1. Оригінальна структура таблиць

Таблиці

customers(id, name, email)
employees(id, name, position)
categories(id, name)
products(id, name, price, category_id)
orders(id, order_date, customer_id, employee_id)
order_items(order_id, product_id, quantity)

2. Функціональні залежності

1. customers: id → name, email
Пояснення: Кожен клієнт має унікальний id, за яким можна однозначно визначити його ім’я та електронну пошту.
Це приклад повної функціональної залежності: усі атрибути залежать лише від ключа.

2. employees: id → name, position
Пояснення: Кожен працівник має унікальний id, що однозначно визначає його ім’я та посаду.
Теж класичний приклад функціональної залежності від первинного ключа.

3. categories: id → name
Пояснення: Категорія визначається унікальним id, за яким можна знайти її назву (name).
В таблиці лише один додатковий атрибут, і він повністю залежить від ключа.

4. products: id → name, price, category_id
Пояснення: Товар має унікальний id, який визначає його назву, ціну та категорію.

5. orders: id → order_date, customer_id, employee_id
Пояснення: Замовлення має унікальний id, який повністю визначає дату створення, клієнта та працівника, який його обробив.

Жоден з цих атрибутів не залежить один від одного — лише від id.

6. order_items: (order_id, product_id) → quantity

Пояснення: У таблиці order_items використовується складений ключ з двох полів: order_id та product_id.

Кожен запис — це конкретна позиція товару в замовленні, і кількість (quantity) залежить одночасно від order_id та product_id.

3.Аналіз нормальних форм (1NF → 3NF)

1. Таблиця products

1NF: Всі значення атомарні, жодних списків або повторюваних груп немає.

2NF: Первинний ключ простий (id), і всі інші атрибути (name, price, category_id) залежать від усього ключа.

3NF: Атрибути не залежать один від одного. Зв'язок з категоріями реалізований через зовнішній ключ (category_id).

2. Таблиця orders

1NF: Значення атомарні. Немає групованих або повторюваних значень.

2NF: Ключ простий (id), всі атрибути (order_date, customer_id, employee_id) залежать від нього.

3NF: Всі атрибути без транзитивної залежності. Наприклад, employee_id не залежить від customer_id.

3. Таблиця order_items

1NF: Значення атомарні (quantity — число).

2NF: Первинний ключ складений (order_id, product_id), і quantity залежить від обох складових, а не лише від однієї.

3NF: Немає атрибутів, які залежать від інших неключових полів.

4. Таблиця customers

1NF: Дані атомарні: ім'я і email — окремі значення.

2NF: Ключ простий (id), інші поля залежать від нього.

3NF: Немає залежностей типу name → email або подібних.

5. Таблиця employees

1NF: Ім’я та посада подані окремо, значення атомарні.

2NF: Первинний ключ — id, всі атрибути залежать від нього.

3NF: Немає транзитивних залежностей.

6. Таблиця categories

1NF: Одне поле name — атомарне.

2NF: name залежить від id.

3NF: Тільки один атрибут, жодних внутрішніх залежностей.

4.Гіпотетичне удосконалення

Гіпотетична помилка в orders

Уявна зміна:

Додамо в таблицю orders поле customer_email, щоб зберігати email клієнта прямо в замовленні:

```
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    customer_id INTEGER NOT NULL,
    customer_email VARCHAR(100), -- дублює дані
    employee_id INTEGER NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers(id),
    FOREIGN KEY (employee_id) REFERENCES employees(id)
);
```

Проблема:

customer_email вже є в таблиці customers.

Це створює транзитивну залежність: id → customer_id → customer_email.

Порушує 3NF, оскільки customer_email залежить не напряму від первинного ключа orders.id, а через іншу колонку.

Виправлення:

Усунути дублювання: Видалити customer_email з таблиці orders.

Витягати email через JOIN, а не зберігати дубль:

```
SELECT orders.id, customers.email
FROM orders
JOIN customers ON orders.customer_id = customers.id;
```

Зміна схеми назад до правильної:

```
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    customer_id INTEGER NOT NULL,
    employee_id INTEGER NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers(id),
    FOREIGN KEY (employee_id) REFERENCES employees(id)
);
```

5. Висновок

На підставі аналізу, вся структура відповідає нормальним формам 1NF, 2NF і 3NF. Немає надлишковості, повторюваних груп, часткових чи транзитивних залежностей. Усі ключі визначені, зовнішні зв’язки — правильно реалізовані.