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**

1. ЗМІСТ.............................................................................................3

2. ВСТУП.............................................................................................4

3. ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА.........................................5-6

4. ПОСТАНОВКА ЗАВДАННЯ...........................................................7-8

5. ПРОЕКТУВАННЯ БАЗИ ДАНИХ...................................................9-13

4.1. Побудована ER-модель.......................................................9

4.2. Даталогічна модель бази даних........................................10-13

6. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ.........................................................14-18

5.1. Структура бази даних.......................................................14

5.2. Обґрунтування вибору типів даних.................................17

5.3. Зв'язки між таблицями...................................................17

5.4. Індексація та оптимізація...............................................17-18

7. РОБОТА З БАЗОЮ ДАНИХ.........................................................19-44

6.1. Робота з таблицею APARTMENTS....................................19-20

6.2. Робота з таблицею BENEFIT\_TYPES...............................20

6.3. Робота з таблицею BUILDINGS.......................................21-22

6.4. Робота з таблицею RESIDENTS.......................................21-22

6.5. Робота з таблицею RESIDENCY......................................22-23

6.6. Робота з таблицею STREETS..........................................23-24

6.7. Робота з таблицею UTILITY\_SERVICES...........................24-25

6.8. Робота з таблицею PAYMENTS.......................................25-44

8. ВИСНОВОК.................................................................................26-30

9. ДОДАТКИ...................................................................................31-44

8.1. SQL-скрипти створення та роботи з таблицями..............31-44

**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.

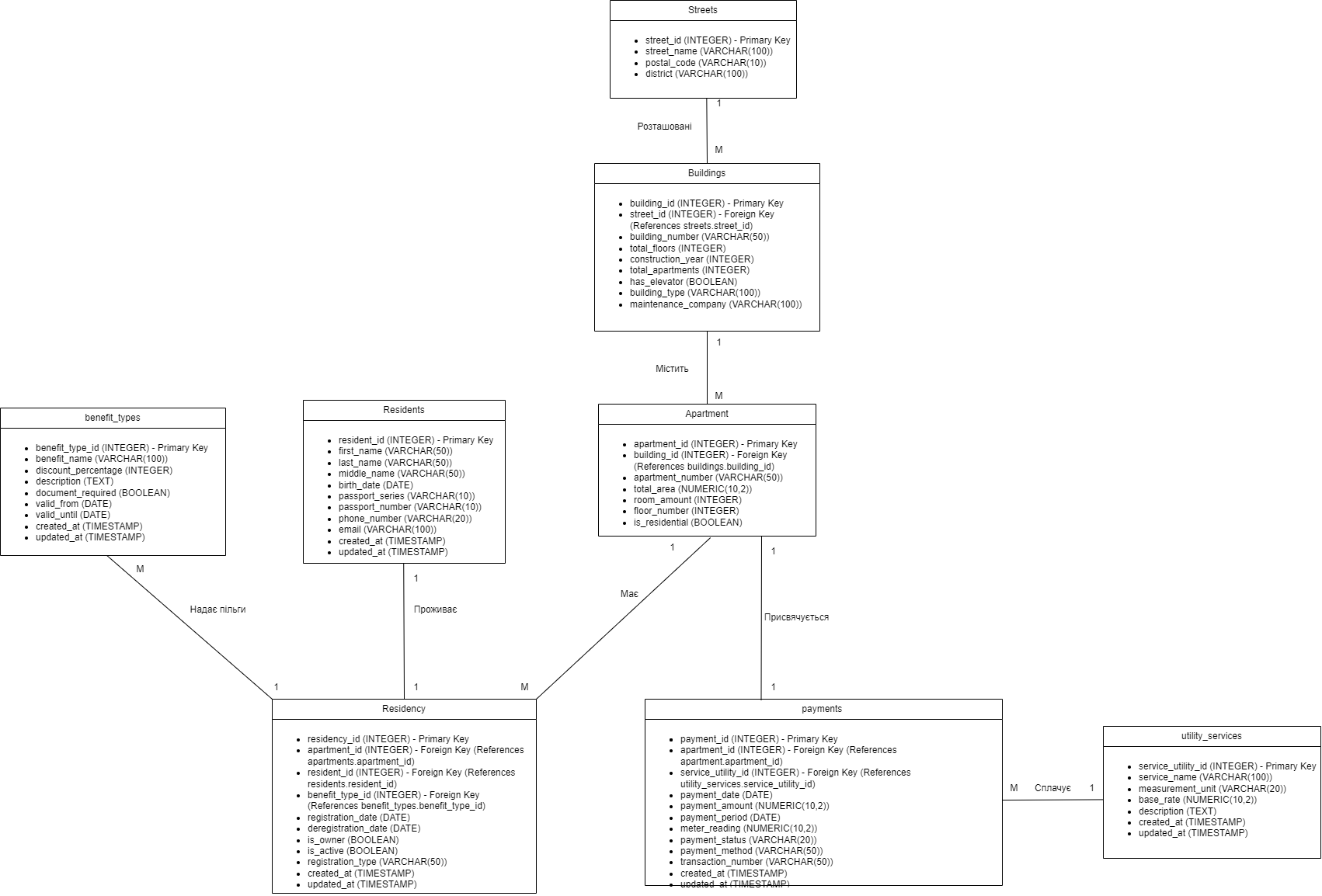
**Очікуваний результат**

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;

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;

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;

І**нформація 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. Отримати список всіх квартир з їх основними характеристиками:
  2. Знайти всі квартири з площею більше 100 м² та більше 2 кімнат:
  3. Отримати статистику квартир за поверхами
  4. Знайти квартири з найбільшою кількістю мешканців

1. **РОБОТА З ТАБЛИЦЕЮ BENEFIT\_TYPES**
   1. Отримати список всіх активних пільг з сортуванням за розміром знижки
   2. Знайти пільги з максимальною знижкою
   3. Знайти пільги, термін дії яких закінчується протягом найближчих 3 місяців
   4. Отримати список пільг із зазначенням їх актуального статусу
2. **РОБОТА З ТАБЛИЦЕЮ BUILDINGS**
   1. Отримати загальну інформацію про всі будинки
   2. C татистика будинків за типами
   3. Аналіз будинків за наявністю ліфтів
3. **РОБОТА З ТАБЛИЦЕЮ RESIDENTS**
   1. Отримати повний список мешканців з основною інформацією
   2. Знайти мешканців за віковими групами
   3. Отримати список мешканців з їх адресами проживання
   4. Знайти мешканців з пільгами:
4. **РОБОТА З ТАБЛИЦЕЮ RESIDENCY**
   1. Аналіз історії реєстрацій та виписок:
   2. Знайти квартири без активних мешканців:
5. **РОБОТА З ТАБЛИЦЕЮ UTILITY\_SERVICES**
   1. Отримати повний список комунальних послуг з тарифами
   2. Аналіз платежів за кожною послугою
   3. Аналіз послуг за методами оплати:
6. **РОБОТА З ТАБЛИЦЕЮ STREETS**
   1. Отримати базову інформацію про всі вулиці
   2. Отримати базову інформацію про всі вулиці
7. **РОБОТА З ТАБЛИЦЕЮ PAYMENTS**
   1. Загальна статистика платежів
   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)

1. 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

1. Забезпечено цілісність даних через:

* Встановлення первинних та зовнішніх ключів
* Додавання необхідних обмежень та перевірок
* Створення індексів для оптимізації пошуку

1. Розроблено набір SQL-запитів для:

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

1. 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;

