**ЛИСТ ЗАВДАННЯ**

Метою даної курсової роботи є розробка бази даних для обліку житлового фонду. У процесі виконання роботи реалізується система, яка дозволяє зберігати, обробляти та аналізувати дані про житлові приміщення та їхніх мешканців.

Система повинна містити інформацію про вулиці, номери будинків, квартири, кількість кімнат, загальну площу, поверхи, а також дані про мешканців. До останніх належать ім’я, дата народження, дата прописки, інформація про пільги (відсоток оплати) та розмір оплати за проживання.

Основними функціями бази даних є можливість введення даних про квартири та мешканців, пошук за номером будинку або прізвищем мешканця, відображення квартир із визначеною кількістю кімнат або площею, що перевищує заданий параметр.

**У процесі виконання курсової роботи виконана така робота:**

* Проаналізувати предметну область, визначити основні сутності, їх атрибути та зв’язки між ними.
* Побудувати ER-модель, яка описує предметну область відповідно до бізнес-вимог.
* На основі ER-моделі створити реляційну схему бази даних із визначенням первинних та зовнішніх ключів, а також встановити обмеження для забезпечення цілісності даних.
* Реалізувати базу даних, використовуючи мову SQL: написати скрипти для створення таблиць, імпорту даних і виконання запитів.
* Розробити запити, які забезпечують виконання основного функціоналу бази даних, включаючи пошук, вибірки та обчислення.

**ЗМІСТ**

**ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА**

Дана курсова робота присвячена розробці бази даних для обліку житлового фонду. Житловий фонд включає інформацію про будинки, квартири та їх мешканців. Метою роботи є створення зручної системи для управління даними про житловий фонд, виконання різноманітних пошукових і аналітичних запитів.

Предметне середовище складається з таких елементів:

1. **Інформація про житловий фонд**:
   * **Вулиця** – назва, на якій знаходиться будинок.
   * **Номер будинку** – унікальний номер будинку на вулиці.
   * **Номер квартири** – унікальний номер квартири в межах будинку.
   * **Кількість кімнат** – характеристика квартири.
   * **Площа** – загальна площа квартири.
   * **Поверх** – номер поверху, на якому розташована квартира.
2. **Інформація про мешканців**:
   * **Ім’я** – ім’я мешканця.
   * **Дата народження** – особисті дані мешканця.
   * **Дата прописки** – дата реєстрації мешканця в квартирі.
   * **Чи користуються пільгами** – відмітка про наявність пільг.
   * **Відсоток оплати** – сума, яку мешканець сплачує, з урахуванням пільг.
3. **Розмір оплати** – загальна сума, яка сплачується за квартиру.

Функціональні можливості бази даних:

* **Додавання нових даних**: введення інформації про квартиру та мешканців.
* **Пошукові запити**:
  + Пошук квартири за номером будинку.
  + Пошук мешканця за прізвищем.
  + Відображення квартир із заданою кількістю кімнат або площею більшою за вказану.
* **Аналітичні запити**:
  + Підрахунок кількості мешканців у конкретному будинку.
  + Розрахунок площі, яка припадає на одного мешканця квартири.

База даних призначена для ефективного управління інформацією про житловий фонд, виконання запитів для пошуку та аналізу, а також автоматизації обліку даних.

**ПОСТАНОВКА ЗАВДАННЯ**

Метою курсової роботи є розробка бази даних для обліку житлового фонду, яка забезпечить ефективне зберігання, управління, пошук та аналіз інформації про квартири та їх мешканців.

Для досягнення цієї мети необхідно виконати такі завдання:

1. **Проаналізувати предметне середовище**:
   * Визначити основні сутності та їх атрибути (вулиця, будинок, квартира, мешканці).
   * Визначити зв’язки між сутностями.
2. **Розробити структуру бази даних**:
   * Побудувати ER-модель (діаграма зв’язків) предметного середовища.
   * Спроектувати реляційну схему бази даних, включаючи таблиці, первинні та зовнішні ключі.
3. **Реалізувати базу даних**:
   * Розробити SQL-скрипти для створення таблиць, зв’язків і обмежень.
   * Наповнити базу даних тестовими даними для перевірки її функціональності.
4. **Розробити функціональні можливості бази даних**:
   * **Введення даних** про квартири та мешканців.
   * **Пошук**:
     + За номером будинку.
     + За прізвищем мешканця.
     + Квартир із певною кількістю кімнат чи площею більше заданої.
   * **Аналітика**:
     + Підрахунок кількості мешканців у конкретному будинку.
     + Розрахунок площі, яка припадає на одного мешканця квартири.
5. **Забезпечити тестування системи**:
   * Перевірити коректність функціонування бази даних при виконанні запитів.
   * Перевірити узгодженість даних та цілісність зв’язків між таблицями.
6. **Оформити результати роботи**:
   * Надати пояснювальну записку, що містить опис предметного середовища, моделі бази даних, SQL-код та результати тестування.
   * Підготувати презентацію для захисту курсової роботи.

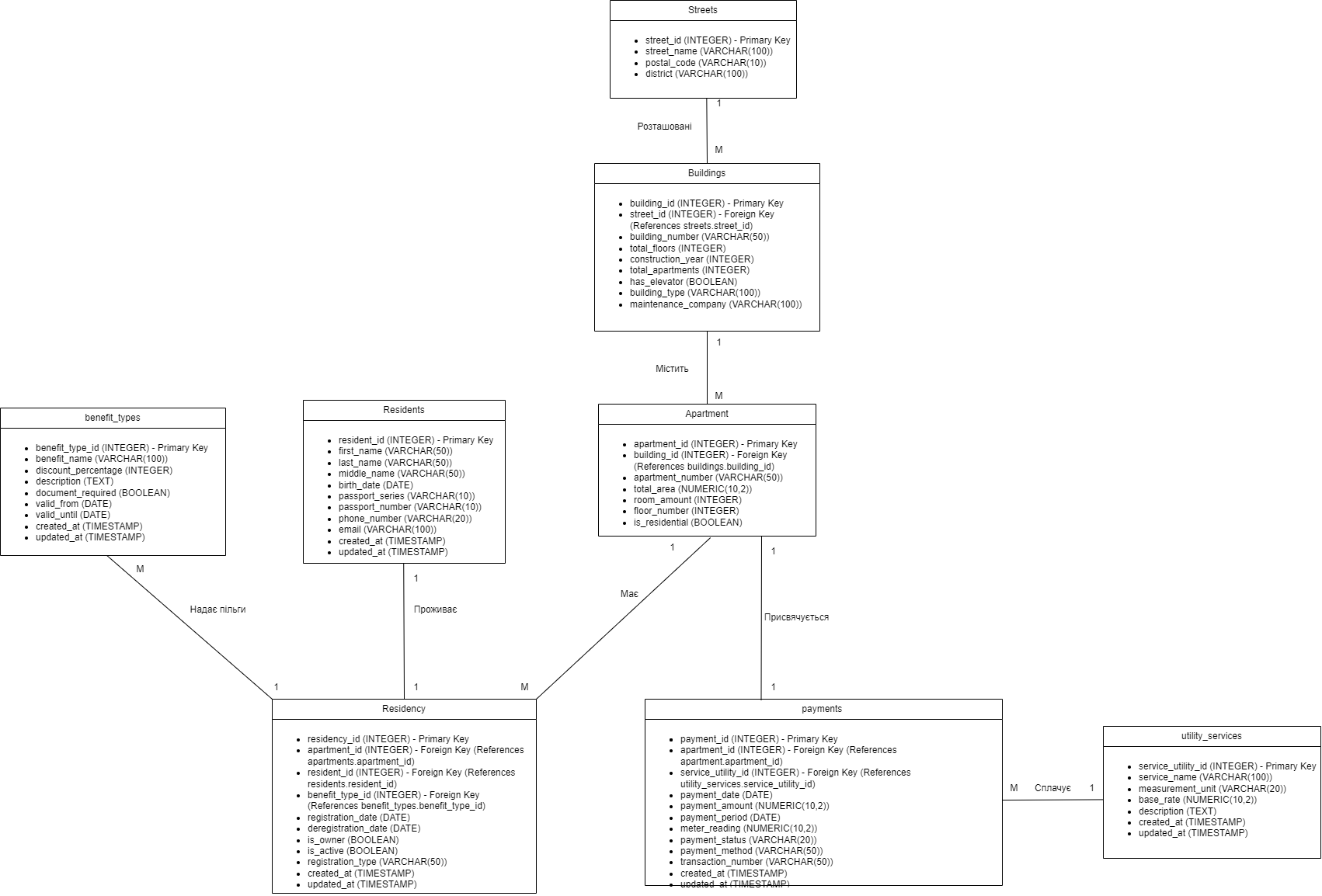
**Очікуваний результат**

Розроблена база даних повинна забезпечувати:

* Зручне введення, зберігання та оновлення інформації про житловий фонд.
* Швидкий пошук і отримання аналітичних даних.
* Узгодженість і цілісність даних у системі.
* Надійність і функціональність під час виконання запитів.

**ПРОЕКТУВАННЯ БАЗИ ДАНИХ**

**Побудована ER-модель:**



**Даталогічна модель бази даних:**

**streets**

* street\_id (INTEGER) - **Primary Key**
* street\_name (VARCHAR(100))
* postal\_code (VARCHAR(10))
* district (VARCHAR(100))

**buildings**

* building\_id (INTEGER) - **Primary Key**
* street\_id (INTEGER) - **Foreign Key** (References streets.street\_id)
* building\_number (VARCHAR(50))
* total\_floors (INTEGER)
* construction\_year (INTEGER)
* total\_apartments (INTEGER)
* has\_elevator (BOOLEAN)
* building\_type (VARCHAR(100))
* maintenance\_company (VARCHAR(100))

**apartments**

* apartment\_id (INTEGER) - **Primary Key**
* building\_id (INTEGER) - **Foreign Key** (References buildings.building\_id)
* apartment\_number (VARCHAR(50))
* total\_area (NUMERIC(10,2))
* room\_amount (INTEGER)
* floor\_number (INTEGER)
* is\_residential (BOOLEAN)

**residents**

* resident\_id (INTEGER) - **Primary Key**
* first\_name (VARCHAR(50))
* last\_name (VARCHAR(50))
* middle\_name (VARCHAR(50))
* birth\_date (DATE)
* passport\_series (VARCHAR(10))
* passport\_number (VARCHAR(10))
* phone\_number (VARCHAR(20))
* email (VARCHAR(100))
* created\_at (TIMESTAMP)
* updated\_at (TIMESTAMP)

**benefit\_types**

* benefit\_type\_id (INTEGER) - **Primary Key**
* benefit\_name (VARCHAR(100))
* discount\_percentage (INTEGER)
* description (TEXT)
* document\_required (BOOLEAN)
* valid\_from (DATE)
* valid\_until (DATE)
* created\_at (TIMESTAMP)
* updated\_at (TIMESTAMP)

**residency**

* residency\_id (INTEGER) - **Primary Key**
* apartment\_id (INTEGER) - **Foreign Key** (References apartments.apartment\_id)
* resident\_id (INTEGER) - **Foreign Key** (References residents.resident\_id)
* benefit\_type\_id (INTEGER) - **Foreign Key** (References benefit\_types.benefit\_type\_id)
* registration\_date (DATE)
* deregistration\_date (DATE)
* is\_owner (BOOLEAN)
* is\_active (BOOLEAN)
* registration\_type (VARCHAR(50))
* created\_at (TIMESTAMP)
* updated\_at (TIMESTAMP)

**utility\_services**

* service\_utility\_id (INTEGER) - **Primary Key**
* service\_name (VARCHAR(100))
* measurement\_unit (VARCHAR(20))
* base\_rate (NUMERIC(10,2))
* description (TEXT)
* created\_at (TIMESTAMP)
* updated\_at (TIMESTAMP)

**payments**

* payment\_id (INTEGER) - Primary Key
* apartment\_id (INTEGER) - Foreign Key (References apartment.apartment\_id)
* service\_utility\_id (INTEGER) - Foreign Key (References utility\_services.service\_utility\_id)
* payment\_date (DATE)
* payment\_amount (NUMERIC(10,2))
* payment\_period (DATE)
* meter\_reading (NUMERIC(10,2))
* payment\_status (VARCHAR(20))
* payment\_method (VARCHAR(50))
* transaction\_number (VARCHAR(50))
* created\_at (TIMESTAMP)
* updated\_at (TIMESTAMP)

**РЕАЛІЗАЦІЯ БАЗИ ДАНИХ**

У цьому розділі описано структуру та реалізацію бази даних для обліку житлового фонду. База даних розроблена з урахуванням усіх вимог щодо зберігання та обробки інформації про будинки, квартири та їх мешканців.

**Структура бази даних**

База даних складається з наступних основних таблиць:

**Таблиця "streets" (Вулиці)**

Зберігає інформацію про вулиці міста. Містить поля:

* street\_id: унікальний ідентифікатор вулиці
* street\_name: назва вулиці
* postal\_code: поштовий індекс
* district: район міста

**Таблиця "buildings" (Будинки)**

Містить дані про житлові будинки:

* building\_id: унікальний ідентифікатор будинку
* street\_id: зовнішній ключ для зв'язку з таблицею вулиць
* building\_number: номер будинку
* total\_floors: кількість поверхів
* construction\_year: рік побудови
* total\_apartments: загальна кількість квартир
* has\_elevator: наявність ліфту
* building\_type: тип будівлі
* maintenance\_company: компанія, що обслуговує будинок

**Таблиця "apartments" (Квартири)**

Зберігає інформацію про квартири:

* apartment\_id: унікальний ідентифікатор квартири
* building\_id: зовнішній ключ для зв'язку з таблицею будинків
* apartment\_number: номер квартири
* total\_area: загальна площа
* room\_amount: кількість кімнат
* floor\_number: поверх
* is\_residential: ознака житлового приміщення

**Таблиця "residents" (Мешканці)**

Містить персональні дані мешканців:

* resident\_id: унікальний ідентифікатор мешканця
* first\_name, last\_name, middle\_name: ПІБ мешканця
* birth\_date: дата народження
* passport\_series, passport\_number: паспортні дані
* phone\_number, email: контактна інформація

**Таблиця "benefit\_types" (Види пільг)**

Зберігає інформацію про доступні пільги:

* benefit\_type\_id: унікальний ідентифікатор пільги
* benefit\_name: назва пільги
* discount\_percentage: відсоток знижки
* description: опис пільги
* valid\_from, valid\_until: період дії пільги

**Таблиця "residency" (Проживання)**

Зв'язуюча таблиця між квартирами та мешканцями:

* residency\_id: унікальний ідентифікатор запису
* apartment\_id, resident\_id: зовнішні ключі
* benefit\_type\_id: тип пільги мешканця
* registration\_date: дата прописки
* deregistration\_date: дата виписки
* is\_owner: ознака власника
* is\_active: активність запису

**Таблиця "utility\_services" (Комунальні послуги)**

Містить інформацію про комунальні послуги:

* service\_utility\_id: унікальний ідентифікатор послуги
* service\_name: назва послуги
* measurement\_unit: одиниця виміру
* base\_rate: базовий тариф

**Таблиця "payments" (Платежі)**Містить інформацію про платежі за комунальні послуги:

* payment\_id: унікальний ідентифікатор платежу (Primary Key)
* apartment\_id: ідентифікатор квартири (Foreign Key, посилається на apartment.apartment\_id)
* service\_utility\_id: ідентифікатор комунальної послуги (Foreign Key, посилається на utility\_services.service\_utility\_id)
* payment\_date: дата здійснення платежу
* payment\_amount: сума платежу (формат NUMERIC(10,2))
* payment\_period: період, за який здійснюється оплата (місяць та рік)
* meter\_reading: показники лічильника (формат NUMERIC(10,2))
* payment\_status: статус платежу (наприклад, 'paid', 'pending', 'overdue')
* payment\_method: спосіб оплати (наприклад, 'Приват24', 'наложений платіж' тощо)
* transaction\_number: номер транзакції
* created\_at: дата та час створення запису (TIMESTAMP)
* updated\_at: дата та час останнього оновлення запису (TIMESTAMP)

**Обґрунтування вибору типів даних**

1. Для ідентифікаторів (ID) використано тип INTEGER, що забезпечує унікальність та ефективну індексацію.
2. Для текстових полів обрано VARCHAR з різними обмеженнями довжини:
   * 100 символів для назв вулиць, послуг тощо
   * 50 символів для імен, прізвищ
   * 10 символів для паспортних даних
3. Для числових значень з десятковими частинами (площа, тарифи) використано NUMERIC(10,2)
4. Для дат використано типи DATE та TIMESTAMP
5. Для логічних значень використано тип BOOLEAN

**Зв'язки між таблицями**

База даних використовує наступні зв'язки:

1. streets ← buildings (один-до-багатьох)
2. buildings ← apartments (один-до-багатьох)
3. apartments ↔ residents (багато-до-багатьох через residency)
4. benefit\_types ← residency (один-до-багатьох)

**Індексація та оптимізація**

Для оптимізації пошуку та вибірки даних створено наступні індекси:

1. Первинні ключі для кожної таблиці (PRIMARY KEY)
2. Зовнішні ключі для зв'язків між таблицями (FOREIGN KEY)
3. Індекси для часто використовуваних полів пошуку:
   * building\_number в таблиці buildings
   * last\_name в таблиці residents
   * registration\_date в таблиці residency

**Забезпечення цілісності даних**

1. Усі первинні ключі мають обмеження NOT NULL та AUTO\_INCREMENT
2. Зовнішні ключі мають обмеження ON DELETE RESTRICT
3. Встановлено CHECK-обмеження для:
   * Позитивних значень площі та кількості кімнат
   * Коректності дат реєстрації
   * Відсотку пільг у межах від 0 до 100

Така структура бази даних забезпечує ефективне зберігання та обробку даних про житловий фонд, дозволяє швидко отримувати необхідну інформацію та підтримує всі необхідні функції системи.

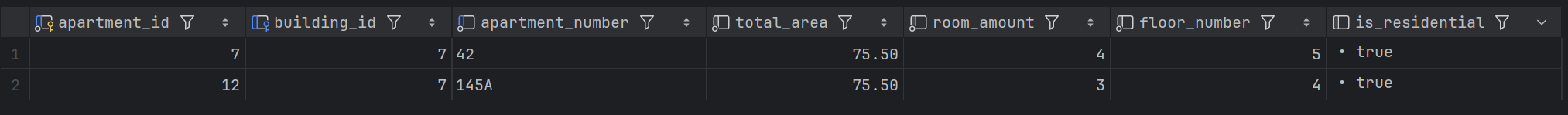
**РОБОТА З БАЗОЮ ДАНИХ**

У цьому розділі описано основні операції та взаємодію з базою даних.

**1. Робота з таблицею apartments**

**1.1 Додавання нової квартири**

INSERT INTO apartments (  
 building\_id,  
 apartment\_number,  
 total\_area,  
 room\_amount,  
 floor\_number,  
 is\_residential  
) VALUES (  
 1, -- ID будинку  
 '145A', -- Номер квартири  
 75.5, -- Загальна площа  
 3, -- Кількість кімнат  
 4, -- Поверх  
 true -- Житлове приміщення  
 );

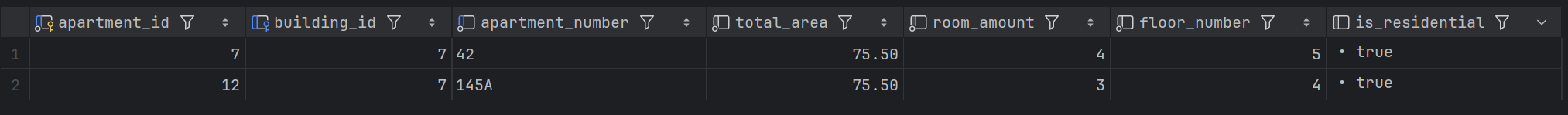
**Результат: :?  
**

**1.2 Отримання інформації про квартири**

Вибірка всіх квартир у будинку:

SELECT  
 a.apartment\_number,  
 a.total\_area,  
 a.room\_amount,  
 a.floor\_number,  
 b.building\_number,  
 s.street\_name  
FROM apartments a  
 JOIN buildings b ON a.building\_id = b.building\_id  
 JOIN streets s ON b.street\_id = s.street\_id  
WHERE b.building\_id = 1;

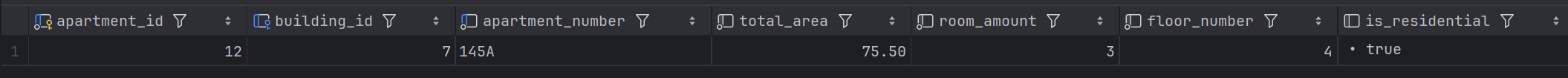
**Результат:**

****

**Пошук квартир за кількістю кімнат:**

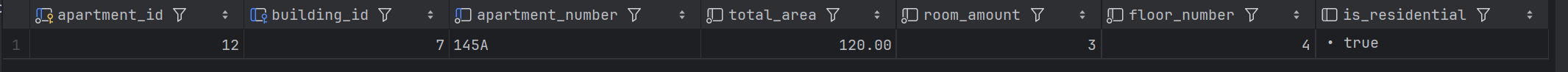
SELECT \* FROM apartments  
WHERE room\_amount = 3  
 AND is\_residential = true;

**Результат:**

****

**Пошук квартир за площею:**

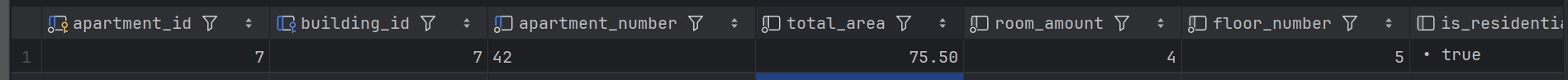
SELECT \* FROM apartments  
WHERE total\_area > 60  
ORDER BY total\_area DESC;

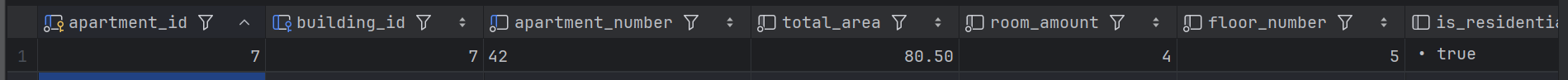
**Результат:**

**1.3 Оновлення даних про квартиру**

UPDATE apartments  
SET  
 total\_area = 80.5,  
 room\_amount = 4  
WHERE apartment\_id = 7;

**До:**

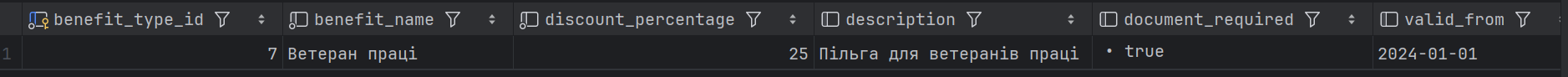


**Після:** ****

**2. РОБОТА З ТАБЛИЦЕЮ BENEFIT\_TYPES**

**2.1 Додавання нового типу пільг**

INSERT INTO benefit\_types (  
 benefit\_name,  
 discount\_percentage,  
 description,  
 document\_required,  
 valid\_from,  
 valid\_until  
) VALUES (  
 'Пільга для ветеранів',  
 50,  
 'Пільга для ветеранів війни та праці',  
 true,  
 '2024-01-01',  
 '2024-12-31'  
 );

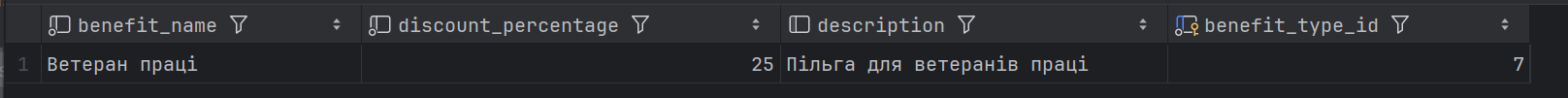


**2.2 Отримання інформації про пільги**

Перегляд всіх активних пільг:

SELECT  
 benefit\_name,  
 discount\_percentage,  
 description  
FROM benefit\_types  
WHERE CURRENT\_DATE BETWEEN valid\_from AND valid\_until;

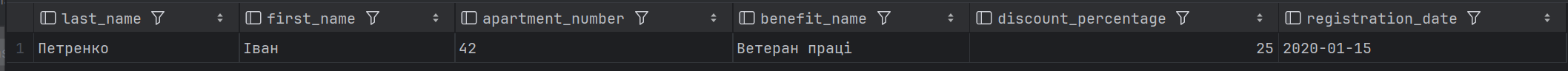
**Результат:**



**2.3 Комплексний запит по пільговикам**

SELECT  
 r.last\_name,  
 r.first\_name,  
 a.apartment\_number,  
 bt.benefit\_name,  
 bt.discount\_percentage,  
 res.registration\_date  
FROM benefit\_types bt  
 JOIN residency res ON bt.benefit\_type\_id = res.benefit\_type\_id  
 JOIN residents r ON res.resident\_id = r.resident\_id  
 JOIN apartments a ON res.apartment\_id = a.apartment\_id  
WHERE res.is\_active = true  
ORDER BY bt.discount\_percentage DESC;

**Результат:**

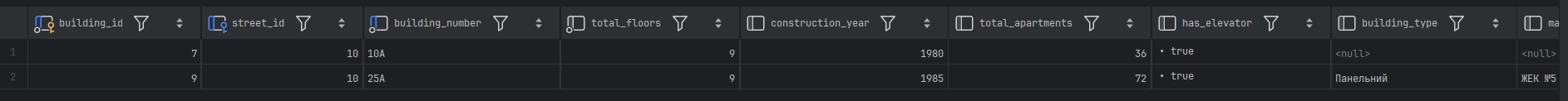


**3. РОБОТА З ТАБЛИЦЕЮ BUILDINGS**

**3.1 Додавання нового будинку**

INSERT INTO buildings (  
 street\_id,  
 building\_number,  
 total\_floors,  
 construction\_year,  
 total\_apartments,  
 has\_elevator,  
 building\_type,  
 maintenance\_company  
) VALUES (1, -- ID вулиці  
 '25A', -- Номер будинку  
 9, -- Кількість поверхів  
 1985, -- Рік побудови  
 72, -- Загальна кількість квартир  
 true, -- Наявність ліфту  
 'Панельний', -- Тип будівлі  
 'ЖЕК №5' -- Обслуговуюча компанія  
 );

**Результат:**

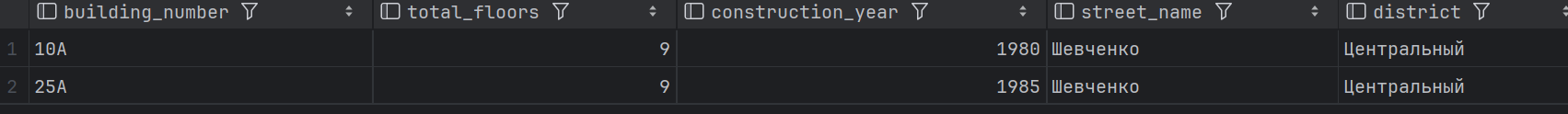


**3.2 Отримання інформації про будинки**

**Вибірка всіх будинків на вулиці:**

SELECT  
 b.building\_number,  
 b.total\_floors,  
 b.construction\_year,  
 s.street\_name,  
 s.district  
FROM buildings b  
 JOIN streets s ON b.street\_id = s.street\_id  
WHERE s.street\_name LIKE 'Шевченко';

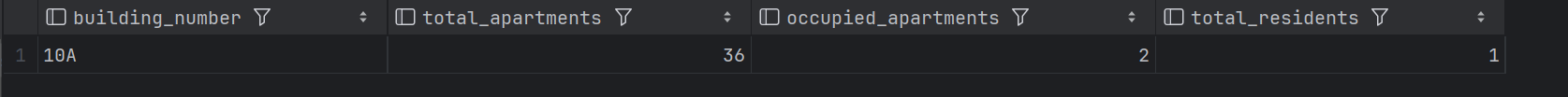
**Результат:**



**3.3 Статистика по будинку:**

SELECT  
 b.building\_number,  
 b.total\_apartments,  
 COUNT(a.apartment\_id) as occupied\_apartments,  
 COUNT(DISTINCT r.resident\_id) as total\_residents  
FROM buildings b  
 LEFT JOIN apartments a ON b.building\_id = a.building\_id  
 LEFT JOIN residency r ON a.apartment\_id = r.apartment\_id  
WHERE b.building\_id = 1  
GROUP BY b.building\_id, b.building\_number, b.total\_apartments;

**Результат:**

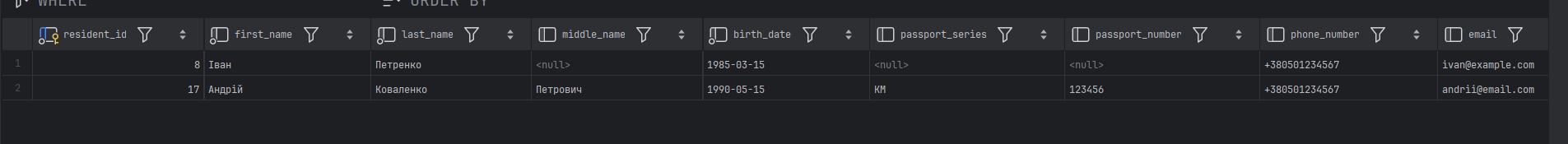


**4. РОБОТА З ТАБЛИЦЕЮ RESIDENTS**

**4.1 Додавання нового мешканця**

INSERT INTO residents (  
 first\_name,  
 last\_name,  
 middle\_name,  
 birth\_date,  
 passport\_series,  
 passport\_number,  
 phone\_number,  
 email,  
) VALUES (  
 'Андрій',  
 'Коваленко',  
 'Петрович',  
 '1990-05-15',  
 'КМ',  
 '123456',  
 '+380501234567',  
 'andrii@email.com',);

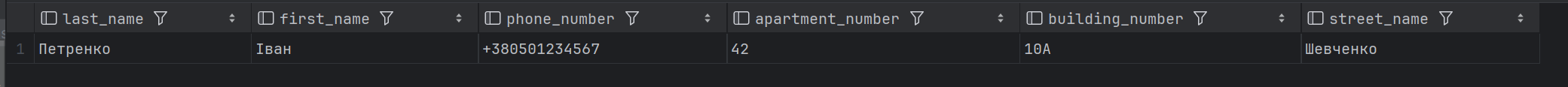
**Результат:**



**3.2 Пошук мешканців**

SELECT  
 r.last\_name,  
 r.first\_name,  
 r.phone\_number,  
 a.apartment\_number,  
 b.building\_number,  
 s.street\_name  
FROM residents r  
 JOIN residency res ON r.resident\_id = res.resident\_id  
 JOIN apartments a ON res.apartment\_id = a.apartment\_id  
 JOIN buildings b ON a.building\_id = b.building\_id  
 JOIN streets s ON b.street\_id = s.street\_id  
WHERE r.last\_name LIKE 'Петренко'  
 AND res.is\_active = true;

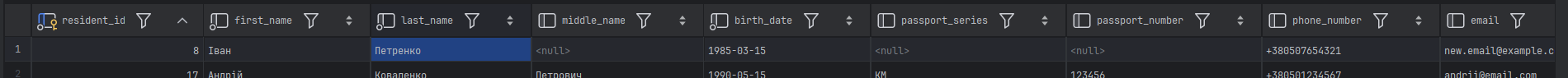
**Результат:**

****

**4.3 Оновлення даних мешканця**

UPDATE residents  
SET  
 phone\_number = '+380507654321',  
 email = 'new.email@example.com',  
WHERE resident\_id = 1;

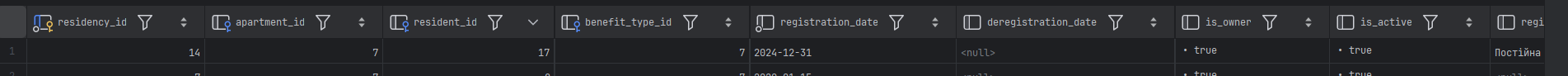
**Результат:**



**5. РОБОТА З ТАБЛИЦЕЮ RESIDENCY**

**5.1 Реєстрація мешканця в квартирі**

INSERT INTO residency (  
 apartment\_id,  
 resident\_id,  
 benefit\_type\_id,  
 registration\_date,  
 is\_owner,  
 is\_active,  
 registration\_type  
) VALUES (  
 7, -- ID квартири  
 17, -- ID мешканця  
 7, -- ID типу пільги  
 *CURRENT\_DATE*, -- Дата реєстрації  
 true, -- Є власником  
 true, -- Активна реєстрація  
 'Постійна' -- Тип реєстрації  
 );

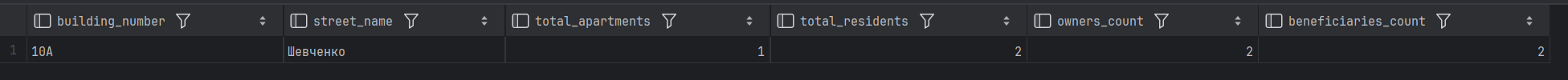
**Результат:**

**5.2 Складні запити для аналізу даних**

Аналіз заселеності будинку:

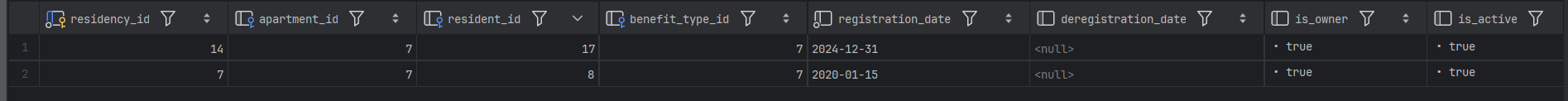
SELECT  
 b.building\_number,  
 s.street\_name,  
 COUNT(DISTINCT a.apartment\_id) as total\_apartments,  
 COUNT(DISTINCT res.resident\_id) as total\_residents,  
 COUNT(DISTINCT CASE WHEN res.is\_owner THEN res.resident\_id END) as owners\_count,  
 COUNT(DISTINCT CASE WHEN bt.benefit\_type\_id IS NOT NULL THEN res.resident\_id END) as beneficiaries\_count  
FROM buildings b  
 JOIN streets s ON b.street\_id = s.street\_id  
 LEFT JOIN apartments a ON b.building\_id = a.building\_id  
 LEFT JOIN residency res ON a.apartment\_id = res.apartment\_id  
 LEFT JOIN benefit\_types bt ON res.benefit\_type\_id = bt.benefit\_type\_id  
WHERE res.is\_active = true  
GROUP BY b.building\_id, b.building\_number, s.street\_name;

**Результат:**

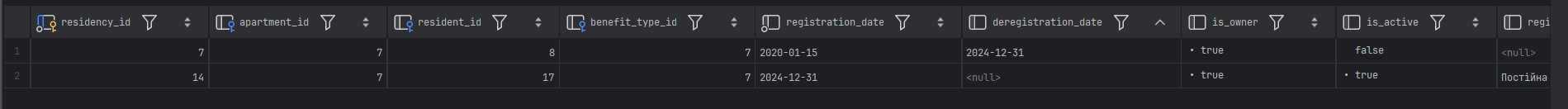


**5.3 Виписка мешканця**

UPDATE residency  
SET  
 is\_active = false,  
 deregistration\_date = *CURRENT\_DATE*WHERE resident\_id = 8  
 AND apartment\_id = 7  
 AND is\_active = true;

**До:**

**Після:**

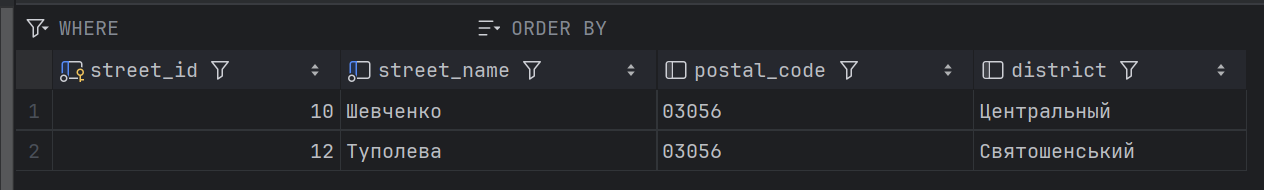
****

**6. РОБОТА З ТАБЛИЦЕЮ STREETS**

**6.1 Додавання нової вулиці**

INSERT INTO streets (  
 street\_name,  
 postal\_code,  
 district  
) VALUES (  
 'Туполева', -- Назва вулиці  
 '03056', -- Поштовий індекс  
 'Святошенський' -- Район міста  
 );

**Результат:**

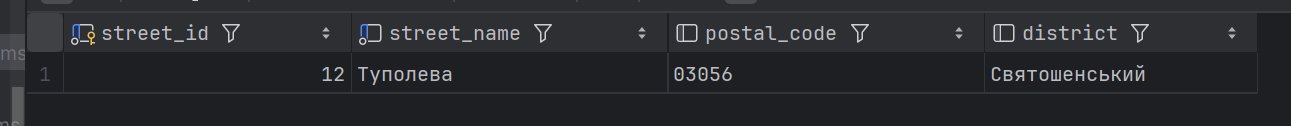


**6.2 Отримання інформації про вулиці**

Пошук всіх вулиць у районі

SELECT \*  
FROM streets  
WHERE district = 'Святошенський'  
ORDER BY street\_name;

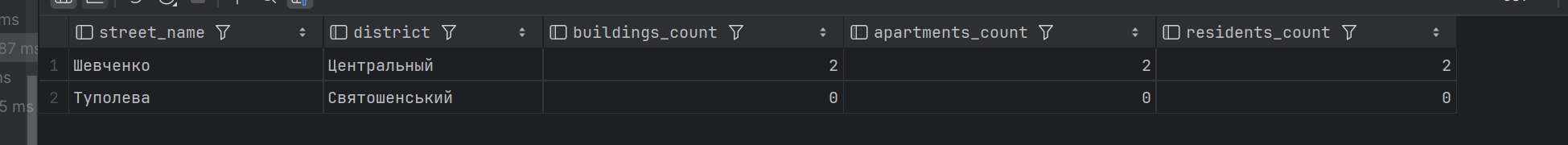
**Результат:**



**6.3 Статистика по вулицях:**

SELECT  
 s.street\_name,  
 s.district,  
 *COUNT*(DISTINCT b.building\_id) as buildings\_count,  
 *COUNT*(DISTINCT a.apartment\_id) as apartments\_count,  
 *COUNT*(DISTINCT res.resident\_id) as residents\_count  
FROM streets s  
 LEFT JOIN buildings b ON s.street\_id = b.street\_id  
 LEFT JOIN apartments a ON b.building\_id = a.building\_id  
 LEFT JOIN residency res ON a.apartment\_id = res.apartment\_id  
GROUP BY s.street\_id, s.street\_name, s.district;

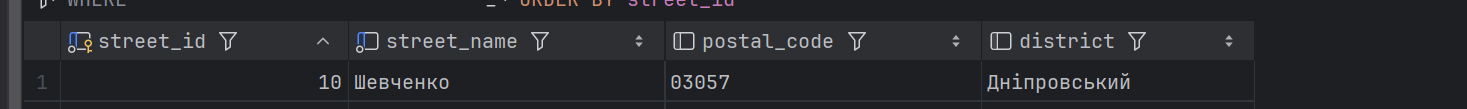
**Результат:**



**6.4 Оновлення даних вулиці**

UPDATE streets  
SET  
 postal\_code = '03057',  
 district = 'Дніпровський'  
WHERE street\_id = 10;

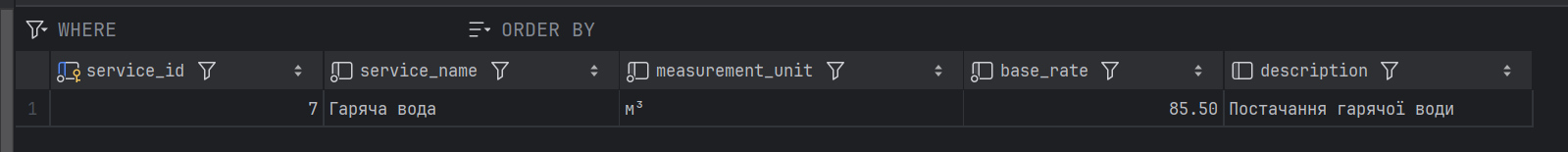
**Результат:**



**7. РОБОТА З ТАБЛИЦЕЮ UTILITY\_SERVICES**

**7.1 Додавання нової комунальної послуги**

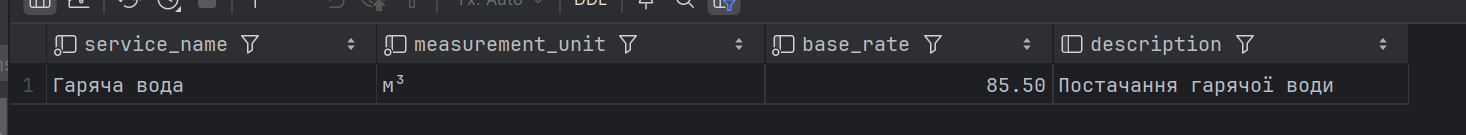
INSERT INTO utility\_services (service\_name,  
 measurement\_unit,  
 base\_rate,  
 description)  
VALUES ('Гаряча вода', -- Назва послуги  
 'м³', -- Одиниця виміру  
 85.50, -- Базовий тариф  
 'Постачання гарячої води');



**7.2 Отримання інформації про послуги**

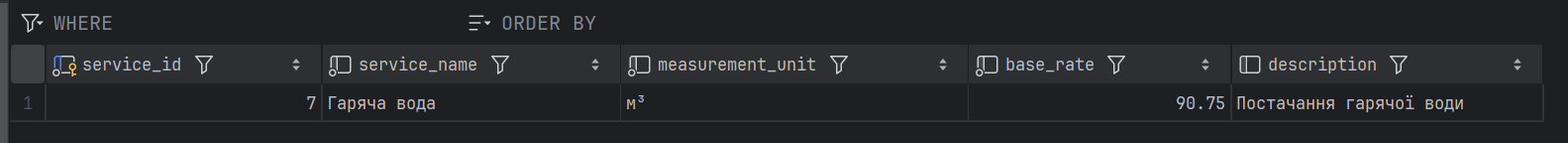
Перегляд всіх комунальних послуг:

SELECT  
 service\_name,  
 measurement\_unit,  
 base\_rate,  
 description  
FROM utility\_services  
ORDER BY service\_name;



7.3 Оновлення тарифів:

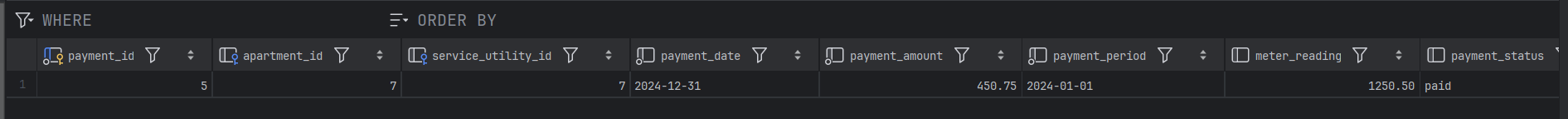
UPDATE utility\_services  
SET  
 base\_rate = 90.75,  
WHERE service\_id = 7;



**РОБОТА З ТАБЛИЦЕЮ PAYMENTS**

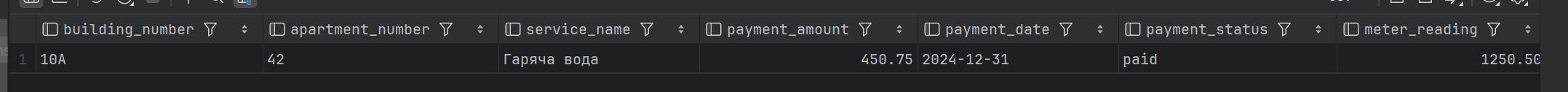
**8.1 Додавання нового платежу**

INSERT INTO payments (  
 apartment\_id,  
 service\_utility\_id,  
 payment\_date,  
 payment\_amount,  
 payment\_period,  
 meter\_reading,  
 payment\_status,  
 payment\_method,  
 transaction\_number  
) VALUES (  
 7, -- ID квартири  
 7, -- ID комунальної послуги  
 *CURRENT\_DATE*, -- Дата платежу  
 450.75, -- Сума платежу  
 '2024-01-01', -- Період оплати  
 1250.5, -- Показники лічильника  
 'paid', -- Статус платежу  
 'Приват24', -- Метод оплати  
 'TRX123456' -- Номер транзакції  
 );

****

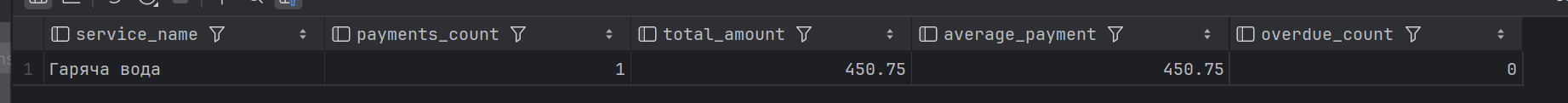
**8.2 Отримання інформації про платежі**

SELECT  
 b.building\_number,  
 a.apartment\_number,  
 us.service\_name,  
 p.payment\_amount,  
 p.payment\_date,  
 p.payment\_status,  
 p.meter\_reading,  
 p.payment\_method  
FROM payments p  
 JOIN apartments a ON p.apartment\_id = a.apartment\_id  
 JOIN buildings b ON a.building\_id = b.building\_id  
 JOIN utility\_services us ON p.service\_utility\_id = us.service\_id  
WHERE a.apartment\_id = 7  
ORDER BY p.payment\_date DESC;



**8.3 Аналіз платежів**

SELECT  
 us.service\_name,  
 *COUNT*(p.payment\_id) as payments\_count,  
 *SUM*(p.payment\_amount) as total\_amount,  
 *AVG*(p.payment\_amount) as average\_payment,  
 *COUNT*(CASE WHEN p.payment\_status = 'overdue' THEN 1 END) as overdue\_count  
FROM utility\_services us  
 LEFT JOIN payments p ON us.service\_id = p.service\_utility\_id  
GROUP BY us.service\_id, us.service\_name;

****