НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

_____ (повна назва кафедри, циклової комісії)

# КУРСОВА РОБОТА

з _____Бази данних_____

(назва дисципліни)

на тему:_____База даних обліку житлового фонду_____

_____

Студента Ткаченка Костянтина Олександровича, другого курсу, групи ІП-з31.

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: _____

Національна оцінка _____

Члени комісії

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2025р

# 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**

# ВСТУП

Розробка бази даних для обліку житлового фонду спрямована на вирішення наступних завдань:

- Централізоване зберігання та управління інформацією про будинки та квартири
- Облік мешканців та їх реєстрації
- Контроль за наданням пільг
- Моніторинг комунальних платежів
- Формування аналітичних звітів

Актуальність розробки такої системи обумовлена необхідністю:

- Підвищення ефективності управління житловим фондом
- Спрощення процесу обліку мешканців
- Автоматизації розрахунків комунальних платежів
- Забезпечення прозорості надання пільг
- Оптимізації роботи комунальних служб

Метою даної курсової роботи є розробка бази даних, яка забезпечить:

1. Зберігання та управління інформацією про:
    - Будинки та їх характеристики
    - Квартири та їх параметри
    - Мешканців та їх реєстрацію
    - Пільги та комунальні платежі
2. Реалізацію основних функцій:
    - Введення та редагування даних
    - Пошук інформації за різними критеріями
    - Формування звітів та статистики
    - Контроль за платежами та заборгованістю

# 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.

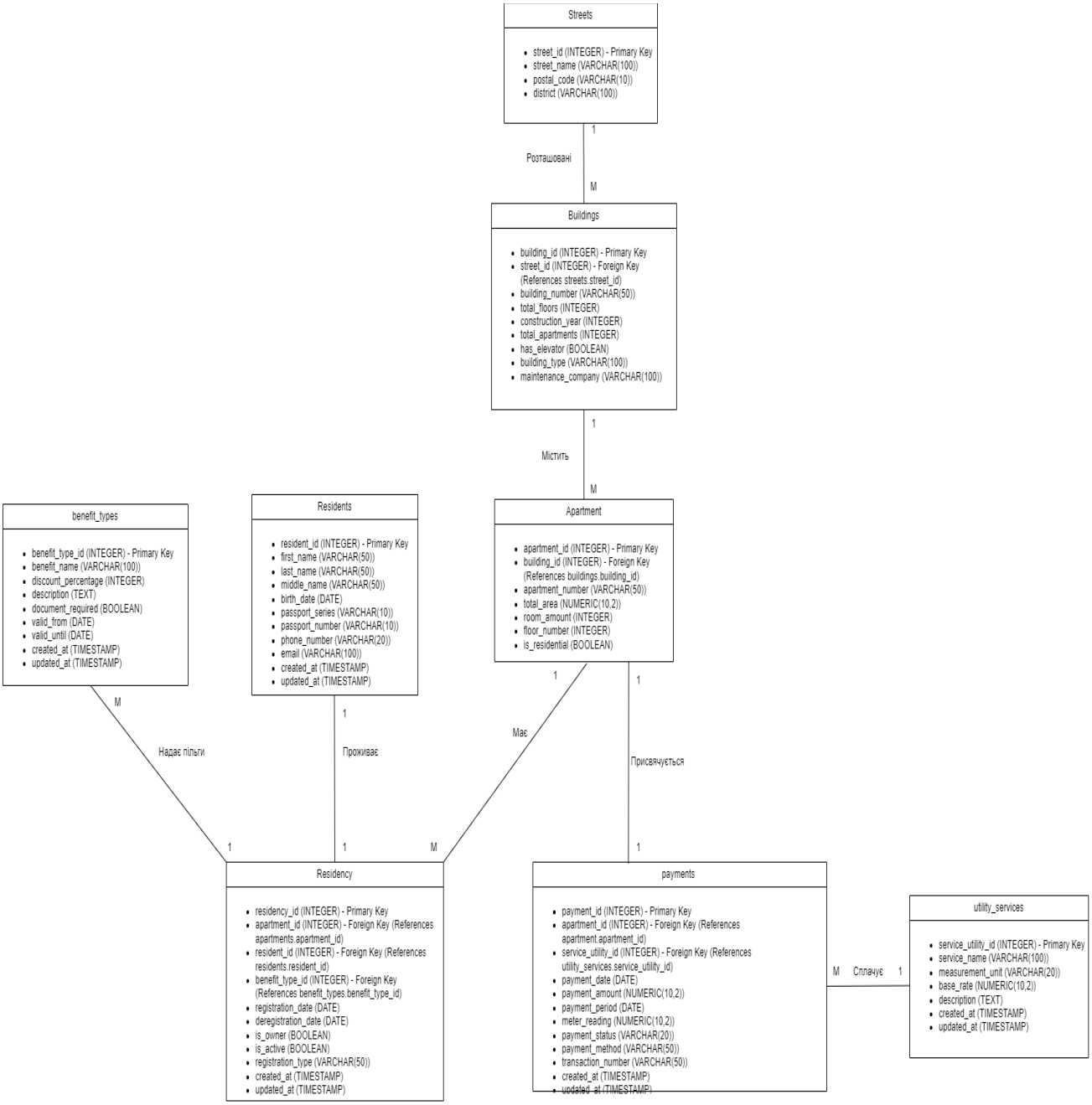
**Очікуваний результат**

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.

# Тригери:

### Перевірка площі на одного мешканця

```
CREATE OR REPLACE FUNCTION check_area_per_resident()
  RETURNS TRIGGER AS $$
DECLARE
  total_residents INTEGER;
  apartment_area NUMERIC;
```

```
    area_per_resident NUMERIC;
BEGIN
    IF TG_OP = 'UPDATE' THEN
        IF (OLD.apartment_id = NEW.apartment_id AND
            OLD.resident_id = NEW.resident_id AND
            OLD.is_active = NEW.is_active AND
            OLD.is_owner = NEW.is_owner) THEN
            RETURN NEW;
        END IF;
    END IF;

    -- Основна логіка перевірки площі
    SELECT total_area INTO apartment_area
    FROM apartments
    WHERE apartment_id = NEW.apartment_id;

    SELECT COUNT(*) INTO total_residents
    FROM residency
    WHERE apartment_id = NEW.apartment_id
      AND is_active = true;

    area_per_resident := apartment_area / (total_residents + 1);

    IF area_per_resident < 6 THEN
        RAISE EXCEPTION 'Недостатня площа на одного мешканця (%.2f кв.м). Мінімум - 6
кв.м', area_per_resident;
    END IF;

    RETURN NEW;
END;
CREATE TRIGGER check_area_before_registration
```

```
   BEFORE INSERT OR UPDATE ON residency
   FOR EACH ROW
EXECUTE FUNCTION check_area_per_resident();
```

## Розрахунок оплати з урахуванням пільг

```
CREATE  FUNCTION calculate_payment_with_benefits()
   RETURNS TRIGGER AS $$
DECLARE
   benefit_percent NUMERIC;
BEGIN
   SELECT bt.discount_percentage INTO benefit_percent
   FROM residency r
         JOIN benefit_types bt ON r.benefit_type_id = bt.benefit_type_id
   WHERE r.apartment_id = NEW.apartment_id
     AND r.is_active = true
     AND r.is_owner = true
   LIMIT 1;

   IF benefit_percent IS NOT NULL THEN
      NEW.payment_amount := NEW.payment_amount * (1 - benefit_percent/100);
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER apply_benefits_before_payment
   BEFORE INSERT OR UPDATE ON payments
   FOR EACH ROW
EXECUTE FUNCTION calculate_payment_with_benefits();
```

### Перевірка коректності поверху квартири

```
CREATE OR REPLACE FUNCTION verify_apartment_floor()
RETURNS TRIGGER AS $$
DECLARE
   max_floor INTEGER;
BEGIN

   SELECT total_floors INTO max_floor
   FROM buildings
   WHERE building_id = NEW.building_id;

   -- Перевіряємо коректність поверху
   IF NEW.floor_number > max_floor OR NEW.floor_number < 1 THEN
      RAISE EXCEPTION 'Некоректний номер поверху %. Допустимий діапазон: 1-%.',
         NEW.floor_number, max_floor;
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER check_floor_before_insert
   BEFORE INSERT OR UPDATE ON apartments
   FOR EACH ROW
EXECUTE FUNCTION verify_apartment_floor();
```

### Автоматичне оновлення дати зміни запису

```
CREATE OR REPLACE FUNCTION update_timestamp()
   RETURNS TRIGGER AS $$
BEGIN
   NEW.updated_at = CURRENT_TIMESTAMP;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_timestamp_trigger
    BEFORE UPDATE ON payments
    FOR EACH ROW
EXECUTE FUNCTION update_timestamp();
```

**Тригер для перевірки власника квартири**

**Підставлення (Views)**

**Загальна інформація про квартири з адресами**

```
CREATE VIEW apartment_details AS
SELECT
    a.apartment_id,
    s.street_name,
    b.building_number,
    a.apartment_number,
    a.total_area,
    a.room_amount,
    a.floor_number,
    b.total_floors,
    a.is_residential
FROM apartments a
        JOIN buildings b ON a.building_id = b.building_id
        JOIN streets s ON b.street_id = s.street_id;
```

**Інформація про мешканців та їх квартири**

```
CREATE VIEW resident_info AS
SELECT
    r.resident_id,
    r.last_name,
    r.first_name,
    r.middle_name,
    s.street_name,
    b.building_number,
    a.apartment_number,
    rs.registration_date,
    rs.is_owner,
    bt.benefit_name,
    bt.discount_percentage
FROM residents r
        JOIN residency rs ON r.resident_id = rs.resident_id
        JOIN apartments a ON rs.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
        LEFT JOIN benefit_types bt ON rs.benefit_type_id = bt.benefit_type_id
WHERE rs.is_active = true;
```

**Статистика платежів по квартирах**

```
CREATE VIEW payment_statistics AS
SELECT
    a.apartment_id,
    s.street_name,
```

```
    b.building_number,
    a.apartment_number,
    us.service_name,
    COUNT(p.payment_id) as total_payments,
    SUM(p.payment_amount) as total_amount,
    AVG(p.payment_amount) as average_payment,
    MAX(p.payment_date) as last_payment_date
FROM apartments a
        JOIN buildings b ON a.building_id = b.building_id
        JOIN streets s ON b.street_id = s.street_id
        JOIN payments p ON a.apartment_id = p.apartment_id
        JOIN utility_services us ON p.service_utility_id = us.service _id
GROUP BY a.apartment_id, s.street_name, b.building_number,
a.apartment_number, us.service_name;
```

**Боржники (прострочені платежі)**

```
CREATE VIEW overdue_payments AS
SELECT
    r.last_name,
    r.first_name,
    s.street_name,
    b.building_number,
    a.apartment_number,
    us.service_name,
    p.payment_amount,
    p.payment_period,
    p.payment_date,
```

```
    r.phone_number
FROM payments p
        JOIN apartments a ON p.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
        JOIN utility_services us ON p.service_utility_id = us.service _id
        JOIN residency rs ON a.apartment_id = rs.apartment_id
        JOIN residents r ON rs.resident_id = r.resident_id
WHERE p.payment_status = 'overdue'
  AND rs.is_owner = true
  AND rs.is_active = true;
```

**Статистика по будинках**

```
CREATE VIEW building_statistics AS
SELECT
    b.building_id,
    s.street_name,
    b.building_number,
    COUNT(DISTINCT a.apartment_id) as total_apartments,
    COUNT(DISTINCT rs.resident_id) as total_residents,
    SUM(a.total_area) as total_area,
    AVG(a.total_area) as average_apartment_area,
    COUNT(DISTINCT CASE WHEN rs.benefit_type_id IS NOT NULL THEN
rs.resident_id END) as residents_with_benefits
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 rs ON a.apartment_id = rs.apartment_id AND
rs.is_active = true
GROUP BY b.building_id, s.street_name, b.building_number;
```

**Загальна площа на мешканця**

```
CREATE VIEW area_per_resident AS
SELECT
   a.apartment_id,
   s.street_name,
   b.building_number,
   a.apartment_number,
   a.total_area,
   COUNT(rs.resident_id) as residents_count,
   ROUND(a.total_area / NULLIF(COUNT(rs.resident_id), 0), 2) as
area_per_resident
FROM apartments a
        JOIN buildings b ON a.building_id = b.building_id
        JOIN streets s ON b.street_id = s.street_id
        LEFT JOIN residency rs ON a.apartment_id = rs.apartment_id AND
rs.is_active = true
GROUP BY a.apartment_id, s.street_name, b.building_number,
a.apartment_number, a.total_area;
```

# РОБОТА З БАЗОЮ ДАНИХ

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

## 1.РОБОТА З ТАБЛИЦЕЮ APARTMENTS

1.1 Отримати список всіх квартир з їх основними характеристиками:

1.2 Знайти всі квартири з площею більше 100 м² та більше 2 кімнат:

1.3Отримати статистику квартир за поверхами

1.4 Знайти квартири з найбільшою кількістю мешканців

## 2 РОБОТА З ТАБЛИЦЕЮ BENEFIT_TYPES

2.1 Отримати список всіх активних пільг з сортуванням за розміром знижки

2.2 Знайти пільги з максимальною знижкою

2.3 Знайти пільги, термін дії яких закінчується протягом найближчих 3 місяців

2.4 Отримати список пільг із зазначенням їх актуального статусу

## 3 РОБОТА З ТАБЛИЦЕЮ BUILDINGS

3.1Отримати загальну інформацію про всі будинки

3.2С татистика будинків за типами

3.3Аналіз будинків за наявністю ліфтів

## 4 РОБОТА З ТАБЛИЦЕЮ RESIDENTS

4.1Отримати повний список мешканців з основною інформацією

4.2Знайти мешканців за віковими групами

4.3Отримати список мешканців з їх адресами проживання

4.4Знайти мешканців з пільгами:

## 5 РОБОТА З ТАБЛИЦЕЮ RESIDENCY

5.1Аналіз історії реєстрацій та виписок:

5.2Знайти квартири без активних мешканців:

## 6 РОБОТА З ТАБЛИЦЕЮ UTILITY_SERVICES

6.1 Отримати повний список комунальних послуг з тарифами

6.2 Аналіз платежів за кожною послугою

6.3Аналіз послуг за методами оплати:

## 7 РОБОТА З ТАБЛИЦЕЮ STREETS

7.1 Отримати базову інформацію про всі вулиці

7.2 Отримати базову інформацію про всі вулиці

## 8 РОБОТА З ТАБЛИЦЕЮ PAYMENTS

8.1 Загальна статистика платежів

8.2 Аналіз платежів за періодами

# ВИСНОВОК

У ході виконання курсової роботи було розроблено базу даних для обліку житлового фонду, яка забезпечує ефективне управління та облік інформації про будинки, квартири, мешканців та комунальні платежі.

Основні результати роботи:

1. Спроектовано структуру бази даних, що включає 8 взаємопов'язаних таблиць:
   - streets (вулиці)
   - buildings (будинки)
   - apartments (квартири)
   - residents (мешканці)
   - benefit_types (типи пільг)
   - residency (проживання)
   - utility_services (комунальні послуги)
   - payments (платежі)
2. Реалізовано всі необхідні функціональні вимоги системи:
   - Введення та зберігання даних про квартири та їх характеристики
   - Облік мешканців та їх реєстрації
   - Управління пільгами та знижками
   - Облік комунальних платежів
   - Розрахунок площі на одного мешканця
   - Пошук за різними критеріями
3. Забезпечено цілісність даних через:
   - Встановлення первинних та зовнішніх ключів
   - Додавання необхідних обмежень та перевірок
   - Створення індексів для оптимізації пошуку
4. Розроблено набір SQL-запитів для:

- Додавання, оновлення та видалення даних
- Пошуку інформації за різними параметрами
- Формування статистичних звітів
- Аналізу даних про платежі та заборгованості

5. Створено представлення (Views) для спрощення доступу до часто використовуваної інформації та статистики.

Розроблена база даних дозволяє ефективно:

- Вести облік житлового фонду
- Керувати інформацією про мешканців
- Відстежувати комунальні платежі
- Формувати різноманітні звіти
- Аналізувати дані для прийняття управлінських рішень

База даних має потенціал для подальшого розширення функціональності, зокрема можливе додавання:

- Модуля для обліку ремонтних робіт
- Системи автоматичного нарахування платежів
- Інтеграції з платіжними системами
- Розширеної аналітики та звітності

Таким чином, розроблена система повністю відповідає поставленим вимогам та може бути використана для ефективного управління житловим фондом.

# ДОДАТКИ

**SQL-скрипти створення та роботи з таблицями:**

**1.1 Отримати список всіх квартир з їх основними характеристиками:**

```
SELECT
    apartment_number as номер_квартири,
    total_area as загальна_площа,
    room_amount as кількість_кімнат,
    floor_number as поверх,
    CASE
        WHEN is_residential = true THEN 'Житлова'
        ELSE 'Нежитлова'
        END as тип_приміщення
FROM apartments
ORDER BY apartment_number;
```

1.2 **Знайти всі квартири з площею більше 100 м² та більше 2 кімнат:**

```
SELECT
    a.apartment_number,
    a.total_area,
    a.room_amount,
    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 a.total_area > 100 AND a.room_amount > 2
ORDER BY a.total_area DESC;
```

**1.3 Отримати статистику квартир за поверхами:**

```
SELECT
   floor_number as  floor,
   COUNT(*) as room_amount,
   ROUND(AVG(total_area), 2) as avg_square,
   ROUND(AVG(room_amount), 1) as avg_room_amount
FROM apartments
GROUP BY floor_number
ORDER BY floor_number;
```

**1.4 Знайти квартири з найбільшою кількістю мешканців:**

```
SELECT
   a.apartment_number as номер_квартири,
   a.total_area as площа,
   a.room_amount as кількість_кімнат,
   COUNT(r.resident_id) as кількість_мешканців
FROM apartments a
    JOIN residency r ON a.apartment_id = r.apartment_id
WHERE r.is_active = true
GROUP BY a.apartment_id, a.apartment_number, a.total_area, a.room_amount
ORDER BY кількість_мешканців DESC;
```

**2.1 Отримати список всіх активних пільг з сортуванням за розміром знижки:**

```
SELECT
    benefit_name,
    discount_percentage,
    description,
    valid_from,
    valid_until
FROM benefit_types
WHERE CURRENT_DATE BETWEEN valid_from AND valid_until
ORDER BY discount_percentage DESC;
```

**2.2 Знайти пільги з максимальною знижкою:**

```
SELECT
    benefit_name,
    discount_percentage,
    description
FROM benefit_types
WHERE discount_percentage = (
    SELECT MAX(discount_percentage)
    FROM benefit_types
);
```

**2.3 Знайти пільги, термін дії яких закінчується протягом найближчих 3** місяців:

```
SELECT
   CASE
      WHEN discount_percentage <= 25 THEN '0-25%'
      WHEN discount_percentage <= 50 THEN '26-50%'
      WHEN discount_percentage <= 75 THEN '51-75%'
      ELSE '76-100%'
      END AS discount_range,
   benefit_name,
   COUNT(*) AS benefits_count
FROM benefit_types
GROUP BY
   discount_range, benefit_name,
   CASE
      WHEN discount_percentage <= 25 THEN '0-25%'
      WHEN discount_percentage <= 50 THEN '26-50%'
      WHEN discount_percentage <= 75 THEN '51-75%'
      ELSE '76-100%'
      END
ORDER BY discount_range, benefit_name;
```

**2.4 Отримати список пільг із зазначенням їх актуального статусу:**

```
SELECT
   benefit_name,
   discount_percentage,
   valid_from,
   valid_until,
   CASE
```

```
        WHEN CURRENT_DATE < valid_from THEN 'Майбутня пільга'
        WHEN CURRENT_DATE > valid_until THEN 'Закінчена пільга'
        ELSE 'Активна пільга'
        END as benefit_status
FROM benefit_types
ORDER BY
    CASE
        WHEN CURRENT_DATE BETWEEN valid_from AND valid_until THEN 1
        WHEN CURRENT_DATE < valid_from THEN 2
        ELSE 3
        END,
    valid_from;
```

**3.1 Отримати загальну інформацію про всі будинки:**

```
SELECT
    b.building_number,
    s.street_name,
    b.total_floors,
    b.total_apartments,
    b.construction_year,
    CASE
        WHEN b.has_elevator THEN 'Є'
        ELSE 'Немає'
        END as elevator
FROM buildings b
```

```
        JOIN streets s ON b.street_id = s.street_id
ORDER BY s.street_name, b.building_number;
```

**3.2 Статистика будинків за типами:**

```
SELECT
   building_number,
   building_type,
   COUNT(*) as buildings_count,
   ROUND(AVG(total_floors)) as avg_floors,
   ROUND(AVG(total_apartments)) as avg_apartments,
   MIN(construction_year) as oldest_year,
   MAX(construction_year) as newest_year
FROM buildings
GROUP BY building_type, building_number
ORDER BY buildings_count DESC;
```

**3.3 Аналіз будинків за наявністю ліфтів:**

```
SELECT
   CASE
      WHEN total_floors <= 5 THEN '1-5 поверхів'
      WHEN total_floors <= 9 THEN '6-9 поверхів'
      ELSE '10+ поверхів'
      END as building_height,
   COUNT(*) as total_buildings,
   COUNT(CASE WHEN has_elevator THEN 1 END) as with_elevator,
   COUNT(CASE WHEN NOT has_elevator THEN 1 END) as without_elevator,
```

```
    ROUND(COUNT(CASE WHEN has_elevator THEN 1 END)::numeric /
COUNT(*) * 100, 2) as elevator_percentage
FROM buildings
GROUP BY
    CASE
        WHEN total_floors <= 5 THEN '1-5 поверхів'
        WHEN total_floors <= 9 THEN '6-9 поверхів'
        ELSE '10+ поверхів'
        END
ORDER BY building_height;
```

**4.1 Отримати повний список мешканців з основною інформацією:**

```
SELECT
    resident_id,
    last_name,
    first_name,
    middle_name,
    birth_date,
    phone_number,
    email
FROM residents
ORDER BY last_name, first_name;
```

**4.2 Знайти мешканців за віковими групами:**

```
SELECT
    CASE
```

```
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date)) <
18 THEN 'До 18 років'
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date)) <
35 THEN '18-35 років'
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date)) <
60 THEN '35-60 років'
    ELSE 'Старше 60 років'
    END as age_group,
  COUNT(*) as residents_count,
  ROUND(AVG(EXTRACT(YEAR FROM AGE(CURRENT_DATE,
birth_date))), 1) as avg_age
FROM residents
GROUP BY
  CASE
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date)) <
18 THEN 'До 18 років'
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date)) <
35 THEN '18-35 років'
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date)) <
60 THEN '35-60 років'
    ELSE 'Старше 60 років'
    END
ORDER BY age_group;
```

**4.3 Отримати список мешканців з їх адресами проживання:**

```
SELECT
```

```
    r.last_name,
    r.first_name,
    s.street_name,
    b.building_number,
    a.apartment_number,
    CASE
        WHEN res.is_owner THEN 'Власник'
        ELSE 'Мешканець'
        END as status
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 res.is_active = true
ORDER BY s.street_name, b.building_number, a.apartment_number;
```

**4.4 Знайти мешканців з пільгами:**

```
SELECT
    r.last_name,
    r.first_name,
    bt.benefit_name,
    bt.discount_percentage,
    res.registration_date
FROM residents r
        JOIN residency res ON r.resident_id = res.resident_id
        JOIN benefit_types bt ON res.benefit_type_id = bt.benefit_type_id
```

```
WHERE res.is_active = true
ORDER BY bt.discount_percentage DESC;
```

**5.1 Аналіз історії реєстрацій та виписок:**

```
SELECT
   EXTRACT(YEAR FROM registration_date) as year,
   COUNT(*) as total_registrations,
   COUNT(CASE WHEN deregistration_date IS NOT NULL THEN 1 END) as
deregistrations,
   COUNT(CASE WHEN is_owner THEN 1 END) as owner_registrations
FROM residency
GROUP BY EXTRACT(YEAR FROM registration_date)
ORDER BY year;
```

**5.2 Знайти квартири без активних мешканців:**

```
SELECT
   s.street_name,
   b.building_number,
   a.apartment_number,
   MAX(res.deregistration_date) as last_resident_date
FROM apartments a
      LEFT JOIN residency res ON a.apartment_id = res.apartment_id
      JOIN buildings b ON a.building_id = b.building_id
      JOIN streets s ON b.street_id = s.street_id
```

```
GROUP BY a.apartment_id, s.street_name, b.building_number,
a.apartment_number
HAVING COUNT(CASE WHEN res.is_active THEN 1 END) = 0
ORDER BY last_resident_date DESC NULLS LAST;
```

**6.1 Отримати повний список комунальних послуг з тарифами:**

```
SELECT
    service_id,
    service_name,
    measurement_unit,
    base_rate,
    ROUND(base_rate * 1.2, 2) as rate_with_vat
FROM utility_services
ORDER BY base_rate DESC;
```

**6.2 Аналіз платежів за кожною послугою:**

```
SELECT
    us.service_name,
    COUNT(p.payment_id) as total_payments,
    SUM(p.payment_amount) as total_amount,
    ROUND(AVG(p.payment_amount), 2) as avg_payment,
    COUNT(CASE WHEN p.payment_status = 'overdue' THEN 1 END) as
overdue_payments
FROM utility_services us
        LEFT JOIN payments p ON us.service_id = p.service_utility_id
GROUP BY us.service_id, us.service_name
```

```
ORDER BY total_amount DESC;\=
```

**6.2 Аналіз послуг за методами оплати:**

```
SELECT
    us.service_name,
    p.payment_method,
    COUNT(*) as payments_count,
    ROUND(AVG(p.payment_amount), 2) as avg_payment_amount,
    SUM(p.payment_amount) as total_amount
FROM utility_services us
        JOIN payments p ON us.service_id = p.service_utility_id
GROUP BY us.service_name, p.payment_method
ORDER BY us.service_name, payments_count DESC;
```

**7.1 Отримати базову інформацію про всі вулиці**

```
SELECT
    s.street_id,
    s.street_name,
    s.postal_code,
    s.district,
    COUNT(DISTINCT b.building_id) AS buildings_count
FROM streets s
        LEFT JOIN buildings b ON s.street_id = b.street_id
GROUP BY s.street_id, s.street_name, s.postal_code, s.district
```

```
ORDER BY s.district, s.street_name;
```

**7.2 Аналіз житлового фонду по районах:**

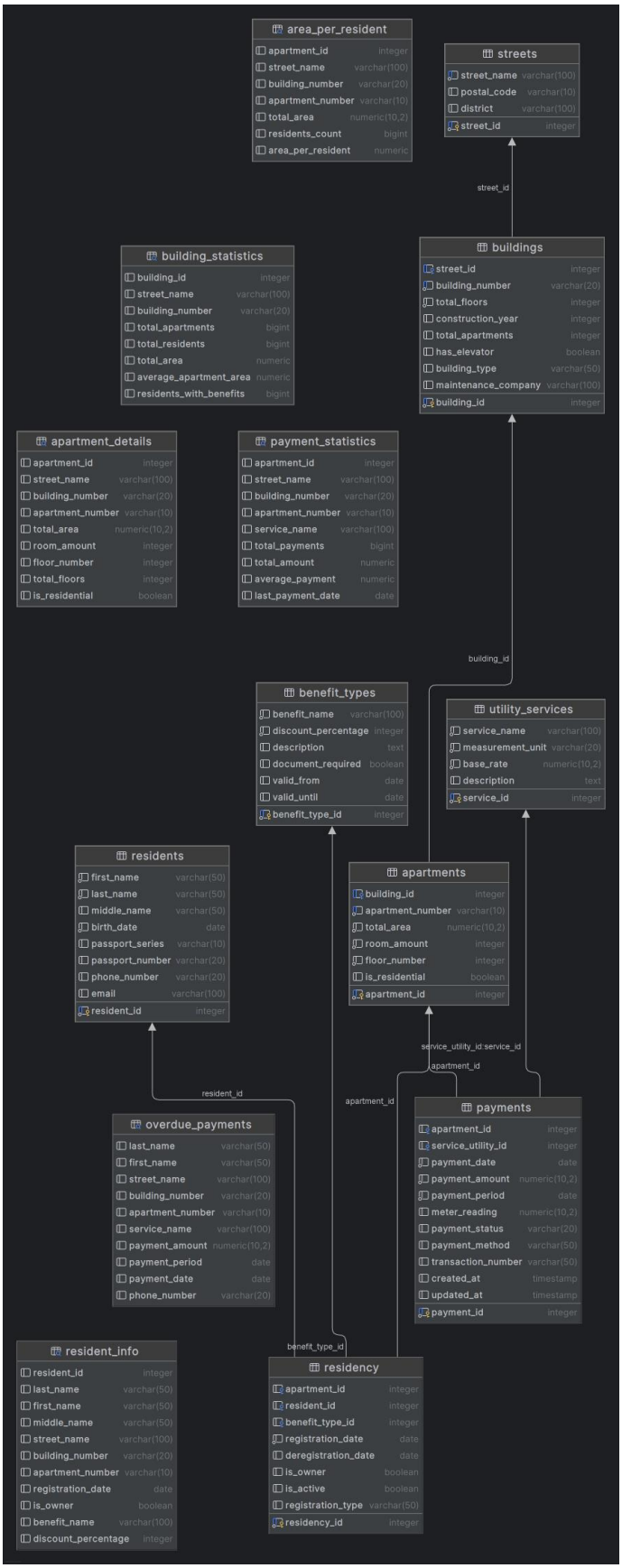
```
SELECT
   s.district,
   COUNT(DISTINCT s.street_id) as streets_count,
   COUNT(DISTINCT b.building_id) as buildings_count,
   COUNT(DISTINCT a.apartment_id) as apartments_count,
   SUM(b.total_apartments) as total_possible_apartments
FROM streets s
       LEFT JOIN buildings b ON s.street_id = b.street_id
       LEFT JOIN apartments a ON b.building_id = a.building_id
GROUP BY s.district
ORDER BY buildings_count DESC;
```

**8.1 Загальна статистика платежів:**

```
SELECT
   payment_status,
   COUNT(*) as total_payments,
   ROUND(AVG(payment_amount), 2) as avg_payment,
   SUM(payment_amount) as total_amount,
   COUNT(DISTINCT apartment_id) as unique_apartments
FROM payments
GROUP BY payment_status
ORDER BY total_amount DESC;
```

**8.2 Аналіз платежів за періодами:**

```
SELECT
    payment_period,
    us.service_name,
    COUNT(*) as payments_count,
    SUM(p.payment_amount) as total_amount,
    ROUND(AVG(p.meter_reading), 2) as avg_consumption,
    COUNT(CASE WHEN p.payment_status = 'overdue' THEN 1 END) as
overdue_count
FROM payments p
        JOIN utility_services us ON p.service_utility_id = us.service_id
GROUP BY payment_period, us.service_id, us.service_name
ORDER BY payment_period DESC, total_amount DESC;
```


area_per_resident
apartment_id integer
street_name varchar(100)
building_number varchar(20)
apartment_number varchar(10)
total_area numeric(10,2)
residents_count bigint
area_per_resident numeric
streets
street_name varchar(100)
postal_code varchar(10)
district varchar(100)
street_id integer
street_id
building_statistics
building_id integer
street_name varchar(100)
building_number varchar(20)
total_apartments bigint
total_residents bigint
total_area numeric
average_apartment_area numeric
residents_with_benefits bigint
buildings
street_id integer
building_number varchar(20)
total_floors integer
construction_year integer
total_apartments integer
has_elevator boolean
building_type varchar(50)
maintenance_company varchar(100)
building_id integer
apartment_details
apartment_id integer
street_name varchar(100)
building_number varchar(20)
apartment_number varchar(10)
total_area numeric(10,2)
room_amount integer
floor_number integer
total_floors integer
is_residential boolean
payment_statistics
apartment_id integer
street_name varchar(100)
building_number varchar(20)
apartment_number varchar(10)
service_name varchar(100)
total_payments bigint
total_amount numeric
average_payment numeric
last_payment_date date
building_id
benefit_types
benefit_name varchar(100)
discount_percentage integer
description text
document_required boolean
valid_from date
valid_until date
benefit_type_id integer
utility_services
service_name varchar(100)
measurement_unit varchar(20)
base_rate numeric(10,2)
description text
service_id integer
residents
first_name varchar(50)
last_name varchar(50)
middle_name varchar(50)
birth_date date
passport_series varchar(10)
passport_number varchar(20)
phone_number varchar(20)
email varchar(100)
resident_id integer
apartments
building_id integer
apartment_number varchar(10)
total_area numeric(10,2)
room_amount integer
floor_number integer
is_residential boolean
apartment_id integer
service_utility_id:service_id
apartment_id
resident_id
apartment_id
overdue_payments
last_name varchar(50)
first_name varchar(50)
street_name varchar(100)
building_number varchar(20)
apartment_number varchar(10)
service_name varchar(100)
payment_amount numeric(10,2)
payment_period date
payment_date date
phone_number varchar(20)
payments
apartment_id integer
service_utility_id integer
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
payment_id integer
resident_info
resident_id integer
last_name varchar(50)
first_name varchar(50)
middle_name varchar(50)
street_name varchar(100)
building_number varchar(20)
apartment_number varchar(10)
registration_date date
is_owner boolean
benefit_name varchar(100)
discount_percentage integer
benefit_type_id
residency
apartment_id integer
resident_id integer
benefit_type_id integer
registration_date date
deregistration_date date
is_owner boolean
is_active boolean
registration_type varchar(50)
residency_id integer