats до додавання учасника



Рис 5.5 – Поле таблиці Stats після додавання учасника

1. Тригер на видалення учасника. При цьому в таблиці зі статистикою кількість зареєстрованих зменшується на 1

CREATE TRIGGER Student\_delete

ON [dbo].[Student]

AFTER DELETE

AS

BEGIN

UPDATE [dbo].[Stats]

SET [dbo].[Stats].registered\_num = [dbo].[Stats].registered\_num - 1

FROM [dbo].[Stats] JOIN [dbo].[Student]

ON [dbo].[Student].location\_id = [dbo].[Stats].location\_id

END;



Рис 5.6 – Поле таблиці Stats до видалення учасника



Рис 5.7 – Поле таблиці Stats після видалення учасника

* + 1. Представлення

Представлення для id учасника, номеру його інформаційної картки, назви предмету, кількості балів, максимальної кількості балів з даного предмету, результату в шкалі 100-200 балів та шкалі 1-12 балів

CREATE VIEW results AS

SELECT

[dbo].[Student].student\_id, [dbo].[InformationCard].info\_card\_id, [dbo].[Subject].subject\_name, [dbo].[ResultInfo].res\_info\_id,

[dbo].[ResultInfo].test\_score, [dbo].[Subject].max\_test\_score, [dbo].[ResultInfo].result, [dbo].[ResultInfo].sfe

FROM

[dbo].[Student] JOIN [dbo].[InformationCard] ON [dbo].[Student].info\_card\_id = [dbo].[InformationCard].info\_card\_id

JOIN [dbo].[ResultInfo] ON [dbo].[ResultInfo].info\_card\_id = [dbo].[InformationCard].info\_card\_id

JOIN [dbo].[Subject] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

За допомогою SQL-скрипту: SELECT \* FROM results ORDER BY student\_id отримаємо

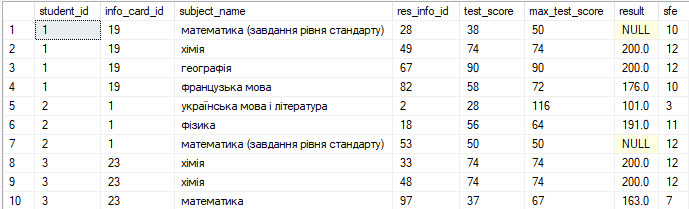


Рисунок 5.8 – Приклад роботи представлення

* + 1. SQL-запити

1. Запит з використанням JOIN для відображення балів ЗНО по всіх предметах для кожного учасника. А також сортування за прізвищем

SELECT

[dbo].[Student].full\_name, [dbo].[Subject].subject\_name, [dbo].[ResultInfo].result

FROM

[dbo].[Student] JOIN [dbo].[ResultInfo] ON [dbo].[Student].info\_card\_id = [dbo].[ResultInfo].info\_card\_id

JOIN [dbo].[Subject] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

ORDER BY full\_name

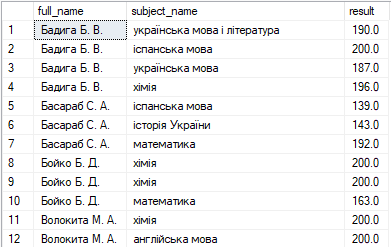


Рисунок 5.9 – Результат запиту

1. Запит з використанням JOIN для відображення даних інформаційної картки та сертифікати для кожного учасника

SELECT

[dbo].[Student].full\_name, [dbo].[Student].info\_card\_id, [dbo].[InformationCard].pin\_code,

[dbo].[InformationCard].year\_of\_receipt, [dbo].[Certificate].subjects

FROM

[dbo].[Student] JOIN [dbo].[InformationCard] ON [dbo].[Student].info\_card\_id = [dbo].[InformationCard].info\_card\_id

JOIN [dbo].[Certificate] ON [dbo].[InformationCard].certificate\_id = [dbo].[Certificate].certificate\_id

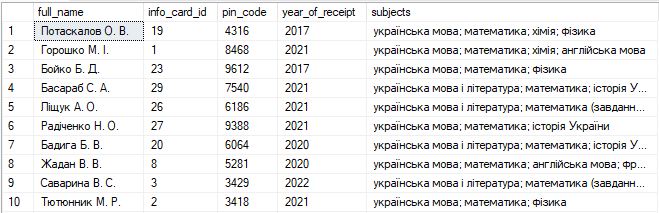


Рисунок 5.10 – Результат запиту

1. Запит з використанням JOIN, функції SUM() та GROUP BY для відображення кількості зареєстрованих та тих, хто взяли участь у ЗНО по областях

SELECT

[dbo].[TestingLocation].region, SUM([dbo].[Stats].registered\_num) AS registered,

SUM([dbo].[Stats].participated\_num) AS participated

FROM

[dbo].[TestingLocation] JOIN [dbo].[Stats] ON [dbo].[TestingLocation].location\_id = [dbo].[Stats].location\_id

GROUP BY region

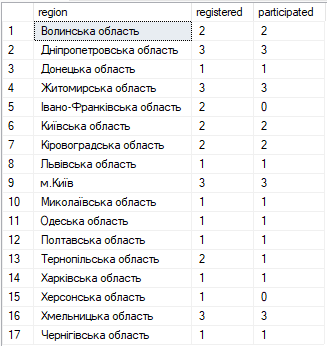


Рисунок 5.11 – Результат запиту

1. Запит з використанням JOIN, функції AVG() та GROUP BY для відображення середнього балу ЗНО по областях

SELECT

[dbo].[TestingLocation].region, AVG([dbo].[Stats].average\_score) AS average\_score

FROM

[dbo].[TestingLocation] JOIN [dbo].[Stats] ON [dbo].[TestingLocation].location\_id = [dbo].[Stats].location\_id

GROUP BY region

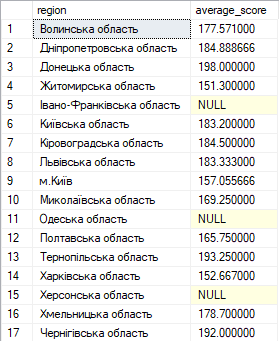


Рисунок 5.12 – Результат запиту

1. Запит з використанням JOIN, функції AVG() та GROUP BY для відображення середнього балу ЗНО по предметах

SELECT

[dbo].[Subject].subject\_name, AVG([dbo].[ResultInfo].result) AS average\_score

FROM

[dbo].[Subject] JOIN [dbo].[ResultInfo] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

GROUP BY subject\_name

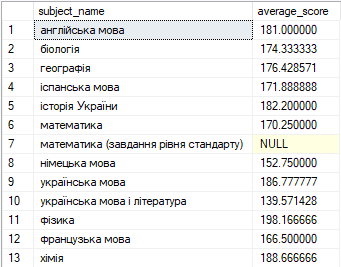


Рисунок 5.13 – Результат запиту

1. Запит з використанням JOIN, функції COUNT() та GROUP BY для відображення кількості людей, які отримали максимальний бал ЗНО по предметах

SELECT

[dbo].[Subject].subject\_name, COUNT([dbo].[ResultInfo].result) AS max\_num

FROM

[dbo].[Subject] JOIN [dbo].[ResultInfo] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

WHERE [dbo].[ResultInfo].result = 200

GROUP BY subject\_name

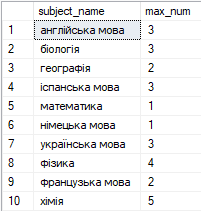


Рисунок 5.14 – Результат запиту

1. Запит з використанням JOIN, функції COUNT() та GROUP BY для відображення кількості людей в кожній сесії

SELECT

[dbo].[ResultInfo].session\_name, COUNT([dbo].[Student].student\_id) AS stud\_num

FROM [dbo].[ResultInfo] JOIN [dbo].[Student] ON [dbo].[ResultInfo].info\_card\_id = [dbo].[Student].info\_card\_id

GROUP BY session\_name

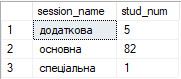


Рисунок 5.15 – Результат запиту

1. Запит з використанням JOIN, функції COUNT() та GROUP BY для відображення кількості людей, які писали екзамен по предметах

SELECT

[dbo].[Subject].subject\_name, COUNT([dbo].[ResultInfo].subject\_id) AS stud\_num

FROM [dbo].[Subject] JOIN [dbo].[ResultInfo] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

GROUP BY subject\_name

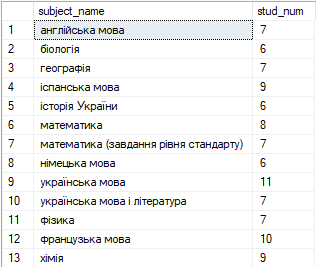


Рисунок 5.16 – Результат запиту

1. Запит з використанням JOIN для отримання шкали балів для предмета по його індексу

SELECT [dbo].[Subject].subject\_name, [dbo].[ScoreScale].test\_score, [dbo].[ScoreScale].rating\_score, [dbo].[ScoreScale].sfe

FROM [dbo].[Subject] JOIN [dbo].[ScoreScale] ON [dbo].[Subject].subject\_id = [dbo].[ScoreScale].subject\_id

WHERE [dbo].[Subject].subject\_id = 2

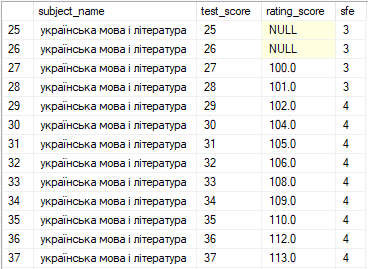


Рисунок 5.17 – Результат запиту

1. Запит з використанням JOIN та функції COUNT() для отримання кількості учасників, які отримали 200 балів

SELECT COUNT(DISTINCT [student\_id]) AS stud\_num

FROM [dbo].[Student]

JOIN [dbo].[ResultInfo] ON [dbo].[ResultInfo].info\_card\_id = [dbo].[Student].info\_card\_id

WHERE [dbo].[ResultInfo].result = 200



Рисунок 5.18 – Результат запиту

1. Запит з використанням JOIN для отримання імен учасників, які складали ЗНО у 2022 році

SELECT full\_name

FROM [dbo].[Student]

JOIN [dbo].[InformationCard] ON [dbo].[Student].info\_card\_id = [dbo].[InformationCard].info\_card\_id

WHERE [dbo].[InformationCard].year\_of\_receipt = 2022

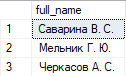


Рисунок 5.19 – Результат запиту

1. Запит з використанням JOIN та функції COUNT() для отримання кількості учасників, які складали ЗНО по роках

SELECT

[dbo].[InformationCard].year\_of\_receipt, COUNT([dbo].[Student].student\_id) AS stud\_num

FROM

[dbo].[InformationCard] JOIN [dbo].[Student] ON [dbo].[Student].info\_card\_id = [dbo].[InformationCard].info\_card\_id

GROUP BY year\_of\_receipt

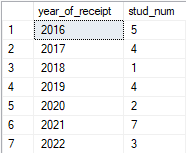


Рисунок 5.20 – Результат запиту

1. Запит з використанням JOIN для отримання імен учасників, які зареєструвалися, але не взяли участь у ЗНО із зазначенням регіону

SELECT [dbo].[Student].full\_name,

([dbo].[TestingLocation].region + ', ' + [dbo].[TestingLocation].district + ', ' + [dbo].[TestingLocation].settlement) AS region

FROM [dbo].[Student] JOIN [dbo].[TestingLocation] ON [dbo].[Student].location\_id = [dbo].[TestingLocation].location\_id

WHERE [dbo].[Student].participated = 0

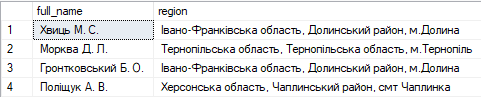


Рисунок 5.21 – Результат запиту

1. Запит із використанням JOIN, функції SUM(), GROUP BY та ORDER BY для сортування учасників за сумою балів ЗНО

SELECT [dbo].[Student].full\_name, SUM([dbo].[ResultInfo].result) AS sum\_result

FROM [dbo].[Student] JOIN [dbo].[ResultInfo] ON [dbo].[Student].info\_card\_id = [dbo].[ResultInfo].info\_card\_id

GROUP BY full\_name

ORDER BY sum\_result DESC

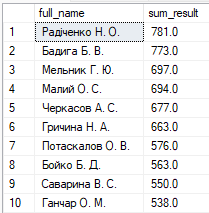


Рисунок 5.22 – Результат запиту

1. Запит із використанням JOIN, функції AVG(), GROUP BY та ORDER BY для сортування учасників за середнім балом ЗНО

SELECT [dbo].[Student].full\_name, AVG([dbo].[ResultInfo].result) AS avg\_result

FROM [dbo].[Student] JOIN [dbo].[ResultInfo] ON [dbo].[Student].info\_card\_id = [dbo].[ResultInfo].info\_card\_id

GROUP BY full\_name

ORDER BY avg\_result DESC

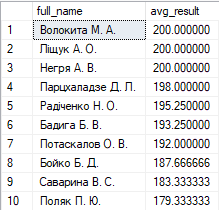


Рисунок 5.23 – Результат запиту

1. Запит із використанням JOIN, функції COUNT(), GROUP BY та HAVING для знаходження кількості учасників, які отримали 200 балів з двох і більше предметів

SELECT COUNT(\*) AS stud\_num

FROM (

SELECT [student\_id]

FROM [dbo].[Student]

JOIN [dbo].[ResultInfo] ON [dbo].[ResultInfo].info\_card\_id = [dbo].[Student].info\_card\_id

WHERE [dbo].[ResultInfo].result = 200

GROUP BY [student\_id]

HAVING COUNT(\*) > 1) AS students



Рисунок 5.24 – Результат запиту

1. Запит з використанням JOIN, функції COUNT() та GROUP BY для відображення кількості людей, які писали екзамен по предметах та подолали поріг склав/не склав

SELECT

[dbo].[Subject].subject\_name, COUNT([dbo].[ResultInfo].subject\_id) AS stud\_num

FROM [dbo].[Subject] JOIN [dbo].[ResultInfo] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

WHERE [dbo].[ResultInfo].result IS NOT NULL

GROUP BY subject\_name

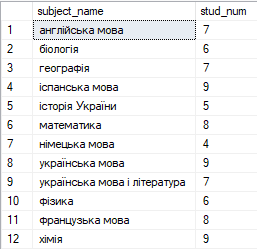


Рисунок 5.25 – Результат запиту

1. Запит з використанням JOIN, функції COUNT() та GROUP BY для відображення кількості людей, які складали ЗНО в основну сесію та отримали більше 150 балів по предметах

SELECT

[dbo].[Subject].subject\_name, COUNT([dbo].[ResultInfo].subject\_id) AS stud\_num

FROM [dbo].[Subject] JOIN [dbo].[ResultInfo] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

WHERE [dbo].[ResultInfo].result >= 150 AND [dbo].[ResultInfo].session\_name = 'основна'

GROUP BY subject\_name

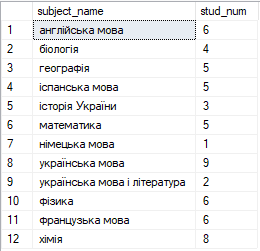


Рисунок 5.26 – Результат запиту

1. Запит з використанням JOIN, функції COUNT(), ROW\_NUMBER(), GROUP BY та ORDER BY для відображення рейтингу популярності серед предметів

SELECT

ROW\_NUMBER() OVER(ORDER BY COUNT([dbo].[ResultInfo].subject\_id) DESC) subject\_num, [dbo].[Subject].subject\_name

FROM [dbo].[Subject] JOIN [dbo].[ResultInfo] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

GROUP BY subject\_name

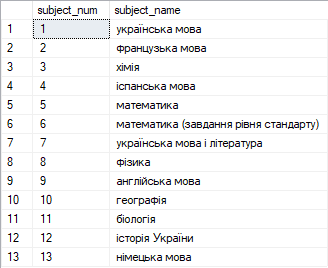


Рисунок 5.27 – Результат запиту

1. Запит з використанням JOIN, функції MIN() та GROUP BY для відображення мінімального балу з предметів (не враховуючи тих, хто не подолав поріг склав/не склав)

SELECT

[dbo].[Subject].subject\_name, MIN([dbo].[ResultInfo].result) AS average\_score

FROM

[dbo].[Subject] JOIN [dbo].[ResultInfo] ON [dbo].[Subject].subject\_id = [dbo].[ResultInfo].subject\_id

GROUP BY subject\_name

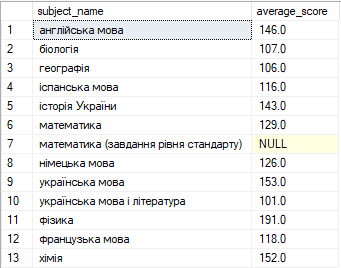


Рисунок 5.28 – Результат запиту

* + 1. Індекси та їхній приклад роботи

Для пришвидшення пошуку в таблицях з великою кількістю даних використовують індекси.

Створимо індекс на колонку subject\_id для таблиці ScoreScale:

CREATE INDEX idx\_subject\_id

ON [dbo].[ScoreScale] ([subject\_id])

Перевіримо роботу індексу на простому запиті SELECT \* FROM [dbo].[ScoreScale];

Час роботи запиту без використання індексу: 31 мс

Час роботи запиту з використанням індексу: 24 мс

Як бачимо використання індексу пришвидшило роботу запиту.

# ВИСНОВКИ

В ході даної лабораторної роботи було створено базу даних з результатами з результатами зовнішнього незалежного оцінювання.

Першим кроком був аналіз предметного середовища. На основі якого були визначені сутності, їхні атрибути та зв’язки між об’єктами.

На наступному етапі бела побудована ER-діаграма заданого предметного середовища. На її основі створено реляційну схему бази даних. Виділено первинні та зовнішні ключі, додано обмеження для підтримки цілісності бази даних.

Написано SQL скрипти для побудови спроєктованої бази даних. А саме: створення таблиць, обмежень, користувачів; заповнення таблиць відповідними даними; створення збережених процедур, функцій, представлень, тригерів та індексів.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. <https://testportal.gov.ua/zno-dpa-2/>
2. <https://www.w3schools.com/sql/default.asp>
3. <https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16>
4. <https://learn.microsoft.com/en-us/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver16>