Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» ім. Ігоря Сікорського

Розрахунково-графічна робота

з дисципліни «Бази Даних»

«Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав студент групи: КВ-31

Фізик Назар

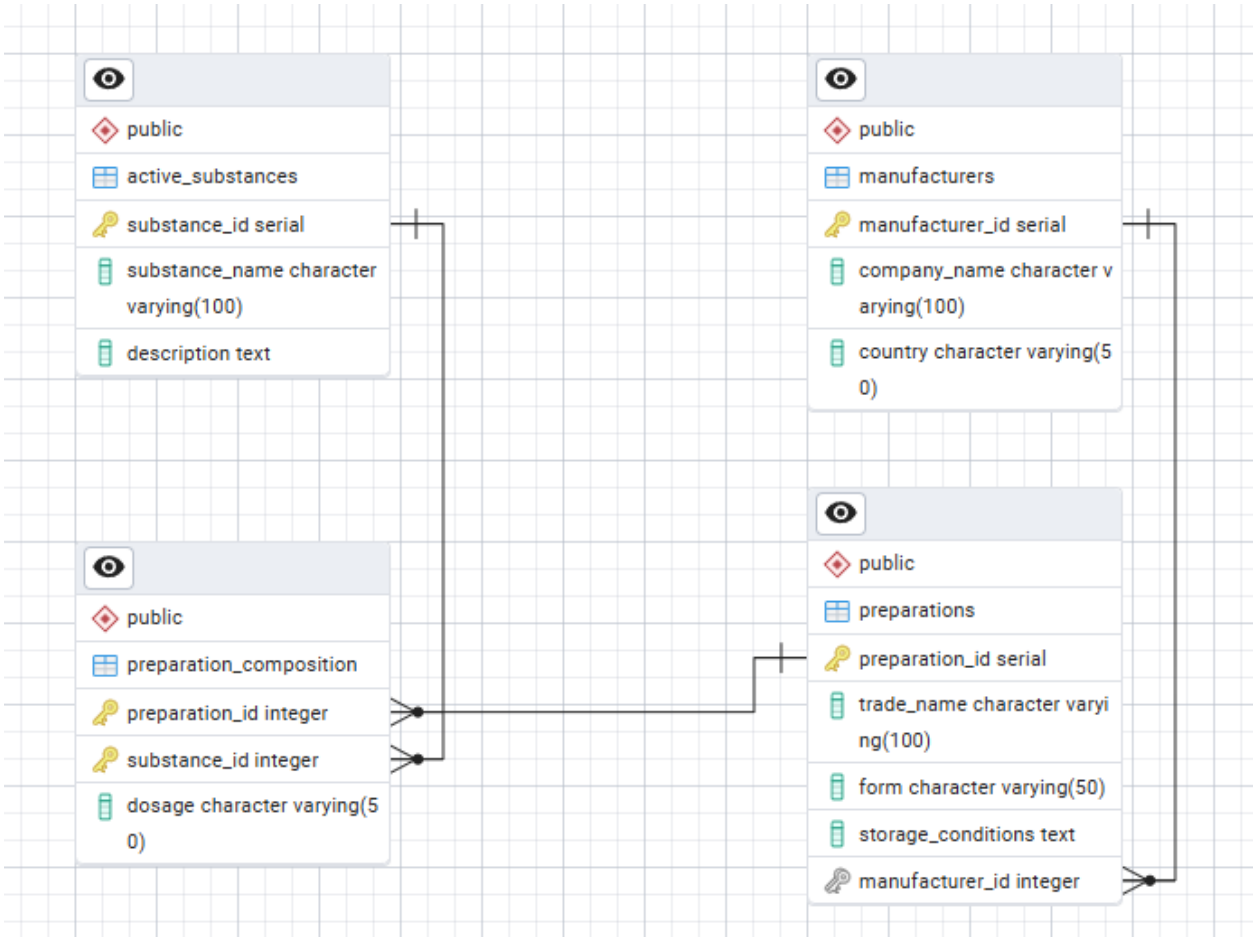
Варіант: Електронний довідник медичних препаратів

Репозиторій на GitHub: https://github.com/nazarfizyk1907/BD

Telegram: @PtahaF

Київ 2025


Електронний довідник медичних препаратів
Назва речовини
id
Опис
Діючі речовини
містить
Склад
id
Дозування
є в
Назва компанії
id
Країна
Виробники
виробляють
id
Препарати
Торгова назва
Форма випуску
Умови зберігання


public
active_substances
substance_id serial
substance_name character varying(100)
description text
public
manufacturers
manufacturer_id serial
company_name character varying(100)
country character varying(50)
public
preparation_composition
preparation_id integer
substance_id integer
dosage character varying(50)
public
preparations
preparation_id serial
trade_name character varying(100)
form character varying(50)
storage_conditions text
manufacturer_id integer

Опис

Сутність Preparations (Препарати)

- Призначення: Зберігає інформацію про конкретні торгові назви ліків.
- Атрибути: preparation_id (ключ), trade_name (торгова назва), form (форма випуску, напр., "таблетки", "сироп"), storage_conditions (умови зберігання).

Сутність Manufacturers (Виробники)

- Призначення: Зберігає інформацію про компанії, які виробляють препарати.
- Атрибути: manufacturer_id (ключ), company_name (назва компанії), country (країна).

Сутність ActiveSubstances (Діючі_Речовини)

- Призначення: Довідник діючих речовин, які є основою препаратів
- Атрибути: substance_id (ключ), substance_name (назва речовини), description (опис).

Manufacturers 1:N Preparations

- Один виробник (Manufacturers) може випускати багато препаратів (Preparations). Один препарат (Preparations) випускається лише одним виробником.

Preparations N:M ActiveSubstances

- Один препарат (Preparations) може містити багато діючих речовин. І одна діюча речовина (ActiveSubstances) може входити до складу багатьох різних препаратів.

Зв'язок з атрибутом

- Зв'язок N:M між Preparations та ActiveSubstances (назвемо його Composition - "Склад") ідеально підходить для цього.
- Атрибут зв'язку: dosage (дозування).
- Дозування не є властивістю лише препарату (бо він може мати кілька речовин) і не є властивістю лише речовини (бо вона може бути в різних дозах у різних препаратах). Дозування — це атрибут, який описує зв'язок між конкретним препаратом і конкретною речовиною.

Схема меню

Схема меню користувача

Програма має головне меню з 19 основними опціями та одним підменю для пошуку.

1. Головне меню

- 1. Показати всіх виробників
    - Функціональність: Виконує SELECT до таблиці manufacturers та виводить у консоль список всіх виробників (ID, Назва, Країна).
- 2. Додати нового виробника
    - Функціональність: Запитує у користувача назву компанії та країну. Виконує INSERT для створення нового запису в manufacturers.
- 3. Редагувати виробника
    - Функціональність: Запитує ID виробника, знаходить його, показує поточні дані та запитує нові (назву, країну). Виконує UPDATE для цього ID.
- 4. Видалити виробника
    - Функціональність: Запитує ID виробника. Виконує DELETE. Програма перехоплює помилку ForeignKeyViolation, якщо цей виробник має пов'язані препарати, та повідомляє користувача про неможливість видалення.
- 5. Показати всі діючі речовини
    - Функціональність: Виконує SELECT до таблиці active_substances та виводить список (ID, Назва, Опис).
- 6. Додати нову діючу речовину
    - Функціональність: Запитує у користувача назву та опис. Виконує INSERT у active_substances.
- 7. Редагувати діючу речовину
    - Функціональність: Запитує ID речовини, знаходить її та дозволяє оновити назву або опис (UPDATE).
- 8. Видалити діючу речовину

    - Функціональність: Запитує ID речовини. Виконує DELETE. Перехоплює помилку ForeignKeyViolation, якщо речовина використовується у preparation_composition.
- 9. Показати всі препарати
    - Функціональність: Виконує SELECT з JOIN таблиці manufacturers для відображення списку препаратів разом з назвою компанії-виробника.
- 10. Додати новий препарат
    - Функціональність: Запитує дані препарату (назву, форму, умови зберігання) та показує список виробників для вибору валідного manufacturer_id. Виконує INSERT у preparations.
- 11. Редагувати препарат
    - Функціональність: Запитує ID препарату, знаходить його та дозволяє оновити всі дані, включаючи manufacturer_id.
- 12. Видалити препарат
    - Функціональність: Запитує ID препарату. Виконує DELETE. Перехоплює помилку ForeignKeyViolation (якщо препарат має записи у preparation_composition).
- 13. Показати весь склад препаратів
    - Функціональність: Виконує SELECT до N:M таблиці preparation_composition з JOIN таблиць preparations та active_substances для відображення повних даних (Назва препарату, Назва речовини, Дозування).
- 14. Додати речовину до препарату
    - Функціональність: Запитує ID препарату, ID речовини та дозування. Виконує INSERT у preparation_composition. Перехоплює помилки UniqueViolation та ForeignKeyViolation.
- 15. Редагувати дозування у складі
    - Функціональність: Запитує композитний ключ (ID препарату, ID речовини) та нове значення для дозування. Виконує UPDATE для preparation_composition.
- 16. Видалити речовину зі складу
    - Функціональність: Запитує композитний ключ (ID препарату, ID речовини) та виконує DELETE з preparation_composition.

- 17. Автоматична генерація даних
    - Функціональність: Запитує у користувача кількість записів для кожної з 4-х таблиць. Виконує пакетні SQL-запити для генерації випадкових, але валідних даних.
- 18. Меню пошуку
    - Функціональність: Перехід до окремого підменю для виконання трьох комплексних пошукових запитів.
- 19. ОЧИСТИТИ ВСІ ДАНІ
    - Функціональність: Запитує підтвердження у користувача. Виконує TRUNCATE ... RESTART IDENTITY CASCADE для миттєвого очищення всіх даних у 4-х таблицях та скидання авто-інкрементних лічильників.
- 0. Вихід
    - Функціональність: Завершує виконання програми.

2. Підменю пошуку (Пункт 18)

- 1. Пошук виробників за кількістю препаратів
    - Функціональність: Запитує у користувача шаблон країни (LIKE), точну форму препарату (=) та мінімальну кількість препаратів (>=). Виконує запит з JOIN, WHERE, GROUP BY та HAVING[6]. Виводить результат та час виконання.
- 2. Пошук речовин за температурою зберігання препаратів
    - Функціональність: Запитує шаблон назви речовини (LIKE) та числовий діапазон температур (BETWEEN). Виконує запит, що об'єднує 3 таблиці, витягуючи числа з рядка storage_conditions. Виводить результат та час виконання.
- 3. Пошук препаратів за описом речовин у складі
    - Функціональність: Запитує шаблон назви препарату (LIKE), точну форму (=) та шаблон опису діючої речовини (LIKE). Виконує запит, що об'єднує 3 таблиці. Виводить результат та час виконання.
- 0. Повернутись до головного меню
    - Функціональність: Повертає користувача до головного меню.

Мова програмування та використані бібліотеки

Мова програмування

Для реалізації програмного додатку було обрано мову **Python** (версія 3.12). Це високорівнева інтерпретована мова, що відрізняється чистим, лаконічним синтаксисом та потужною стандартною бібліотекою, що дозволило швидко розробити консольний додаток. Вибір мови відповідає вимогам розрахунково-графічної роботи.

Використані бібліотеки

У проекті були використані наступні ключові бібліотеки:

- **psycopg (версія 3):** Це основна бібліотека, яка виступає в ролі сучасного драйвера для взаємодії з СУБД **PostgreSQL** у Python. Вона використовується у **Model**-частині додатку для:
    - Встановлення та керування з'єднаннями з базою даних.
    - Виконання всіх SQL-запитів (SELECT, INSERT, UPDATE, DELETE, TRUNCATE).
    - Безпечної передачі параметрів у запити (захист від SQL-ін'єкцій).
    - Керування транзакціями (commit(), rollback()).
    - Перехоплення специфічних помилок PostgreSQL (напр., ForeignKeyViolation).
- **time:** Стандартна бібліотека Python. Вона була використана для реалізації пункту 3 деталізованого завдання — вимірювання часу виконання складних пошукових запитів у мілісекундах.

№1

1.1. Вилучення запису батьківської таблиці

Завдання: Продемонструвати неможливість вилучення запису з батьківської таблиці (manufacturers), якщо на нього посилаються записи з дочірньої таблиці (preparations).

Хід виконання:

1. Спочатку переконуємось, що в базі даних є виробник та препарати, які на нього посилаються.

```
------------------------------------------------
0. Вихід
Оберіть опцію [0-19]: 1

--- Список Виробників ---
ID    | Назва компанії                | Країна
-------------------------------------------------------
4     | BioSolutions AG               | Ukraine
2     | GeneGenics AG                 | Poland
1     | GeneLabs AG                   | Poland
3     | MediGenics Inc.               | USA
5     | NovaLabs AG                   | Japan
```

```
0. Вихід
Оберіть опцію [0-19]: 9

--- Список Препаратів ---
ID    | Торгова назва             | Форма           | Виробник
-----------------------------------------------------------------------------
2     | Aspirin-Eco               | Ointment        | NovaLabs AG
1     | Nurorin-Eco               | Ointment        | GeneLabs AG
5     | Streptoclav-Max           | Tablets         | BioSolutions AG
3     | Teraform-Max              | Injection       | GeneGenics AG
4     | Vibrofen-Max              | Ointment        | GeneLabs AG
```

2. Запускаємо операцію "4. Видалити виробника" та вводимо ID=1.

```
------------------------------------------------
0. Вихід
Оберіть опцію [0-19]: 4

--- Видалення Виробника ---
Введіть ID: 1

[ПОВІДОМЛЕННЯ] Неможливо видалити виробника, оскільки за ним закріплені препарати.
```

Пояснення причини помилки:

Операція DELETE була заблокована на рівні СУБД PostgreSQL.

1. **Причина:** Була спроба видалити запис ID=1 з батьківської таблиці manufacturers.
2. **Обмеження:** Дочірня таблиця preparations містить щонайменше один запис, у якого в стовпці manufacturer_id встановлено значення 1.
3. **Результат:** Це порушує **обмеження зовнішнього ключа (Foreign Key constraint)**. Видалення батьківського запису призвело б до появи "запису-сироти" в дочірній таблиці, що порушує **посилальну цілісність** (referential integrity) бази даних.
4. **Реакція програми:** Модуль model.py (функція delete_manufacturer) коректно перехопив виняток psycopg.errors.ForeignKeyViolation у блоці try...except та повернув контролеру повідомлення про помилку, яке було виведене користувачу.

1.2. Вставка запису в дочірню таблицю

Завдання: Продемонструвати неможливість вставки запису в дочірню таблицю (preparations), якщо вказаний manufacturer_id не існує в батьківській таблиці (manufacturers).

Хід виконання:

1. Запускаємо операцію "10. Додати новий препарат".
2. Вводимо дані для нового препарату, але для ID виробника вказуємо свідомо неіснуюче значення, наприклад, 9999.

```
--------------------------------------------------
0. Вихід
Оберіть опцію [0-19]: 10

--- Додавання Нового Препарату ---
Введіть торгову назву: Септефріл
Введіть форму (таблетки, сироп...): таблетки
Введіть умови зберігання: +15градусів Цельсія

Оберіть ID виробника з списку:

--- Список Виробників ---
ID    | Назва компанії                | Країна
-----------------------------------------------------
4     | BioSolutions AG               | Ukraine
2     | GeneGenics AG                 | Poland
1     | GeneLabs AG                   | Poland
3     | MediGenics Inc.               | USA
5     | NovaLabs AG                   | Japan
Введіть ID виробника: 9999

[ПОВІДОМЛЕННЯ] Помилка! Виробника з таким ID не існує.
```

Пояснення причини помилки:

Операція INSERT була заблокована на рівні СУБД PostgreSQL.

1. **Причина:** Була спроба додати запис у дочірню таблицю preparations зі значенням manufacturer_id = 9999.
2. **Обмеження:** Батьківська таблиця manufacturers не містить жодного запису, де manufacturer_id дорівнює 9999.
3. **Результат:** Це також порушує **обмеження зовнішнього ключа**. Неможливо створити "дочірній" запис, який посилається на неіснуючий "батьківський".
4. **Реакція програми:** Модуль model.py (функція add_preparation) аналогічно перехопив помилку psycopg.errors.ForeignKeyViolation та повернув відповідне повідомлення користувачу.

№2

Відповідно до завдання, у програмі реалізовано функцію пакетної генерації випадкових, але валідних даних засобами СУБД PostgreSQL, а не циклами Python. Користувач вводить бажану кількість записів для кожної таблиці.

2.1. Копії SQL-запитів для генерації

Нижче наведено SQL-запити, що використовуються у модулі model.py для генерації даних. Параметр %s у кожному запиті замінюється на число, введене користувачем.

1. generate_manufacturers (Генерація виробників):

SQL

```
INSERT INTO manufacturers (company_name, country)
SELECT
   (ARRAY['Astra', 'Bio', 'Medi', 'Pharma', 'Gene', 'Nova', 'Hemo', 'Zene'])[trunc(random()*8+1)] ||
   (ARRAY['Core', 'Tech', 'Labs', 'Solutions', 'Genics', 'Life', 'Farm'])[trunc(random()*7+1)] || ' ' ||
   (ARRAY['Inc.', 'AG', 'LLC', 'Group'])[trunc(random()*4+1)],

   (ARRAY['USA', 'Germany', 'Ukraine', 'India', 'Poland', 'Switzerland', 'Japan'])[trunc(random()*7 + 1)]
FROM generate_series(1, %s) AS s(id);
```

2. generate_active_substances (Генерація діючих речовин):

SQL

```
INSERT INTO active_substances (substance_name, description)
SELECT
   (ARRAY['Aceta', 'Ibu', 'Levo', 'Dextra', 'Meto', 'Serta', 'Amlo', 'Ome', 'Lisi', 'Atorva'])[trunc(random()*10+1)] ||
   (ARRAY['profen', 'minophen', 'cetin', 'line', 'dipine', 'prazole', 'nacin', 'xetil', 'pril', 'statin'])[trunc(random()*10+1)],

   (ARRAY[
      'Non-steroidal anti-inflammatory drug (NSAID)',
      'Beta-blocker for hypertension',
      'Proton pump inhibitor (PPI)',
      'SSRI antidepressant',
      'Analgesic and antipyretic',
      'Calcium channel blocker',
      'ACE inhibitor'
   ])[trunc(random()*7+1)] || ' (Gen ' || s.id || ')'
FROM generate_series(1, %s) AS s(id);
```

3. generate_preparations (Генерація препаратів): *Цей запит коректно обирає випадковий, але існуючий manufacturer_id для кожного нового запису, забезпечуючи цілісність даних.*

SQL

```
INSERT INTO preparations (trade_name, form, storage_conditions, manufacturer_id)
SELECT
   (ARRAY['Nuro', 'Pana', 'Aspi', 'Vibro', 'Strepto', 'Tera', 'Amoxi', 'Mezy'])[trunc(random()*8+1)] ||
   (ARRAY['fen', 'dol', 'rin', 'cil', 'flu', 'sil', 'clav', 'form'])[trunc(random()*8+1)] || '-' ||
   (ARRAY['D', 'Forte', 'Max', 'Eco'])[trunc(random()*4+1)],

   (ARRAY['Tablets', 'Syrup', 'Capsules', 'Ointment', 'Injection'])[trunc(random()*5 + 1)],
   'Store at ' || trunc(random()*20 + 5) || 'C',

   (SELECT manufacturer_id FROM manufacturers WHERE manufacturer_id > 0 OR g.id > 0 ORDER BY random()
LIMIT 1)
FROM generate_series(1, %s) AS g(id);
```

4. generate_compositions (Генерація складу N:M): *Цей запит генерує пари (preparation_id, substance_id) і використовує ON CONFLICT ... DO NOTHING для автоматичного уникнення помилок дублювання складеного первинного ключа.*

SQL

```
INSERT INTO preparation_composition (preparation_id, substance_id, dosage)
SELECT
   (SELECT preparation_id FROM preparations WHERE preparation_id > 0 OR g.id > 0 ORDER BY random()
LIMIT 1),
   (SELECT substance_id FROM active_substances WHERE substance_id > 0 OR g.id > 0 ORDER BY random()
LIMIT 1),
   trunc(random()*500 + 50)::text || ' mg'
FROM generate_series(1, %s) g(id)
ON CONFLICT (preparation_id, substance_id) DO NOTHING;
```

2.2. Ілюстрації згенерованих даних

Нижче наведено фрагменти даних, отримані з таблиць після виконання пакетної генерації (наприклад, 100 записів).

```
--- Список Виробників ---
ID    | Назва компанії                  | Країна
-------------------------------------------------------------
67    | AstraCore AG                    | Germany
59    | AstraCore AG                    | Poland
74    | AstraCore Group                 | USA
53    | AstraCore Inc.                  | Germany
27    | AstraCore Inc.                  | Japan
62    | AstraCore LLC                   | Japan
24    | AstraFarm AG                    | USA
26    | AstraFarm Group                 | Ukraine
21    | AstraFarm Group                 | Japan
35    | AstraGenics LLC                 | India
47    | AstraGenics LLC                 | Ukraine
85    | AstraLabs Inc.                  | India
60    | AstraSolutions Group            | Japan
49    | AstraSolutions Group            | Switzerland
7     | AstraTech Group                 | Germany
```

```
--- Список Діючих Речовин ---
ID    | Назва речовини                  | Опис
-----------------------------------------------------------------------------
84    | Acetacetin                      | ACE inhibitor (Gen 84)
17    | Acetacetin                      | ACE inhibitor (Gen 17)
96    | Acetadipine                     | Beta-blocker for hypertension (Gen 96)
72    | Acetadipine                     | Beta-blocker for hypertension (Gen 72)
43    | Acetanacin                      | Non-steroidal anti-inflammatory drug (NSAID) (Gen 43)
75    | Acetaprofen                     | Proton pump inhibitor (PPI) (Gen 75)
40    | Acetaprofen                     | Proton pump inhibitor (PPI) (Gen 40)
60    | Amlodipine                      | SSRI antidepressant (Gen 60)
14    | Amloline                        | Non-steroidal anti-inflammatory drug (NSAID) (Gen 14)
92    | Amlominophen                    | ACE inhibitor (Gen 92)
30    | Amlonacin                       | Calcium channel blocker (Gen 30)
58    | Amloprazole                     | Non-steroidal anti-inflammatory drug (NSAID) (Gen 58)
38    | Atorvacetin                     | SSRI antidepressant (Gen 38)
66    | Atorvaline                      | SSRI antidepressant (Gen 66)
7     | Atorvanacin                     | Analgesic and antipyretic (Gen 7)
```

```
--- Список Препаратів ---
ID    | Торгова назва              | Форма           | Виробник
-------------------------------------------------------------------------------
82    | Amoxicil-Max               | Syrup           | GeneFarm Group
63    | Amoxiclav-Eco              | Syrup           | NovaLife LLC
30    | Amoxidol-Eco               | Syrup           | MediTech AG
99    | Amoxifen-D                 | Injection       | NovaGenics Inc.
77    | Amoxifen-D                 | Tablets         | NovaSolutions Inc.
80    | Amoxifen-Eco               | Tablets         | MediTech AG
74    | Amoxifen-Max               | Syrup           | NovaTech Group
32    | Amoxiflu-D                 | Tablets         | MediSolutions Group
18    | Amoxiflu-Eco               | Capsules        | GeneFarm Group
8     | Amoxiflu-Forte             | Injection       | GeneFarm LLC
45    | Amoxiflu-Max               | Capsules        | NovaTech AG
96    | Amoxirin-Eco               | Tablets         | AstraCore AG
52    | Amoxisil-D                 | Injection       | ZeneLabs Group
58    | Aspicil-Eco                | Injection       | MediSolutions Group
31    | Aspicil-Max                | Capsules        | MediGenics AG
6     | Aspicil-Max                | Ointment        | HemoLife Group
```

```
--- Склад Препаратів ---
ID Преп. | Назва Препарату          | ID Реч.  | Назва Речовини           | Дозування
-------------------------------------------------------------------------------
82       | Amoxicil-Max             | 38       | Atorvacetin              | 329 mg
82       | Amoxicil-Max             | 41       | Sertastatin              | 358 mg
99       | Amoxifen-D               | 28       | Lisiminophen             | 547 mg
99       | Amoxifen-D               | 73       | Lisistatin               | 115 mg
74       | Amoxifen-Max             | 64       | Metoprofen               | 276 mg
74       | Amoxifen-Max             | 3        | Sertaxetil               | 330 mg
8        | Amoxiflu-Forte           | 84       | Acetacetin               | 259 mg
52       | Amoxisil-D               | 76       | Atorvaprazole            | 317 mg
6        | Aspicil-Max              | 62       | Metostatin               | 302 mg
19       | Aspiflu-Forte            | 34       | Lisipril                 | 548 mg
19       | Aspiflu-Forte            | 71       | Sertaline                | 313 mg
64       | Aspiflu-Max              | 30       | Amlonacin                | 268 mg
64       | Aspiflu-Max              | 50       | Dextracetin              | 479 mg
93       | Aspirin-D                | 20       | Ibudipine                | 317 mg
93       | Aspirin-D                | 73       | Lisistatin               | 502 mg
76       | Mezycil-D                | 83       | Dextracetin              | 323 mg
12       | Mezycil-Forte            | 87       | Lisiline                 | 403 mg
```

№3

Відповідно до завдання, у програмі реалізовано 3 складні пошукові запити, що об'єднують дані з декількох таблиць, використовують фільтрацію (WHERE), групування (GROUP BY) та агрегатні функції. Також реалізовано вимірювання часу виконання кожного запиту.

3.1. Запит 1: Пошук виробників за кількістю препаратів

Завдання: Знайти виробників з певної країни (шаблон ILIKE), які випускають препарати у конкретній формі (=), та їхня загальна кількість препаратів цієї форми більша за N.

SQL-запит (з model.py): *Параметри %s приймають значення, введені користувачем.*

SQL

```
SELECT m.company_name, m.country, COUNT(p.preparation_id) as preparation_count
FROM manufacturers m
JOIN preparations p ON m.manufacturer_id = p.preparation_id
WHERE m.country ILIKE %s AND p.form = %s
GROUP BY m.manufacturer_id, m.company_name, m.country
HAVING COUNT(p.preparation_id) > %s
ORDER BY preparation_count DESC;
```

```
--- Пошук виробників за кількістю препаратів ---
Введіть шаблон країни (напр. 'Ukr%'): Ukr%
Введіть точну форму препарату (напр. 'Tablets'): Tablets
Мінімальна кількість препаратів (напр. 5): 3

--- Результати Пошуку ---
company_name  | country | preparation_count
------------------------------------------
BioLabs AG    | Ukraine | 5
AstraFarm LLC | Ukraine | 5
BioFarm Group | Ukraine | 5
------------------------------------------
Знайдено рядків: 3. Час виконання: 0.00 мс
```

3.2. Запит 2: Пошук речовин за температурою зберігання

Завдання: Знайти діючі речовини (шаблон ILIKE), які входять до складу препаратів, що вимагають зберігання у певному числовому діапазоні температур (BETWEEN).

SQL-запит (з model.py): *Запит витягує числа з текстового поля storage_conditions за допомогою regexp_replace.*

```
SQL
SELECT s.substance_name, s.description, COUNT(p.preparation_id) as used_in_preparations
FROM active_substances s
JOIN preparation_composition pc ON s.substance_id = pc.substance_id
JOIN preparations p ON pc.preparation_id = p.preparation_id
WHERE s.substance_name ILIKE %s
  AND CAST(regexp_replace(p.storage_conditions, '[^0-9]', '', 'g') AS INTEGER) BETWEEN %s AND %s
GROUP BY s.substance_id, s.substance_name, s.description
ORDER BY used_in_preparations DESC;
```

```
Мін. температура зберігання (напр. 5): 5
Макс. температура зберігання (напр. 20): 100

--- Результати Пошуку ---
substance_name | description                                                    | used_in_preparations
-------------------------------------------------------------------------------------------------
Omeprofen      | SSRI antidepressant (Gen 35)                                   | 18
Sertaprofen    | Non-steroidal anti-inflammatory drug (NSAID) (Gen 343)         | 17
Sertaprofen    | Analgesic and antipyretic (Gen 397)                            | 16
Sertaprofen    | Analgesic and antipyretic (Gen 223)                            | 15
Omeprofen      | Analgesic and antipyretic (Gen 114)                            | 14
Lisiprofen     | SSRI antidepressant (Gen 248)                                  | 14
Atorvaprofen   | ACE inhibitor (Gen 331)                                        | 14
Acetaprofen    | Beta-blocker for hypertension (Gen 444)                        | 14
Metoprofen     | Beta-blocker for hypertension (Gen 455)                        | 14
Omeprofen      | SSRI antidepressant (Gen 429)                                  | 13
Sertaprofen    | SSRI antidepressant (Gen 56)                                   | 13
Dextraprofen   | SSRI antidepressant (Gen 66)                                   | 13
Amloprofen     | SSRI antidepressant (Gen 326)                                  | 13
Amloprofen     | SSRI antidepressant (Gen 377)                                  | 13
Metoprofen     | ACE inhibitor (Gen 3)                                          | 13
Ibuprofen      | SSRI antidepressant (Gen 134)                                  | 13
Metoprofen     | Calcium channel blocker (Gen 10)                               | 12
Omeprofen      | SSRI antidepressant (Gen 50)                                   | 12
Atorvaprofen   | Beta-blocker for hypertension (Gen 240)                        | 12
Dextraprofen   | Proton pump inhibitor (PPI) (Gen 350)                          | 11
Omeprofen      | ACE inhibitor (Gen 163)                                        | 11
Acetaprofen    | ACE inhibitor (Gen 159)                                        | 10
Sertaprofen    | Analgesic and antipyretic (Gen 300)                            | 10
Sertaprofen    | Calcium channel blocker (Gen 178)                              | 10
Omeprofen      | Beta-blocker for hypertension (Gen 72)                         | 10
Omeprofen      | Proton pump inhibitor (PPI) (Gen 497)                          | 10
Sertaprofen    | Beta-blocker for hypertension (Gen 104)                        | 10
Omeprofen      | ACE inhibitor (Gen 473)                                        | 9
Ibuprofen      | Analgesic and antipyretic (Gen 490)                            | 9
Atorvaprofen   | Beta-blocker for hypertension (Gen 100)                        | 9
Amloprofen     | Proton pump inhibitor (PPI) (Gen 298)                          | 9
Amloprofen     | Calcium channel blocker (Gen 151)                              | 9
Ibuprofen      | Beta-blocker for hypertension (Gen 210)                        | 8
Lisiprofen     | Beta-blocker for hypertension (Gen 137)                        | 8
Atorvaprofen   | Beta-blocker for hypertension (Gen 107)                        | 8
Omeprofen      | SSRI antidepressant (Gen 69)                                   | 8
Dextraprofen   | Calcium channel blocker (Gen 421)                              | 7
Acetaprofen    | ACE inhibitor (Gen 184)                                        | 7
Atorvaprofen   | Calcium channel blocker (Gen 36)                               | 7
Dextraprofen   | Non-steroidal anti-inflammatory drug (NSAID) (Gen 129)         | 7
Atorvaprofen   | SSRI antidepressant (Gen 330)                                  | 6
Levoprofen     | Non-steroidal anti-inflammatory drug (NSAID) (Gen 293)         | 6
Omeprofen      | Proton pump inhibitor (PPI) (Gen 282)                          | 6
Metoprofen     | Beta-blocker for hypertension (Gen 458)                        | 5
-------------------------------------------------------------------------------------------------
Знайдено рядків: 44. Час виконання: 31.00 мс
```

3.3. Запит 3: Пошук препаратів за описом речовин

Завдання: Знайти препарати (шаблон ILIKE та точна форма =), які у своєму складі містять речовини з певним описом (шаблон ILIKE).

SQL-запит (з model.py):

SQL
```
SELECT p.trade_name, p.form, COUNT(s.substance_id) as matching_substances
FROM preparations p
JOIN preparation_composition pc ON p.preparation_id = pc.preparation_id
JOIN active_substances s ON pc.substance_id = s.substance_id
WHERE p.trade_name ILIKE %s AND p.form = %s AND s.description ILIKE %s
GROUP BY p.preparation_id, p.trade_name, p.form
ORDER BY matching_substances DESC;
```

```
Введіть шаблон назви препарату (напр. 'Nuro%'): Nuro%
Введіть точну форму препарату (напр. 'Tablets'): Tablets
Введіть шаблон опису речовини (напр. '%anti-inflammatory%'):

--- Результати Пошуку ---
trade_name     | form    | matching_substances
-----------------------------------------------
Nurosil-D      | Tablets | 6
Nuroform-Max   | Tablets | 6
Nuroform-D     | Tablets | 5
Nurosil-Eco    | Tablets | 5
Nuroclav-Eco   | Tablets | 5
Nuroform-D     | Tablets | 5
Nuroclav-Eco   | Tablets | 5
Nurocil-Eco    | Tablets | 4
Nurofen-Forte  | Tablets | 4
Nuroflu-Forte  | Tablets | 4
Nuroform-D     | Tablets | 3
Nuroform-Eco   | Tablets | 3
Nuroform-Eco   | Tablets | 3
Nurocil-D      | Tablets | 3
Nuroform-Eco   | Tablets | 3
```

№4

Програмний код організовано згідно з шаблоном **Model-View-Controller (MVC)**. Модель, Подання та Контролер реалізовані в окремих файлах Python.

4.1. Ілюстрація програмного коду модуля "Model"

Відповідно до вимог, нижче наведено повний лістинг коду модуля model.py. Цей файл інкапсулює всю логіку взаємодії з базою даних PostgreSQL. Він містить лише функції, що виконують SQL-запити, і не містить жодного коду, пов'язаного з інтерфейсом користувача (print/input).

```
Python
import psycopg
from psycopg import errors
import time

DB_CONFIG = {
    "dbname": "Ptaha",
    "user": "postgres",
    "password": "password",
    "host": "localhost",
    "port": "5432"
}

def get_db_connection():
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            return conn
    except psycopg.Error as e:
        print(f"Помилка підключення до бази даних: {e}")
        return None

# --- Manufacturers ---
```

```
def get_all_manufacturers():
    query = "SELECT manufacturer_id, company_name, country FROM manufacturers ORDER BY
company_name"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query)
                return cur.fetchall()
    except psycopg.Error as e:
        print(f"Помилка при отриманні виробників: {e}")
        return []

def add_manufacturer(company_name, country):
    query = "INSERT INTO manufacturers (company_name, country) VALUES (%s, %s)"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (company_name, country))
                conn.commit()
                return True
    except psycopg.Error as e:
        print(f"Помилка при доданні виробника: {e}")
        return False

def get_manufacturer_by_id(manufacturer_id):
    query = "SELECT manufacturer_id, company_name, country FROM manufacturers WHERE
manufacturer_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (manufacturer_id,))
                return cur.fetchone()
    except psycopg.Error as e:
        print(f"Помилка при отриманні виробника: {e}")
        return None

def update_manufacturer(manufacturer_id, company_name, country):
    query = "UPDATE manufacturers SET company_name = %s, country = %s WHERE
manufacturer_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (company_name, country, manufacturer_id))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Виробника з таким ID не знайдено для оновлення.")
                return (True, "Дані виробника успішно оновлено.")
    except psycopg.Error as e:
        print(f"Помилка при оновленні виробника: {e}")
        return (False, f"Виникла помилка: {e}")

def delete_manufacturer(manufacturer_id):
    query = "DELETE FROM manufacturers WHERE manufacturer_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (manufacturer_id,))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Виробника з таким ID не знайдено.")
                return (True, "Виробника успішно видалено.")
    except errors.ForeignKeyViolation:
```

```
        return (False, "Неможливо видалити виробника, оскільки за ним закріплені
препарати.")
    except psycopg.Error as e:
        print(f"Помилка при видаленні виробника: {e}")
        return (False, f"Виникла помилка: {e}")

# --- Active Substances ---

def get_all_active_substances():
    query = "SELECT substance_id, substance_name, description FROM active_substances
ORDER BY substance_name"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query)
                return cur.fetchall()
    except psycopg.Error as e:
        print(f"Помилка при отриманні діючих речовин: {e}")
        return []

def add_active_substance(substance_name, description):
    query = "INSERT INTO active_substances (substance_name, description) VALUES (%s,
%s)"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (substance_name, description))
                conn.commit()
                return True
    except psycopg.Error as e:
        print(f"Помилка при доданні діючої речовини: {e}")
        return False

def get_active_substance_by_id(substance_id):
    query = "SELECT substance_id, substance_name, description FROM active_substances
WHERE substance_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (substance_id,))
                return cur.fetchone()
    except psycopg.Error as e:
        print(f"Помилка при отриманні діючої речовини: {e}")
        return None

def update_active_substance(substance_id, substance_name, description):
    query = "UPDATE active_substances SET substance_name = %s, description = %s WHERE
substance_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (substance_name, description, substance_id))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Речовину з таким ID не знайдено для оновлення.")
                return (True, "Дані речовини успішно оновлено.")
    except psycopg.Error as e:
        print(f"Помилка при оновленні речовини: {e}")
        return (False, f"Виникла помилка: {e}")

def delete_active_substance(substance_id):
    query = "DELETE FROM active_substances WHERE substance_id = %s"
```

```
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (substance_id,))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Речовину з таким ID не знайдено.")
                return (True, "Речовину успішно видалено.")
    except errors.ForeignKeyViolation:
        return (False, "Неможливо видалити речовину, оскільки вона входить до складу
препаратів.")
    except psycopg.Error as e:
        print(f"Помилка при видаленні речовини: {e}")
        return (False, f"Виникла помилка: {e}")

# --- Preparations ---

def get_all_preparations():
    query = """
        SELECT   p.preparation_id,   p.trade_name,   p.form,   p.storage_conditions,
m.company_name
        FROM preparations p
        JOIN manufacturers m ON p.manufacturer_id = m.manufacturer_id
        ORDER BY p.trade_name
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query)
                return cur.fetchall()
    except psycopg.Error as e:
        print(f"Помилка при отриманні препаратів: {e}")
        return []

def add_preparation(trade_name, form, storage_conditions, manufacturer_id):
    query  =  "INSERT  INTO  preparations  (trade_name,  form,  storage_conditions,
manufacturer_id) VALUES (%s, %s, %s, %s)"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query,     (trade_name,     form,     storage_conditions,
manufacturer_id))
                conn.commit()
                return (True, "Препарат успішно додано.")
    except errors.ForeignKeyViolation:
        return (False, "Помилка! Виробника з таким ID не існує.")
    except psycopg.Error as e:
        print(f"Помилка при доданні препарату: {e}")
        return (False, f"Виникла помилка: {e}")

def get_preparation_by_id(preparation_id):
    query = """
        SELECT   p.preparation_id,   p.trade_name,   p.form,   p.storage_conditions,
p.manufacturer_id, m.company_name
        FROM preparations p
        JOIN manufacturers m ON p.manufacturer_id = m.manufacturer_id
        WHERE p.preparation_id = %s
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (preparation_id,))
```

```
                return cur.fetchone()
    except psycopg.Error as e:
        print(f"Помилка при отриманні препарату: {e}")
        return None

def update_preparation(preparation_id, trade_name, form, storage_conditions,
manufacturer_id):
    query = """
        UPDATE preparations
        SET trade_name = %s, form = %s, storage_conditions = %s, manufacturer_id = %s
        WHERE preparation_id = %s
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (trade_name, form, storage_conditions,
manufacturer_id, preparation_id))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Препарат з таким ID не знайдено для оновлення.")
                return (True, "Дані препарату успішно оновлено.")
    except errors.ForeignKeyViolation:
        return (False, "Помилка! Виробника з таким ID не існує.")
    except psycopg.Error as e:
        print(f"Помилка при оновленні препарату: {e}")
        return (False, f"Виникла помилка: {e}")

def delete_preparation(preparation_id):
    query = "DELETE FROM preparations WHERE preparation_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (preparation_id,))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Препарат з таким ID не знайдено.")
                return (True, "Препарат успішно видалено.")
    except errors.ForeignKeyViolation:
        return (False, "Неможливо видалити препарат, оскільки він має зв'язані дані
(склад).")
    except psycopg.Error as e:
        print(f"Помилка при видаленні препарату: {e}")
        return (False, f"Виникла помилка: {e}")

# --- Preparation Composition (N:M Table) ---

def get_all_compositions():
    query = """
        SELECT
            p.preparation_id, p.trade_name,
            s.substance_id, s.substance_name,
            pc.dosage
        FROM preparation_composition pc
        JOIN preparations p ON pc.preparation_id = p.preparation_id
        JOIN active_substances s ON pc.substance_id = s.substance_id
        ORDER BY p.trade_name, s.substance_name
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query)
                return cur.fetchall()
```

```
    except psycopg.Error as e:
        print(f"Помилка при отриманні складу препаратів: {e}")
        return []

def get_composition_by_id(preparation_id, substance_id):
    query = "SELECT preparation_id, substance_id, dosage FROM preparation_composition
WHERE preparation_id = %s AND substance_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (preparation_id, substance_id))
                return cur.fetchone()
    except psycopg.Error as e:
        print(f"Помилка при отриманні запису зі складу: {e}")
        return None

def add_composition(preparation_id, substance_id, dosage):
    query = "INSERT INTO preparation_composition (preparation_id, substance_id,
dosage) VALUES (%s, %s, %s)"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (preparation_id, substance_id, dosage))
                conn.commit()
                return (True, "Запис до складу успішно додано.")
    except errors.UniqueViolation:
        return (False, "Помилка! Ця речовина вже є у складі цього препарату.")
    except errors.ForeignKeyViolation:
        return (False, "Помилка! Вказано неіснуючий ID препарату або речовини.")
    except psycopg.Error as e:
        print(f"Помилка при доданні до складу: {e}")
        return (False, f"Виникла помилка: {e}")

def update_composition(preparation_id, substance_id, new_dosage):
    query = "UPDATE preparation_composition SET dosage = %s WHERE preparation_id = %s
AND substance_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (new_dosage, preparation_id, substance_id))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Запис з такими ID не знайдено для оновлення.")
                return (True, "Дозування успішно оновлено.")
    except psycopg.Error as e:
        print(f"Помилка при оновленні складу: {e}")
        return (False, f"Виникла помилка: {e}")

def delete_composition(preparation_id, substance_id):
    query = "DELETE FROM preparation_composition WHERE preparation_id = %s AND
substance_id = %s"
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (preparation_id, substance_id))
                conn.commit()
                if cur.rowcount == 0:
                    return (False, "Запис з такими ID не знайдено.")
                return (True, "Запис зі складу успішно видалено.")
    except psycopg.Error as e:
        print(f"Помилка при видаленні зі складу: {e}")
        return (False, f"Виникла помилка: {e}")
```

```
# --- Data Generation & Clearing ---

def clear_all_data():
    query = """
        TRUNCATE
            manufacturers,
            active_substances,
            preparations,
            preparation_composition
        RESTART IDENTITY CASCADE;
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query)
                conn.commit()
                return (True, "Всі дані з таблиць було успішно видалено, лічильники
скинуто.")
    except psycopg.Error as e:
        print(f"Помилка при очищенні таблиць: {e}")
        return (False, f"Виникла помилка: {e}")

def generate_manufacturers(count):
    query = """
        INSERT INTO manufacturers (company_name, country)
        SELECT
            (ARRAY['Astra',  'Bio',  'Medi',  'Pharma',  'Gene',  'Nova',  'Hemo',
'Zene'])[trunc(random()*8+1)] ||
            (ARRAY['Core',   'Tech',   'Labs',   'Solutions',   'Genics',   'Life',
'Farm'])[trunc(random()*7+1)] || ' ' ||
            (ARRAY['Inc.', 'AG', 'LLC', 'Group'])[trunc(random()*4+1)],

            (ARRAY['USA',  'Germany',  'Ukraine',  'India',  'Poland',  'Switzerland',
'Japan'])[trunc(random()*7 + 1)]
        FROM generate_series(1, %s) AS s(id);
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (count,))
                conn.commit()
                return (True, f"Успішно згенеровано {count} виробників.")
    except psycopg.Error as e:
        print(f"Помилка генерації виробників: {e}")
        return (False, f"Помилка: {e}")

def generate_active_substances(count):
    query = """
        INSERT INTO active_substances (substance_name, description)
        SELECT
            (ARRAY['Aceta', 'Ibu', 'Levo', 'Dextra', 'Meto', 'Serta', 'Amlo', 'Ome',
'Lisi', 'Atorva'])[trunc(random()*10+1)] ||
            (ARRAY['profen',  'minophen',  'cetin',  'line',  'dipine',  'prazole',
'nacin', 'xetil', 'pril', 'statin'])[trunc(random()*10+1)],

            (ARRAY[
                'Non-steroidal anti-inflammatory drug (NSAID)',
                'Beta-blocker for hypertension',
                'Proton pump inhibitor (PPI)',
                'SSRI antidepressant',
                'Analgesic and antipyretic',
```

```
                'Calcium channel blocker',
                'ACE inhibitor'
            ])[trunc(random()*7+1)] || ' (Gen ' || s.id || ')'
        FROM generate_series(1, %s) AS s(id);
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (count,))
                conn.commit()
                return (True, f"Успішно згенеровано {count} діючих речовин.")
    except psycopg.Error as e:
        print(f"Помилка генерації речовин: {e}")
        return (False, f"Помилка: {e}")

def generate_preparations(count):
    query = """
        INSERT INTO preparations (trade_name, form, storage_conditions,
manufacturer_id)
        SELECT
            (ARRAY['Nuro', 'Pana', 'Aspi', 'Vibro', 'Strepto', 'Tera', 'Amoxi',
'Mezy'])[trunc(random()*8+1)] ||
            (ARRAY['fen', 'dol', 'rin', 'cil', 'flu', 'sil', 'clav',
'form'])[trunc(random()*8+1)] || '-' ||
            (ARRAY['D', 'Forte', 'Max', 'Eco'])[trunc(random()*4+1)],

            (ARRAY['Tablets', 'Syrup', 'Capsules', 'Ointment',
'Injection'])[trunc(random()*5 + 1)],
            'Store at ' || trunc(random()*20 + 5) || 'C',

            (SELECT manufacturer_id FROM manufacturers WHERE manufacturer_id > 0 OR
g.id > 0 ORDER BY random() LIMIT 1)
        FROM generate_series(1, %s) AS g(id);
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                cur.execute(query, (count,))
                conn.commit()
                return (True, f"Успішно згенеровано {count} препаратів.")
    except psycopg.Error as e:
        print(f"Помилка генерації препаратів: {e}")
        if "null value in column \"manufacturer_id\"" in str(e) or "is not present in
table \"manufacturers\"" in str(e):
            return (False, "Помилка: Неможливо згенерувати препарати. Таблиця
виробників порожня.")
        return (False, f"Помилка: {e}")

def generate_compositions(count):
    query = """
        INSERT INTO preparation_composition (preparation_id, substance_id, dosage)
        SELECT
            (SELECT preparation_id FROM preparations WHERE preparation_id > 0 OR g.id
> 0 ORDER BY random() LIMIT 1),
            (SELECT substance_id FROM active_substances WHERE substance_id > 0 OR g.id
> 0 ORDER BY random() LIMIT 1),
            trunc(random()*500 + 50)::text || ' mg'
        FROM generate_series(1, %s) g(id)
        ON CONFLICT (preparation_id, substance_id) DO NOTHING;
    """
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
```

```
            with conn.cursor() as cur:
                cur.execute(query, (count,))
                conn.commit()
                return (True, f"Успішно згенеровано {count} записів у склад (дублікати
проігноровано).")
    except psycopg.Error as e:
        print(f"Помилка генерації складу: {e}")
        if "null value in column" in str(e) or "is not present in table" in str(e):
             return (False, "Помилка: Неможливо згенерувати склад. Таблиці препаратів
або речовин порожні.")
        return (False, f"Помилка: {e}")

# --- Search Queries (RGR Task) ---

def execute_search_query(query, params=()):
    try:
        with psycopg.connect(**DB_CONFIG) as conn:
            with conn.cursor() as cur:
                start_time = time.monotonic()
                cur.execute(query, params)
                results = cur.fetchall()
                end_time = time.monotonic()

                duration_ms = (end_time - start_time) * 1000
                column_names = [col.name for col in cur.description]

                return (results, column_names, duration_ms, None)
    except psycopg.Error as e:
        print(f"Помилка виконання пошукового запиту: {e}")
        return (None, None, 0, str(e))

def search_manufacturers_by_prep_count(country_pattern, form_exact, min_prep_count):
    query = """
        SELECT m.company_name, m.country, COUNT(p.preparation_id) as preparation_count
        FROM manufacturers m
        JOIN preparations p ON m.manufacturer_id = p.manufacturer_id
        WHERE m.country ILIKE %s AND p.form = %s
        GROUP BY m.manufacturer_id, m.company_name, m.country
        HAVING COUNT(p.preparation_id) > %s
        ORDER BY preparation_count DESC;
    """
    return execute_search_query(query, (country_pattern, form_exact, min_prep_count))

def search_substances_by_storage_temp(substance_pattern, min_temp, max_temp):
    query = """
        SELECT    s.substance_name,    s.description,    COUNT(p.preparation_id)    as
used_in_preparations
        FROM active_substances s
        JOIN preparation_composition pc ON s.substance_id = pc.substance_id
        JOIN preparations p ON pc.preparation_id = p.preparation_id
        WHERE s.substance_name ILIKE %s
          AND CAST(regexp_replace(p.storage_conditions, '[^0-9]', '', 'g') AS INTEGER)
BETWEEN %s AND %s
        GROUP BY s.substance_id, s.substance_name, s.description
        ORDER BY used_in_preparations DESC;
    """
    return execute_search_query(query, (substance_pattern, min_temp, max_temp))

def         search_preparations_by_composition(prep_name_pattern,         form_exact,
substance_desc_pattern):
    query = """
        SELECT p.trade_name, p.form, COUNT(s.substance_id) as matching_substances
```

```
        FROM preparations p
        JOIN preparation_composition pc ON p.preparation_id = pc.preparation_id
        JOIN active_substances s ON pc.substance_id = s.substance_id
        WHERE p.trade_name ILIKE %s AND p.form = %s AND s.description ILIKE %s
        GROUP BY p.preparation_id, p.trade_name, p.form
        ORDER BY matching_substances DESC;
    """
    return execute_search_query(query, (prep_name_pattern, form_exact,
substance_desc_pattern))
```

4.2. Короткий опис функцій модуля "Model"

Відповідно до вимог, ось опис основних груп функцій, реалізованих у model.py.

Загальні функції

- get_db_connection(): Службова функція, що створює та повертає нове з'єднання з базою даних, використовуючи конфігурацію з DB_CONFIG.
- execute_search_query(query, params): Універсальна функція-обгортка для виконання пошукових запитів. Вона приймає текст SQL-запиту та параметри, вимірює час виконання, та повертає результат, назви стовпців і час у мілісекундах.

CRUD (Create, Read, Update, Delete)

Цей блок функцій реалізує повний набір операцій для кожної з 4-х таблиць бази даних.

- get_all...(): (напр., get_all_manufacturers()) Виконує SELECT * для отримання всіх записів з таблиці.
- get..._by_id(): (напр., get_preparation_by_id()) Виконує SELECT для отримання одного запису за його первинним ключем.
- add...(): (напр., add_active_substance()) Виконує INSERT для додавання нового запису. Перехоплює помилки цілісності (напр., ForeignKeyViolation).
- update...(): (напр., update_composition()) Виконує UPDATE для оновлення існуючого запису за його ID.
- delete...(): (напр., delete_manufacturer()) Виконує DELETE для вилучення запису. Включає перехоплення errors.ForeignKeyViolation для запобігання видаленню батьківських записів, на які є посилання.

Генерація та Очищення

Цей блок реалізує завдання РГР щодо наповнення та очищення бази даних.

- clear_all_data(): Виконує TRUNCATE ... RESTART IDENTITY CASCADE для повного очищення всіх 4 таблиць та скидання лічильників авто-інкременту.
- generate_...(): (напр., generate_preparations()) Група з 4-х функцій, кожна з яких виконує пакетний INSERT ... SELECT ... FROM generate_series() для швидкої генерації великої кількості випадкових, але валідних даних засобами PostgreSQL.

Комплексний Пошук

Цей блок реалізує 3 спеціалізовані пошукові запити згідно з завданням РГР.

- search_manufacturers_by_prep_count(): Виконує запит з JOIN, WHERE, GROUP BY та HAVING для пошуку виробників.
- search_substances_by_storage_temp(): Виконує запит, що об'єднує 3 таблиці та використовує regexp_replace і CAST для фільтрації за числовим діапазоном з текстового поля.
- search_preparations_by_composition(): Виконує запит, що об'єднує 3 таблиці для пошуку препаратів за характеристиками їх складових.

Висновок

У ході виконання розрахунково-графічної роботи було досягнуто її головну мету — здобуття навичок програмування прикладних додатків для баз даних під управлінням СУБД PostgreSQL.

Для цього було розроблено консольний додаток мовою Python, який надає повноцінний інтерфейс для взаємодії з базою даних «Електронний довідник медичних препаратів».

Основні результати роботи:

1. **Архітектура MVC:** Програма була чітко структурована за шаблоном Model-View-Controller. Уся логіка взаємодії з базою даних, SQL-запити та керування транзакціями були повністю інкапсульовані в модулі model.py, що відокремило логіку даних від логіки представлення та керування.
2. **Реалізація CRUD та контроль цілісності:** Було реалізовано повний набір CRUD-операцій (створення, читання, оновлення, видалення) для всіх чотирьох таблиць бази даних. Успішно реалізовано механізми контролю посилальної цілісності: програма коректно перехоплює помилки ForeignKeyViolation та інформує користувача про неможливість видалення батьківського запису, на який є посилання, а

також про неможливість вставки дочірнього запису з неіснуючим зовнішнім ключем.

3. **Пакетна генерація даних:** Реалізовано функціонал для автоматичної генерації великої кількості випадкових, але валідних даних. Генерація відбувається безпосередньо засобами PostgreSQL (INSERT ... SELECT ... FROM generate_series()), що забезпечує високу продуктивність та коректне заповнення зовнішніх ключів.
4. **Комплексний пошук:** Розроблено три складні пошукові запити, що виконують вибірку даних з кількох таблиць одночасно. Запити підтримують фільтрацію за шаблоном (ILIKE) для рядкових даних та числовим діапазоном (BETWEEN), витягнутим з текстового поля. Також реалізовано вимірювання та відображення часу виконання кожного запиту в мілісекундах.

Усі поставлені завдання було виконано в повному обсязі. У процесі роботи було здобуто практичні навички роботи з бібліотекою psycopg, побудови SQL-запитів різної складності та реалізації додатків баз даних за шаблоном MVC.