Вступна

У цьому домашньому завданні ми продовжимо працювати з домашнім завданням із попереднього модуля.
В цій домашній роботі використаємо базу даних `postgres`. У командному рядку запустіть `Docker` контейнер:

```
docker run --name some-postgres -p 5432:5432 -e
POSTGRES_PASSWORD=mysecretpassword -d postgres
```

Замість `some-postgres` виберіть свою назву контейнера, а замість `mysecretpassword` придумайте свій пароль для підключення до бази даних

CAUTION

За домовленістю з ментором та технічною неможливістю використовувати `postgres`, можна замінити її на SQLite

Кроки виконання домашнього завдання

Перший крок

Реалізуйте свої моделі **SQLAlchemy**, для таблиць:

- Таблиця студентів;
- Таблиця груп;
- Таблиця викладачів;
- Таблиця предметів із вказівкою викладача, який читає предмет;
- Таблиця де кожен студент має оцінки з предметів із зазначенням коли оцінку отримано;

Другий крок

Використовуйте `alembic` для створення міграцій у базі даних.

Третій крок

Напишіть скрипт `seed.py` та заповніть отриману базу даних випадковими даними (~30-50 студентів, 3 групи, 5-8 предметів, 3-5 викладачів, до 20 оцінок у кожного студента з усіх предметів). Використовуйте пакет `Faker` для наповнення. При заповненні використовуємо механізм сесій **SQLAlchemy**.

Четвертий крок

Зробити такі вибірки з отриманої бази даних:

- Знайти 5 студентів із найбільшим середнім балом з усіх предметів.
- Знайти студента із найвищим середнім балом з певного предмета.
- Знайти середній бал у групах з певного предмета.
- Знайти середній бал на потоці (по всій таблиці оцінок).
- Знайти які курси читає певний викладач.
- Знайти список студентів у певній групі.

Знайти оцінки студентів у окремій групі з певного предмета.
Знайти середній бал, який ставить певний викладач зі своїх предметів.
Знайти список курсів, які відвідує певний студент.
Список курсів, які певному студенту читає певний викладач.

Для запитів оформити окремий файл `my_select.py`, де будуть `10` функцій від `select_1` до `select_10`. Виконання функцій повинно повертати результат аналогічний попередньої домашньої роботи. При запитах використовуємо механізм сесій **SQLAlchemy**.

Підказки та рекомендації

Це завдання перевірить вашу здатність користуватися документацією **SQLAlchemy**. Але основні підказки та напрямки рішення ми вам дамо одразу. Нехай у нас є наступний запит.
Знайти 5 студентів з найбільшим середнім балом з усіх предметів.

```
   SELECT s.fullname, round(avg(g.grade), 2) AS avg_grade
    FROM grades g
    LEFT JOIN students s ON s.id = g.student_id
    GROUP BY s.id
    ORDER BY avg_grade DESC
    LIMIT 5;
```

Спробуймо його перевести в запит *ORM SQLAlchemy*. Нехай у нас є сесія у змінній `session`. Є описані моделі `Student` та `Grade` для відповідних таблиць.
Вважаємо, що база даних вже заповнена даними. Функції агрегації *SQLAlchemy* зберігає в об'єкті `func`. Його треба спеціально імпортувати `from sqlalchemy import func` і тоді ми зможемо використати методи `func.round` та `func.avg`. Отже перший рядок SQL запиту має виглядати так `session.query(Student.fullname, func.round(func.avg(Grade.grade), 2).label('avg_grade'))`. Тут ми використали ще `label('avg_grade')` так *ORM* виконує найменування поля, із середнім балом, за допомогою оператора `AS`.
Далі `FROM grades g` замінюється методом `select_from(Grade)`. Заміна оператора `JOIN` - тут все просто це функція `join(Student)`, все інше на себе бере *ORM*. Групування по полю виконуємо функцією `group_by(Student.id)`.
За сортування відповідає функція `order_by`, яка, за замовчуванням, сортує як `ASC`, а нам явно треба режим зростання `DESC` та ще й по полю `avg_grade`, яке ми самі створили у запиті. Імпортуємо `from sqlalchemy import func, desc` та остаточний вигляд — `order_by(desc('avg_grade'))`. Ліміт у п'ять значень це функція з такою самою назвою `limit(5)`. Ось і все, наш запит готовий.
Остаточний варіант запиту для ORM SQLAlchemy.

```
session.query(Student.fullname, func.round(func.avg(Grade.grade), 2).label('avg_grade'))\

.select_from(Grade).join(Student).group_by(Student.id).order_by(desc('avg_grade')).limit(5).all()
```

Можливе виведення:

```
[('Mary Smith', Decimal('8.33')), ('Kimberly Howard', Decimal('8.17')), ('Gregory Graves', Decimal('7.92')), ('Mrs. Diamond Carter', Decimal('7.53')), ('Emma Hernandez', Decimal('7.31'))]
```

Інші запити ви повинні побудувати аналогічно викладеному вище прикладу. І остання підказка, якщо ви вирішите зробити вкладені запити, тоді використовуйте `scalar-selects`

Додаткове завдання

Перша частина

Для додаткового завдання зробіть такі запити підвищеної складності:

Середній бал, який певний викладач ставить певному студентові.

Оцінки студентів у певній групі з певного предмета на останньому занятті.

Друга частина

Замість скрипту `seed.py` подумайте та реалізуйте повноцінну **CLI** програму для **CRUD** операцій із базою даних. Використовуйте для цього модуль **`argparse`**.
Використовуйте команду `--action` або скорочений варіант `-a` для **CRUD** операцій. Та команду `--model` (`-m`) для вказівки над якою моделлю проводитися операція.
Приклад:

`--action create -m Teacher --name 'Boris Jonson'` створення вчителя

`--action list -m Teacher` показати всіх вчителів

`--action update -m Teacher --id 3 --name 'Andry Bezos'` оновити дані вчителя з `id=3`

`--action remove -m Teacher --id 3` видалити вчителя з `id=3`

Реалізуйте ці операції для кожної моделі.
INFO
Приклади виконання команд у терміналі.
Створити вчителя

```
py main.py -a create -m Teacher -n 'Boris Jonson'
```

Створити групу

```
py main.py -a create -m Group -n 'AD-101'
```