Lesson 26 - Working with DB from Python

Повторение

1. Теория из прошлого урока
2. Разбор домашнего задания

Legend

```
-- DDL (data definition language)
-- CRUD запросов над структурой таблицы или базы данных

-- DML (data manipulation language)
-- CRUD запросов к записям таблиц
```

Модуль psycopg. Подключение к серверу PostgreSQL

ресурс: https://metanit.com/python/database/2.1.php

Создание базы данных и таблицы в PostgreSQL

ресурс: https://metanit.com/python/database/2.2.php

Добавление данных в PostgreSQL

ресурс: https://metanit.com/python/database/2.3.php

Получение данных из БД PostgreSQL

ресурс: https://metanit.com/python/database/2.4.php

Обновление и удаление данных в PostgreSQL

ресурс: https://metanit.com/python/database/2.5.php

Код с урока

```
import psycopg2

conn = psycopg2.connect(dbname="python_db", host="localhost",
user="postgres", password="postgres", port=5432)
```

```python
conn.autocommit = True
cursor = conn.cursor()

create_users_table_script = """CREATE TABLE IF NOT EXISTS users
(
    id         SERIAL PRIMARY KEY,    full_name  VARCHAR,    age INT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP)"""

create_phones_table_script = """CREATE TABLE phones
(
    id         SERIAL PRIMARY KEY,    model      VARCHAR,    brand
VARCHAR,    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP)"""

# cursor.execute(create_users_table_script)
#
# cursor.execute(create_phones_table_script)

# выполняем код sql
# print("Таблицы успешно создана")
# vasya = ('Vasya', 35)
# john = ('John', 31)
# C - create
# insert_user_query = "INSERT INTO users (full_name, age) VALUES (%s, %s)"

# people = [("Sam", 28), ("Alice", 33), ("Kate", 25)]
# cursor.execute(insert_user_query, vasya)
# cursor.executemany(insert_user_query, people)

# cursor.execute("SELECT * FROM users")
# print(cursor.fetchall())
# for person in cursor.fetchall():
#     print(f"{person[1]} - {person[2]}")
# извлекаем первые 3 строки в полученном наборе
# print(cursor.fetchmany(3))
# print(cursor.fetchmany(3))

# cursor.execute("SELECT full_name, age FROM users WHERE id = 1")
# vasya = cursor.fetchone()
# print(f"Name: {vasya[0]}    Age: {vasya[1]}")
#
# cursor.execute("UPDATE users SET full_name ='Vasiliy' WHERE
full_name='Vasya'")
#
#
# cursor.execute("SELECT full_name, age FROM users WHERE id = 1")
# vasya = cursor.fetchone()
```

```
# print(f"Name: {vasya[0]}    Age: {vasya[1]}")
#
# people = [(99, "Sam"), (98, "Alice")]
# cursor.executemany("UPDATE users SET age =%s WHERE full_name=%s", people)

cursor.execute("DELETE FROM users WHERE name = DELETE FROM users")

cursor.close()
conn.close()
```

ДЗ-26

```
Задача 1: Создание таблицы и добавление данных
1. Создайте таблицу users с полями:
• user_id (целое число, автоинкрементируемый ключ),
• username (строка, уникальный),
• email (строка),
• created_at (дата и время).

2. Напишите скрипт для добавления нескольких пользователей в таблицу.

Задача 2: Извлечение данных

1. Добавьте 5 пользователей в таблицу users, используя скрипт из предыдущей
задачи.
2. Напишите запрос, который извлекает всех пользователей, с возможностью
фильтрации по username. Реализуйте функционал поиска пользователей по части
имени (например, поиск по первым 3 символам).

Задача 3: Обновление данных
1. Напишите скрипт, который обновляет email пользователя по его user_id.
2. Реализуйте проверку, чтобы в случае отсутствия пользователя с указанным
user_id выводилось сообщение об ошибке.


Задача 4: Удаление данных
1. Напишите запрос для удаления задачи по её task_id.
2. Добавьте проверку: если задача с таким task_id не найдена, выведите
соответствующее сообщение.

Задача 5: Пример с параметрами
1. Создайте таблицу orders с полями:
• order_id (автоинкрементируемый ключ),
• user_id (ссылается на user_id из таблицы users),
• amount (целое число),
```

• status (строка).

2. Напишите запрос, который вставляет заказ в таблицу orders, используя параметры для предотвращения SQL-инъекций.