**Звіт до лабораторної роботи №3**

**Завдання 1**

**Мета роботи**

1. Створити базу даних **database.db** та таблицю **Students** з полями *id*, *name*, *age*, *grade*.
2. Додати записи в таблицю **Students**.
3. Отримати та вивести всі записи з таблиці.
4. Оновити дані одного з записів.
5. Видалити один із записів.

**Опис реалізації**

Для виконання завдання було створено кілька окремих файлів Python, кожен з яких реалізує одну операцію з базою даних. Для взаємодії з базою даних використовується бібліотека sqlite3.

**Кроки виконання**

* **Створення бази даних та таблиці Students (файл sqlite.py)**:
  + Програма створює файл бази даних **database.db**, якщо його ще не існує.
  + Виконується SQL-запит для створення таблиці **Students** з полями:
    - id (*ціле число, первинний ключ*),
    - name (*текст, обов'язковий*),
    - age (*ціле число*),
    - grade (*текст*).
  + Таблиця створюється лише один раз, якщо вона ще не існує, завдяки **IF NOT EXISTS**.
  + Після створення таблиці програма зберігає зміни (**conn.commit()**) та закриває з'єднання.

**Вивід у термінал**:

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

* **Додавання даних у таблицю Students (файл sqlite\_add.py)**:
  + За допомогою **INSERT INTO** у таблицю додаються три записи зі студентами: **Іван** (*20 років, оцінка "A"*), **Марія** (*22 роки, оцінка "B"*), та **Олександр** (*21 рік, оцінка "C"*).
  + Після додавання даних зміни зберігаються, і з'єднання закривається.

**Вивід у термінал**:

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

* **Отримання та виведення всіх записів з таблиці Students (файл sqlite\_fetch.py)**:
  + Виконується SQL-запит *SELECT \* FROM* **Students** для отримання всіх записів із таблиці.
  + Отримані записи виводяться у вигляді списку, де кожен елемент – кортеж, що представляє рядок із таблиці (наприклад: *(1, 'Іван', 20, 'A')*).

**Приклад виводу у термінал**:

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

* **Оновлення даних (файл sqlite\_update.py)**:
  + Програма оновлює вік студента на ім'я Іван до 23 років, використовуючи SQL-запит *UPDATE* **Students** *SET* **age** = ? *WHERE* **name** = ?.
  + Після оновлення змінені дані зберігаються.

**Вивід у термінал**:

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

* **Видалення даних (файл sqlite\_delete.py)**:
  + Програма видаляє запис про студента з ім'ям Марія.
  + Використовується SQL-запит *DELETE FROM* **Students** *WHERE* **name** = ?, після чого зміни зберігаються.

**Вивід у термінал**:

Зображення, що містить текст, Шрифт, знімок екрана

Автоматично згенерований опис

**Висновок**

У цьому завданні я навчився:

1. Створювати таблиці в базі даних SQLite та працювати з ними.
2. Реалізував основні операції з базою даних: додавання, читання, оновлення та видалення даних.
3. Отримав досвід використання бібліотеки sqlite3 для маніпуляції даними в локальній базі даних SQLite.

Ця програма є прикладом простої системи управління базою даних для зберігання та обробки інформації про студентів.

**Завдання 2**

**Мета роботи**

Ознайомитись з використанням ORM бібліотеки SQLAlchemy для виконання таких операцій:

1. Створення моделі **Student**, що представляє таблицю *students*.
2. Додавання даних у таблицю через **ORM**.
3. Отримання та виведення всіх записів з таблиці *students*.

**Опис реалізації**

Для виконання завдання було розроблено програму **sqlalchemy\_lab.py**, яка реалізує створення та керування базою даних **students.db** за допомогою **ORM SQLAlchemy**.

**Кроки виконання**

1. **Налаштування бази даних та визначення моделі Student**:
   * Підключення до бази даних SQLite здійснюється через параметр *DATABASE\_URL*, який вказує шлях до бази даних **students.db**.
   * Базовий клас для моделей створюється за допомогою **declarative\_base()**.
   * Модель **Student** представляє таблицю *students* з такими стовпцями:
     + id — *ціле число, первинний ключ з автоінкрементом*
     + name — *текстове поле для імені студента, обов'язкове*
     + age — *ціле число для віку студента, обов'язкове*
     + grade — *текстове поле для оцінки студента, обов'язкове*
   * Метод **\_\_repr\_\_** реалізований для зручного відображення об'єктів **Student** у вигляді рядка.
2. **Створення бази даних та таблиці *students***:
   * Використовується **create\_engine** для встановлення з'єднання з базою даних.
   * **Base.metadata.create\_all(engine)** створює таблицю *students*, якщо вона ще не існує в базі даних.
3. **Додавання записів у таблицю *students***:
   * Функція **add\_students** створює об'єкти класу **Student** для трьох студентів: *Івана, Марії та Олександра*.
   * Метод **session.add\_all** додає створені об'єкти до сесії, а **session.commit** зберігає зміни в базі даних.

**Вивід у термінал** **після додавання студентів**:

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

1. **Отримання та виведення всіх записів з таблиці *students***:
   * Функція **get\_students** виконує запит **session.query(Student).all()**, щоб отримати всі записи з таблиці.
   * Усі записи виводяться у термінал за допомогою циклу **for**.

**Вивід у термінал**:

Зображення, що містить текст, Шрифт, знімок екрана, ряд

Автоматично згенерований опис

1. **Закриття сесії**:
   * Після виконання операцій сесія закривається за допомогою **session.close()**.

**Висновок**

У цьому завданні я зміг:

1. Ознайомитись з основами та використанням ORM бібліотеки SQLAlchemy для створення моделей, що відображають таблиці бази даних.
2. Реалізувати основні операції: створення таблиці, додавання та отримання даних.
3. Дослідити, як використовувати SQLAlchemy для зручного та безпечного управління даними у базі, без необхідності писати SQL-запити.

Ця програма є прикладом простої системи для роботи з базою даних студентів, що використовує ORM для забезпечення абстракції при взаємодії з даними.