**Лабораторна робота №7**

**Вибрані пакети мови програмування Python.**

**Завдання:**

**Варіант 4**

1. На платформі Docker, за допомогою файлу docker-compose.yml, створити контейнер з СУБД PostgreSQL або MySQL. Зробити прокидання портів та папок для зберігання БД.

2. В цьому контейнері, використовуючи мову Python, створити базу даних. Створити в ній необхідні таблиці з відповідними полями (предметна область та дані наведені нижче).

3. Визначить типи даних (лічильник, текстовий, числовий тощо) та опис, якщо потрібно.

4. Встановіть необхідні властивості полів (розмір поля, маску вводу, значення за замовченням, обмеження та повідомлення про помилку) створених таблиць.

5. Визначить первинні ключі в створених таблицях.

6. Визначить необхідні зв’язки між таблицями, задайте необхідні параметри забезпечення цілісності даних.

7. Заповнить створені таблиці даними (4 постачальника, 22 поставки, матеріали: деревина, лак, сталеві деталі)

**Предметна область: Відділ поставок** (складається з 3 класів).

Сутності та дані:

* Постачальники[Код постачальника, назва компанії постачальника, контактна особа, телефон(маска вводу), розрахунковий рахунок],
* Матеріали, що поставляються[Код матеріалу, назва матеріалу, ціна],
* Поставки[номер поставки, дата поставки, код постачальника, код матеріалу, кількість днів, протягом яких здійснюється поставка(1-7; обмеження, повідомлення про помилку), кількість матеріалів, які поставляються].

8. Створіть наступні запити: Відобразити всі поставки, які здійснюються за 3 або менше днів. Відсортувати назви постачальників за алфавітом;

Порахувати суму, яку треба сплатити за кожну поставку (запит з обчислювальним полем);

Відобразити всі поставки обраного матеріалу (запит з параметром);

Порахувати кількість кожного матеріалу, що поставляється кожним постачальником (перехресний запит);

Порахувати загальну кількість кожного матеріалу (підсумковий запит);

Порахувати кількість поставок від кожного постачальника (підсумковий запит).

9. На мові Python написати програму, що підключається до створеної БД, виводить всі таблиці (структура + дані, які в ній зберігаються) та результати виконання запитів в консоль в форматованому вигляді (заголовки стовпців + всі стовпці рівні).

10. На платформі Docker створити контейнер з графічним клієнтом (адмінка) для управління БД. Запустити його і підключитись до створеної БД. Переконатись, що всі таблиці і запити створені вірно.

11. Завантажити проект на GitHub, попередньо додавши до файлу .gitignore всі технічні папки та файли.

12. В текстовому редакторі створіть звіт наступної структури і завантажити на Elearn

• титульний аркуш (приклад наведено вище),

• текст завдання,

• скріншоти з назвами використаних в проекті docker образів та docker контейнерів,

A screenshot of a computer

Description automatically generated

• текст Python коду програм і SQL запитів,

**create\_tables.py**

from faker import Faker

import psycopg2

conn = psycopg2.connect(

    host="db",

    database="supply\_department",

    user="user",

    password="pass",

    port="5432",

)

cur = conn.cursor()

cur.execute(

    """

    CREATE TABLE IF NOT EXISTS suppliers (

        supplier\_id SERIAL PRIMARY KEY,

        company\_name VARCHAR(255) NOT NULL,

        contact\_person VARCHAR(255),

        phone VARCHAR(40),

        account\_number VARCHAR(50) NOT NULL

    );

"""

)

cur.execute(

    """

    CREATE TABLE IF NOT EXISTS materials (

        material\_id SERIAL PRIMARY KEY,

        material\_name VARCHAR(255) NOT NULL,

        price NUMERIC(10, 2) NOT NULL CHECK (price > 0)

    );

"""

)

cur.execute(

    """

    CREATE TABLE IF NOT EXISTS deliveries (

        delivery\_id SERIAL PRIMARY KEY,

        delivery\_date DATE NOT NULL,

        supplier\_id INT REFERENCES suppliers(supplier\_id),

        material\_id INT REFERENCES materials(material\_id),

        delivery\_days INT NOT NULL CHECK (delivery\_days BETWEEN 1 AND 7),

        material\_quantity INT NOT NULL CHECK (material\_quantity > 0)

    );

"""

)

conn.commit()

cur.close()

conn.close()

**fill\_tables.py**

from faker import Faker

import psycopg2

conn = psycopg2.connect(

    host="db",

    database="supply\_department",

    user="user",

    password="pass",

    port="5432",

)

cur = conn.cursor()

fake = Faker("uk\_UA")

for \_ in range(4):

    cur.execute(

        f"""

        INSERT INTO suppliers (company\_name, contact\_person, phone, account\_number)

        VALUES ('{fake.company()}', '{fake.name()}', '{fake.phone\_number()}', {fake.bban()});

        """

    )

for material in ("деревина", "лак", "сталеві деталі"):

    cur.execute(

        f"""

        INSERT INTO materials (material\_name, price)

        VALUES ('{material}', {round(fake.random\_number(digits=3), 2)});

        """

    )

for \_ in range(22):

    cur.execute("SELECT supplier\_id FROM suppliers ORDER BY RANDOM() LIMIT 1;")

    supplier\_id = cur.fetchone()[0]

    cur.execute("SELECT material\_id FROM materials ORDER BY RANDOM() LIMIT 1;")

    material\_id = cur.fetchone()[0]

    cur.execute(

        """

    INSERT INTO deliveries (delivery\_date, supplier\_id, material\_id, delivery\_days, material\_quantity)

    VALUES (%s, %s, %s, %s, %s);

    """,

        (

            fake.date\_this\_year(),

            supplier\_id,

            material\_id,

            fake.random\_int(min=1, max=7),

            fake.random\_int(min=1, max=100),

        ),

    )

conn.commit()

cur.close()

conn.close()

**sql\_queries.py**

import psycopg2

from prettytable import PrettyTable

queries = [

    """

    SELECT d.\*, s.company\_name

    FROM deliveries d

    JOIN suppliers s ON d.supplier\_id = s.supplier\_id

    WHERE d.delivery\_days <= 3

    ORDER BY s.company\_name;

""",

    """

    SELECT d.\*, (m.price \* d.material\_quantity) AS total\_payment

    FROM deliveries d

    JOIN materials m ON d.material\_id = m.material\_id;

""",

    """

    SELECT d.\*, s.company\_name

    FROM deliveries d

    JOIN suppliers s ON d.supplier\_id = s.supplier\_id

    WHERE d.material\_id = %s;

""",

    """

    SELECT s.company\_name, m.material\_name, SUM(d.material\_quantity) AS total\_quantity

    FROM deliveries d

    JOIN suppliers s ON d.supplier\_id = s.supplier\_id

    JOIN materials m ON d.material\_id = m.material\_id

    GROUP BY s.company\_name, m.material\_name

    ORDER BY s.company\_name;

""",

    """

    SELECT m.material\_name, SUM(d.material\_quantity) AS total\_quantity

    FROM deliveries d

    JOIN materials m ON d.material\_id = m.material\_id

    GROUP BY m.material\_name;

""",

    """

    SELECT s.company\_name, COUNT(d.delivery\_id) AS total\_deliveries

    FROM deliveries d

    JOIN suppliers s ON d.supplier\_id = s.supplier\_id

    GROUP BY s.company\_name;

""",

]

def print\_table(data, columns):

    table = PrettyTable()

    table.field\_names = columns

    for row in data:

        table.add\_row(row)

    print(table)

try:

    connection\_params = {

        "dbname": "supply\_department",

        "user": "user",

        "password": "pass",

        "host": "localhost",

        "port": "5544",

    }

    conn = psycopg2.connect(\*\*connection\_params)

    cur = conn.cursor()

    cur.execute(

        "SELECT table\_name FROM information\_schema.tables WHERE table\_schema='public';"

    )

    tables = cur.fetchall()

    for table in tables:

        table\_name = table[0]

        cur.execute(f"SELECT \* FROM {table\_name};")

        data = cur.fetchall()

        columns = [desc[0] for desc in cur.description]

        print(f"\nТаблиця '{table\_name}':")

        print\_table(data, columns)

    for query in queries:

        print(f"\nЗапит:\n{query.strip()}")

        if "%s" in query:

            cur.execute(query, (1,))

        else:

            cur.execute(query)

        results = cur.fetchall()

        columns = [desc[0] for desc in cur.description]

        print\_table(results, columns)

except Exception as e:

    print(f"ERROR: {e}")

• скріншоти виконання програм в власному віртуальному середовищі

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

• скріншоти всіх таблиць і запитів БД в графічному клієнті

A screenshot of a data

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

• посилання на проект на GitHub

**Github** - [https://github.com/Semachko/lab](https://github.com/Semachko/lab4)7