Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Інститут комп’ютерних систем

Кафедра інформаційних систем

Козуб Катерина Олексіївна,

студент групи АІ-212

**ЗВІТ**

з виробничої практики

Спеціальність:

122 Комп’ютерні науки

Освітня програма: Комп’ютерні науки

Керівник від НУОП Керівник від

(науковий ступінь, вчене звання) Keepsolid

(прізвище та ініціали) (підпис) (прізвище ініціали) (підпис)

« » 2024 р. « » 2024 р.

Одеса 2024

ЗМІСТ

[ВСТУП 3](#_Toc159930055)

[1. ОРГАНІЗАЦІЙНА, ФУНКЦІОНАЛЬНА ТА СТРУКТУРНА ХАРАКТЕРИСТИКА ПІДПРИЄМСТВА 4](#_Toc159930056)

[2. ПРАКТИКА НА РОБОЧОМУ МІСТІ 6](#_Toc159930057)

[3. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ 7](#_Toc159930058)

[3.1. Підходи до вибору СУБД 7](#_Toc159930059)

[3.1.1. Реляційна модель даних 7](#_Toc159930060)

[3.1.2. Нереляційна (NoSQL) модель даних 8](#_Toc159930061)

[3.2. Порівняльний аналіз типів СУБД 8](#_Toc159930062)

[3.2.1. Переваги та недоліки реляційних СУБД 9](#_Toc159930063)

[3.2.2. Переваги та недоліки нереляційних (NoSQL) СУБД 9](#_Toc159930064)

[3.2.3. Приклади використання кожного типу СУБД у різних сферах 10](#_Toc159930065)

[3.3. Критерії вибору СУБД 11](#_Toc159930066)

[3.4. 4 Реалізація бази даних. 16](#_Toc159930067)

[ВИСНОВКИ 30](#_Toc159930068)

[ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ 31](#_Toc159930069)

ВСТУП

Виробнича практика відіграє важливу роль у формуванні професійних компетенцій студентів і спрямована на декілька ключових завдань:

* Систематизація, закріплення та розширення теоретичних та практичних знань у відповідності з обраною спеціальністю.
* Розвиток навичок самостійної роботи в умовах реального виробництва та сучасних технологічних процесів.

Метою виробничої практики є:

* Закріплення та розширення теоретичних і практичних знань, формування навичок самостійної практичної діяльності.
* Отримання практичного досвіду з експлуатації сучасних технічних засобів та програмного забезпечення.
* Ознайомлення з економічними, організаційними та технологічними аспектами діяльності підприємства.

Завдання виробничої практики включають:

* Ознайомлення з внутрішньою структурою та процесами функціонування підприємства.
* Оволодіння практичними навичками в області використання сучасних технологій та обладнання.
* ивчення ринкових взаємовідносин та особливостей взаємодії з клієнтами та партнерами.

1. ОРГАНІЗАЦІЙНА, ФУНКЦІОНАЛЬНА ТА СТРУКТУРНА ХАРАКТЕРИСТИКА ПІДПРИЄМСТВА

Розповісти про характеристики підприємства "KeepSolid" можна в контексті організаційної, функціональної та структурної перспективи:

Організаційна характеристика:

* Компанія "KeepSolid" відома своїм спеціалізованим програмним забезпеченням для захисту приватності та безпеки в Інтернеті, включаючи віртуальні приватні мережі (VPN) та інші цифрові рішення.
* Вона може мати різноманітну організаційну структуру, залежно від розміру та потреб бізнесу. Наприклад, вона може бути організована у вигляді функціональних відділів, проектних команд або матричної структури.

Функціональна характеристика:

* "KeepSolid" спеціалізується на розробці та підтримці програмного забезпечення для безпеки Інтернету, яке може включати VPN-сервіси, шифрування даних та інші цифрові рішення.
* У межах функціональної структури можуть існувати різні відділи або підрозділи, такі як девелопмент, маркетинг, продажі, технічна підтримка тощо.

Структурна характеристика:

* Структура "KeepSolid" може бути побудована таким чином, щоб максимально відповідати вимогам бізнесу. Наприклад, вона може мати плоску структуру, де всі рівні ієрархії мають доступ до керівництва, або ж ієрархічну структуру з чітко визначеними ланцюгами командування.
* Можливо, вона також має матричну структуру, де працівники працюють як у функціональних відділах, так і в проектних командах.

Ці аспекти можуть допомогти зрозуміти, як підприємство "KeepSolid" організоване та функціонує з точки зору управління та ділових операцій.

1. ПРАКТИКА НА РОБОЧОМУ МІСТІ

Протягом тривалості практики ми зосереджувалися на вивченні та аналізі різних аспектів роботи в компанії. Основною метою було дослідження та оцінка потенційних напрямків розвитку, а також визначення наших можливостей у цьому процесі.

Щотижня кожен учасник представляв свої результати та пропозиції щодо подальших кроків. Ми зосереджували увагу на аналізі конкурентного оточення та виявленні ринкових можливостей для продуктів компанії.

Моя особиста тема була у створення мобільного додатка “FoodFinder”. Основна мета полягала в тому, щоб продемонструвати функціонал та особливості мобільного додатка FoodFinder, розробленого компанією для пошуку та замовлення їжі у ресторанах та кафе, та багато інших функцій.

Крім того, ми вивчали концепцію "виконання роботи" (JTBD), яка допомагала нам краще розуміти потреби наших клієнтів та спрямовувати розробку продукту у відповідності з цими потребами.

В результаті практики ми отримали цінний досвід у розробці програмного забезпечення, а також вміння аналізувати ринкові тенденції та прогнозувати їхні зміни.

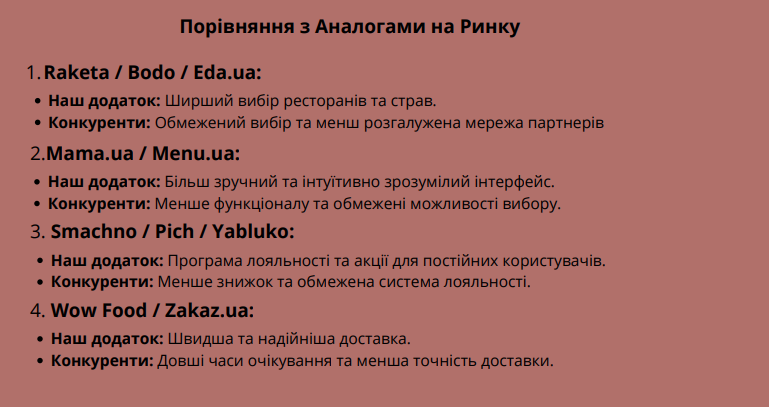


Рис 2.1 - Аналіз ринку під час створення концепції мобільного додатку

1. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

**Тема:** Огляд підходів, критерії вибору СУБД при створенні інформаційних систем. Реалізація бази даних на будь-яку тему узгоджену з викладачем.

3.1. Підходи до вибору СУБД

3.1.1. Реляційна модель даних

Реляційна база даних (РБД) - це структурована колекція даних, організована у вигляді таблиць. Кожна таблиця містить рядки (записи) і стовпці (поля), що визначають типи даних. РБД використовує зв'язки між таблицями для відображення та збереження залежностей між даними.[1]

Перед тим як розглядати деталі роботи з реляційною базою даних, давайте розглянемо основні поняття, які з нею пов’язані:

* Таблиці: Це основна структурна одиниця реляційної бази даних, що складається з рядків і стовпців.
* Рядки і стовпці: Рядок представляє конкретний запис у таблиці, а стовпець визначає тип даних, який може бути збережений у цьому полі.
* Ключі: Використовуються для унікальної ідентифікації записів у таблиці. Первинний ключ визначає унікальний ідентифікатор запису, а зовнішній ключ встановлює зв'язок між таблицями.
* Зв'язки між таблицями: Встановлюють зв'язки між даними. Найпоширеніший тип зв'язку - "один до багатьох", де один запис у першій таблиці відповідає багатьом записам у другій таблиці.[1]

Один із способів взаємодії з реляційною базою даних – це використання мови структуризованих запитів SQL (Structured Query Language). SQL надає набір команд, які можна використовувати для створення, модифікації і вибірки даних у реляційних базах даних.[1]

3.1.2. Нереляційна (NoSQL) модель даних

Нереляційна база даних (NoSQL) – зберігає дані без чітких зв’язків між собою та без чіткої структури. Замість структурованих таблиць, всередині бази знаходиться безліч різнорідних документів, в тому числі і зображення, відео та навіть публікації у соціальних мережах. На відміну від реляційних БД, NoSQL бази не підтримують SQL запити. [3]

Бази даних NoSQL підходять для зберігання великих об’ємів неструктурованої інформації, а також для швидкої розробки та тестування гіпотез.[3]

Масштабованість. NoSQL бази мають розподілену архітектуру, тому легко масштабуються горизонтально та вирізняються високою продуктивністю. Технології NoSQL можуть автоматично розподіляти дані на різних серверах. Це підвищує швидкість читання даних у розподіленому середовищі.[3]

3.2. Порівняльний аналіз типів СУБД

3.2.1. Переваги та недоліки реляційних СУБД

Переваги реляційної бази даних:

* Зручність використання: Реляційна модель даних є логічною і зрозумілою, що робить її зручною для розробників і користувачів.
* Ефективність: Реляційна база даних дозволяє ефективно зберігати інформацію і проводити швидкий доступ до неї за допомогою оптимізованих запитів SQL.
* Надійність: Реляційна база даних забезпечує цілісність даних за допомогою використання ключів і обмежень
* Масштабованість: Реляційна база даних може бути легко масштабована для обробки великих обсягів даних і великої кількості користувачів.[1]

Недоліки реляційної бази даних:

* Обмеження структури: Реляційні бази даних вимагають фіксованої структури, що може бути незручним у випадку, коли потрібно зберігати динамічні дані або дані зі змінюючоюся структурою.
* Проблеми з продуктивністю при великих обсягах даних: При роботі з великими обсягами даних реляційні бази можуть стикатися з проблемами продуктивності, оскільки запити можуть вимагати значних ресурсів обчислювальної системи.
* Складність моделювання зв’язків: У випадку складних залежностей між даними може виникати складність при моделюванні зв’язків між таблицями.[1]

3.2.2. Переваги та недоліки нереляційних (NoSQL) СУБД

Переваги:

* Масштабування. Зазвичай виконується завдяки PartitionKey. Це фрагмент даних, про який відомо, що він лежить в одному місці. В цьому полягає суть горизонтального масштабування. Якщо ми кладемо дані заздалегідь, тому що знаємо PartitionKey, масштабування буде майже безмежним.
* Позбавлення Object-relational impedance mismatch. На прикладах з MongoDB я показав, що дані можуть мати потрібні схему та структуру. Для цього не треба писати складних джойнів або моделювати дані не так, як ви їх використовуєте.
* Schemaless. В одну колекцію можна інсертити об’єкти будь-якої форми так, як вам зручно. Хоча, звичайно, це може бути й недоліком NoSQL.[2]

Недоліки:

* Часткова підтримка транзакцій. Різні парадигми NoSQL та окремі БД частково підтримують концепти ACID, але загалом їх немає. Ними пожертвували задля масштабування системи.
* Відмова від реляційності. Якщо у вас немає джойнів, то у вас з’являється… дублювання даних. Тут важливо розуміти: це норма для подібної системи — не баг, а фіча, як-то кажуть. Тому потрібно моделювати дані таким чином, аби подібна система була настільки ефективною, наскільки це реально.
* Обмежені можливості пошуку по складних критеріях. У NoSQL є find, як я згадував у прикладі. Однак у глобальному концепті масштабування є проблеми. Якщо дані лежать на великій кількості нод, то для пошуку за певним критерієм без Partition, треба обійти буквально всі ноди. Це фактично нівелює масштабування як таке.[2]

3.2.3. Приклади використання кожного типу СУБД у різних сферах

Реляційні бази даних (РБД) широко використовують у реальних проектах у багатьох галузях і сферах. Ось приклади:

* Ecommerce: РБД використовуються для управління даними про продукти, клієнтів, замовлення та інвентаризацію в інтернет-магазинах і платформах електронної комерції.
* Фінанси: Банки, фінансові установи та страхові компанії використовують РБД для зберігання даних про клієнтів, транзакції, активи та пасиви.
* Охорона здоров’я: Медичні установи використовують РБД для управління пацієнтськими даними, медичними записами, призначеннями та розкладами.
* Логістика та управління ланцюгами поставок: РБД використовуються для відстеження та управління інвентарем, доставками, постачальниками і замовленнями.
* Аналітика: РБД є джерелом даних для аналізу та звітності. Вони дають змогу бізнесам ухвалювати поінформовані рішення на основі даних.[4]

Нереляційні (NoSQL) СУБД:

* Веб-аналітика: Платформи веб-аналітики можуть використовувати NoSQL бази даних для швидкого зберігання та аналізу великих обсягів неструктурованих даних про користувачів.
* Інтернет речей (IoT): Для зберігання даних від сенсорів та пристроїв IoT, таких як дані з медичних пристроїв або датчиків в промислових системах, часто використовуються NoSQL бази даних.
* Геопросторові додатки: Додатки, які працюють з геолокаційними даними, такими як карти, навігація або географічні ігри, можуть використовувати NoSQL бази даних для ефективного зберігання та опрацювання геоданих

3.3. Критерії вибору СУБД

Вибір системи управління базами даних представляє собою складну задачу та є одним з важливих етапів при розробці інформаційних систем. Вибраний програмний продукт повинен задовольняти як поточним, так і майбутнім потребам підприємства. При цьому слід враховувати фінансові затрати на придбання необхідного обладнання, самої системи, розробку програмного забезпечення та навчання персоналу. І найголовніше – необхідно впевнитися в тому, що нова СУБД може принести підприємству реальні вигоди.[5]

Очевидно, що найпростіший підхід до вибору СУБД базується на оцінці того, до якої міри існуючі системи задовольняють основні вимоги створюваного проекту інформаційної системи. Складнішим та дорожчим варіантом є попереднє створення прототипу ІС на основі кількох СУБД та наступний вибір найоптимальнішої. Проте і в цьому випадку потрібно обмежувати коло можливих систем, спираючись на деякі критерії відбору. Перелік вимог до СУБД може змінюватися залежно від поставлених цілей. Однак, все ж таки можна виділити декілька груп критеріїв.[5]

* моделювання даних;
* особливості архітектури та функціональні можливості;
* контроль роботи системи;
* особливості розробки програмного забезпечення;
* продуктивність;
* надійність;
* вимоги до робочого середовища;
* змішані критерії.[5]

Розглянемо окремо кожну з цих груп.

***Модель даних***. Існує багато різних моделей даних. Питання про використання тієї чи іншої моделі потрібно вирішувати на початковому етапі проектування інформаційної системи. На даному етапі розвитку тут може стояти питання про вибір між реляційною та об’єктною моделями.[5]

***Особливості архітектури та функціональні можливості***. До цієї групи відносять наступні критерії:

* **Тригери та збережені процедури**: Забезпечують автоматичну обробку подій у базі даних, що дозволяє здійснювати перевірку та обробку змін даних.
* **Засоби пошуку**: Деякі СУБД мають вбудовані інструменти контекстного пошуку, що полегшують пошук інформації у базі даних.
* **Доступні типи даних**: Включають базові типи даних і можливість розширення типів, що впливає на гнучкість та функціональність системи.
* **Реалізація мови запитів**: СУБД зазвичай підтримують стандарт SQL-99 та можуть мати додаткові розширення для роботи з даними
* **Мобільність**: Спроможність працювати у різних середовищах без втрати функціональності.
* **Масштабованість**: Здатність системи адаптуватися до зростання обсягів даних та користувачів.
* **Розподіленість**: Можливість управління розподіленими базами даних для спільного доступу до інформації у всій організації.
* **Мережні можливості**: Підтримка різних мережних протоколів та інструментів для роботи та адміністрування баз даних через мережу.[5]

Наступна група – ***контроль роботи системи*** містить такі критерії:

* **Контроль використання пам'яті комп'ютера**: Можливість системи управляти оперативною пам'яттю та дисковим простором, включаючи стискування баз даних та видалення надлишкових файлів.
* **Автонастроювання**: Здатність системи до самоконфігурування, що базується на результати самодіагностики продуктивності, для виявлення та налаштування слабких місць конфігурації з метою досягнення максимальної продуктивності.[5]

***Особливості розробки інформаційних систем*** засобами СУБД та для конкретної СУБД:

* **Засоби розробки програм**: Багато виробників СУБД надають засоби розробки програм, які дозволяють ефективно використовувати можливості сервера
* **Засоби проектування**: Деякі системи мають інструменти автоматичного проектування баз даних та програм, проте їх функціональність може значно відрізнятися.
* **Підтримка національних мов**: Можливість роботи з різними національними мовами розширює галузі використання системи та розроблених на її основі програм
* **Можливості розробки Web-програм**: Деякі СУБД надають широкий набір інструментів для створення Web-програм.
* **Підтримка різних мов програмування**: Різноманіття мов програмування забезпечує доступність системи для розробника та може вплинути на її швидкодію та функціональність.[5]

***Продуктивність*** СУБД:

* Тест TPC (Transactions per Cent) визначає ефективність системи, враховуючи вартість і кількість оброблених запитів.
* Можливості паралельної архітектури дозволяють розділити обробку даних між кількома процесорами або комп'ютерами, що прискорює роботу системи.
* Оптимізація запитів полягає у виборі оптимального способу виконання запитів для підвищення продуктивності системи.[5]

***Надійність***. Поняття надійності системи містить в собі різні трактування – це і забезпечення збереженості інформації незалежно від довільних збоїв, і безвідмовність роботи системи за будь-яких умов, і забезпечення захисту даних від несанкціонованого доступу:

* **Відновлення після збоїв.** При виникненні програмних чи апаратних збоїв цілісність і працездатність системи може бути порушена. Від того наскільки ефективно сплановано механізм відновлення після збоїв залежить життєздатність системи.
* **Резервне копіювання.** В результаті апаратного збою може бути частково пошкоджений чи виведений з ладу носій інформації, в результаті чого відновлення системи стає неможливим, якщо не було передбачене резервне копіювання бази даних чи її частини.
* **Відкат змін.** При виконанні трансакції використовується просте правило – або трансакція виконується повністю, або вона не виконується взагалі. Це значить, що у випадку виникнення збоїв усі результати незавершених трансакцій повинні бути анульовані. Механізм відкату може мати різну швидкодію та ефективність.
* **Багаторівнева система захисту.** Інформаційна система організації майже завжди містить секретну інформацію, в зв’язку з чим для унеможливлення несанкціонованого доступу використовується служба ідентифікації користувачів. Рівень захисту може бути різним. Крім безпосередньої ідентифікації користувачів при вході в систему також може використовуватися механізм шифрування даних при їх передаванні лініями зв’язку.[5]

***Вимоги до робочого середовища***:

* Підтримка різних апаратних платформ.
* Мінімальні вимоги до апаратного забезпечення.
* Максимальний розмір доступної (адресуємої) пам’яті. Оскільки більшість сучасних потужних СУБД використовують власну файлову систему, важливим фактором є те, який максимальний об’єм фізичної пам’яті вони можуть використовувати.
* Операційні системи, під управлінням яких може працювати СУБД.[5]

***Змішані критерії***:

* Якість і повнота документації. На жаль, далеко не усі системи містять повну і детальну документацію.
* Локалізація. Не в усіх системах повністю реалізовано можливість використання національних мов
* Модель формування вартості. Як правило, виробники СУБД використовують певні моделі формування вартості. Наприклад, вартість одного і того ж продукту може суттєво змінюватися залежно від того, скільки користувачів буде з ним працювати.
* Стабільність виробника.
* Розповсюдженість СУБД.[5]

3.4. 4 Реалізація бази даних.

Обрала я PostgreSQL тому що це потужна та надійна система управління реляційними базами даних (СУБД), яка має відкритий вихідний код та широкі можливості для зберігання, організації та обробки даних. Також вона підтримує багато функціональності, включаючи розширені можливості SQL, підтримку геопросторових даних та високий рівень безпеки. PostgreSQL є популярним та широко використовуваним рішенням, яке має активну спільноту користувачів та розробників, що забезпечує підтримку та постійний розвиток продукту.

Я обрала PgAdmin тому що це потужний інструмент для адміністрування та управління базами даних PostgreSQL. Він надає зручний графічний інтерфейс, який дозволяє легко створювати та адмініструвати бази даних, виконувати SQL-запити, переглядати дані, керувати користувачами та дозволами, моніторити працездатність бази даних та багато іншого. Крім того, PgAdmin безкоштовний та має відкритий вихідний код, що робить його доступним для використання без додаткових витрат. Його активна спільнота користувачів і розробників також забезпечує підтримку та постійний розвиток інструменту.

Спочатку створення бази даних. В мене база даних це магазин квітів

CREATE DATABASE flowershop;

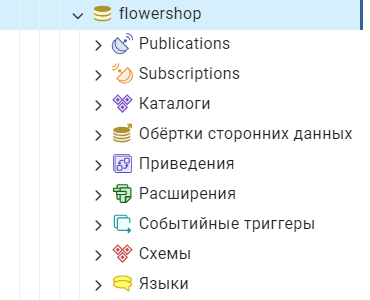


Рис. 3.4.1 – Створення БД

Створення таблиць

Створення таблиці “Квітів”:

CREATE TABLE Flowers (

ProductID INT PRIMARY KEY,

Name VARCHAR(255),

Description TEXT,

Price DECIMAL(10, 2),

QuantityInStock INT,

Category VARCHAR(100)

);

Створення таблиці “Клієнт”:

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

LastName VARCHAR(100),

FirstName VARCHAR(100),

Address VARCHAR(255),

PhoneNumber VARCHAR(20),

Email VARCHAR(100)

);

Створення таблиці “Замовлення”:

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalPrice DECIMAL(10, 2),

Status VARCHAR(50),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

Створення таблиці “Постачальники”:

CREATE TABLE Suppliers (

SupplierID INT PRIMARY KEY,

CompanyName VARCHAR(100),

ContactPerson VARCHAR(100),

Address VARCHAR(255),

PhoneNumber VARCHAR(20),

Email VARCHAR(100)

);

Створення таблиці “Оплата”:

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY,

OrderID INT,

Amount DECIMAL(10, 2),

PaymentDate DATE,

PaymentMethod VARCHAR(50),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

Створення таблиці “Поставки”:

CREATE TABLE Shipments (

ShipmentID INT PRIMARY KEY,

ProductID INT,

ShipmentDate DATE,

QuantityShipped INT,

PricePerUnit DECIMAL(10, 2),

FOREIGN KEY (ProductID) REFERENCES Flowers(ProductID)

);

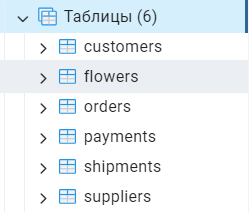


Рис. 3.4.2 -Таблиці

Заповнення таблиць:

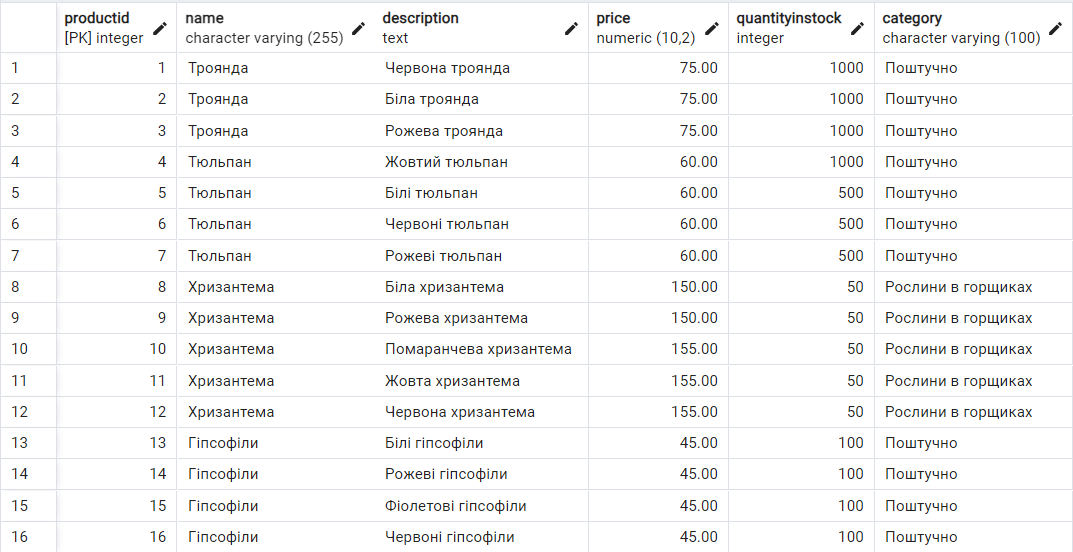


Рис 3.4.3 – Таблиця Квітів

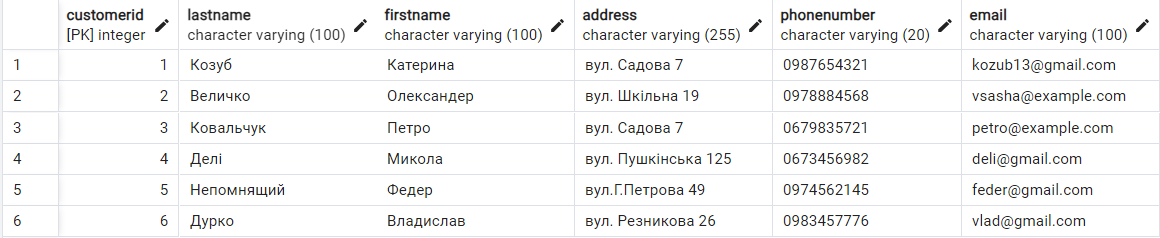


Рис 3.4.4 – Таблиця Клієнти

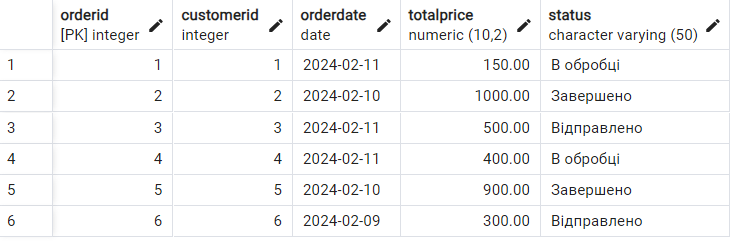


Рис 3.4.5 – Таблиця Замовлення

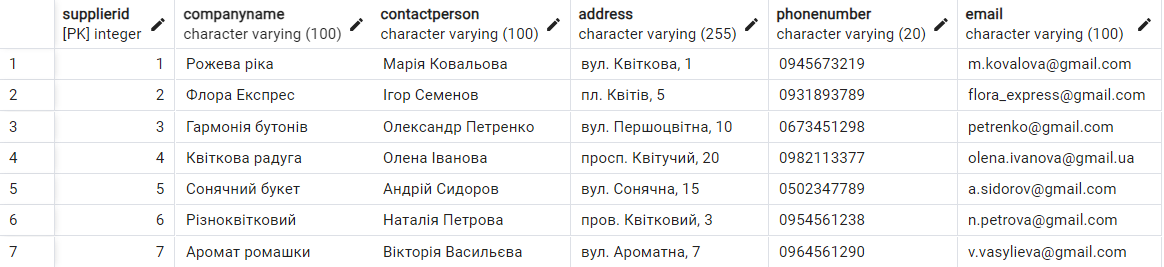


Рис 3.4.6 – Таблиця Постачальники

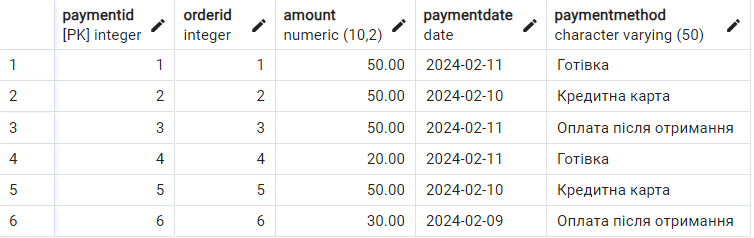


Рис 3.4.7 – Таблиця Оплати

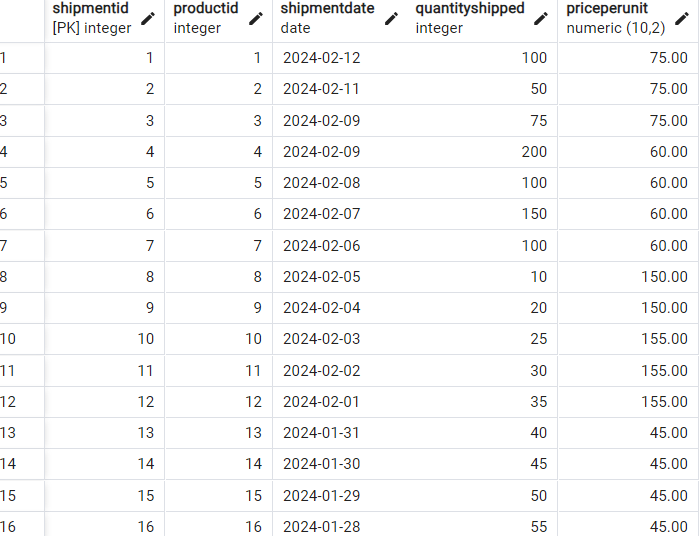


Рис 3.4.8 – Таблиці Поставки

Таблиця Квіти:

1. Цей запит вибирає всі записи з таблиці "Квіти", де значення стовпця "Категорія" дорівнює 'Поштучно'.

SELECT \* FROM Flowers WHERE Category = 'Поштучно';

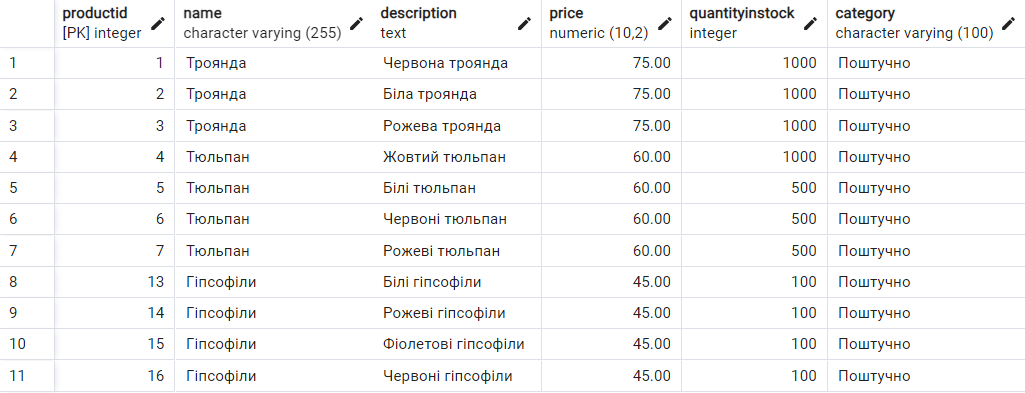


Рис 3.4.9 – Таблиця Операції SELECT

1. Цей запит мовою SQL є командою для вставки нового рядка даних в таблицю з назвою "Flowers"

INSERT INTO Flowers (ProductID, Name, Description, Price, QuantityInStock, Category)

VALUES (17, 'Ромашка', 'Біла ромашка', 55.00, 200, 'Поштучно');

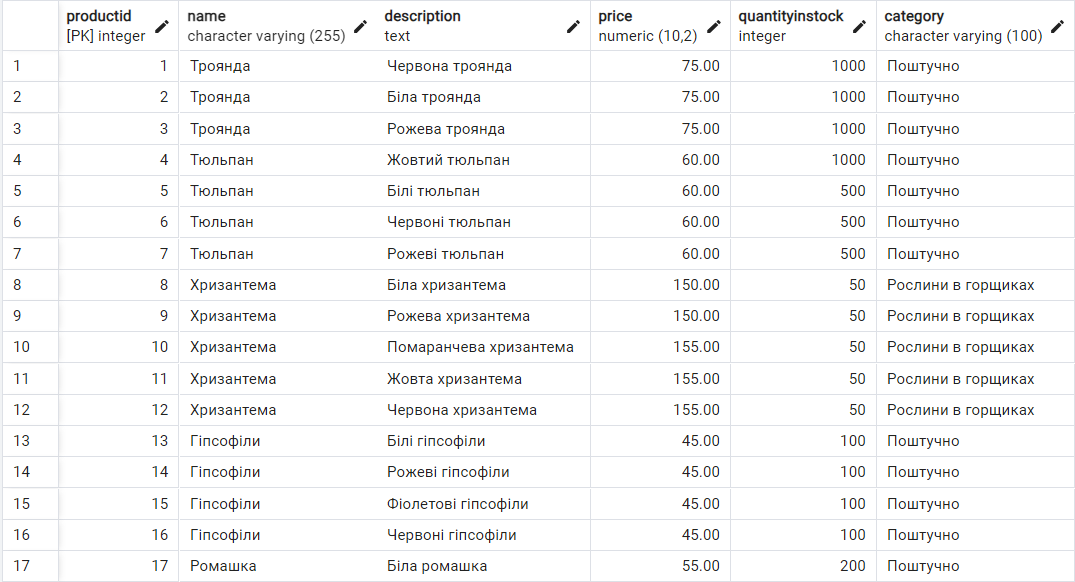


Рис 3.4.10 – Таблиця Операції INSERT INTO

1. Цей запит змінює ціну для всіх записів в таблиці "Flowers", де назва (Name) дорівнює 'Троянда', і встановлює ціну для них на 80.00.

UPDATE Flowers SET Price = 80.00 WHERE Name = 'Троянда';

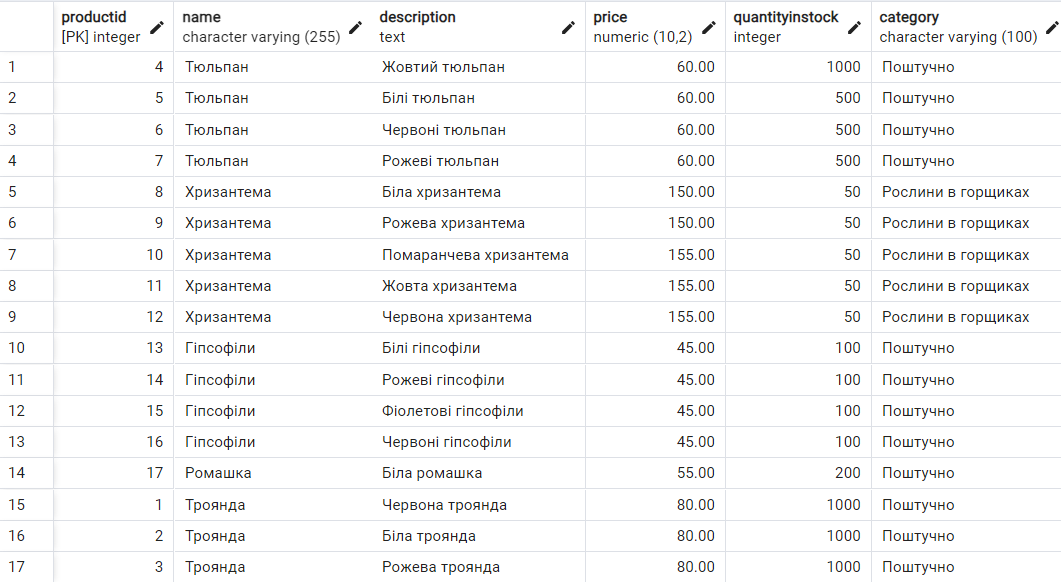


Рис 3.4.11 – Таблиця Операції UPDATE

1. Цей запит вибирає всі рядки з таблиці "Flowers" та сортує їх за полем "Price" у спадному порядку, тобто від найвищої до найнижчої ціни.

SELECT \* FROM Flowers ORDER BY Price DESC;

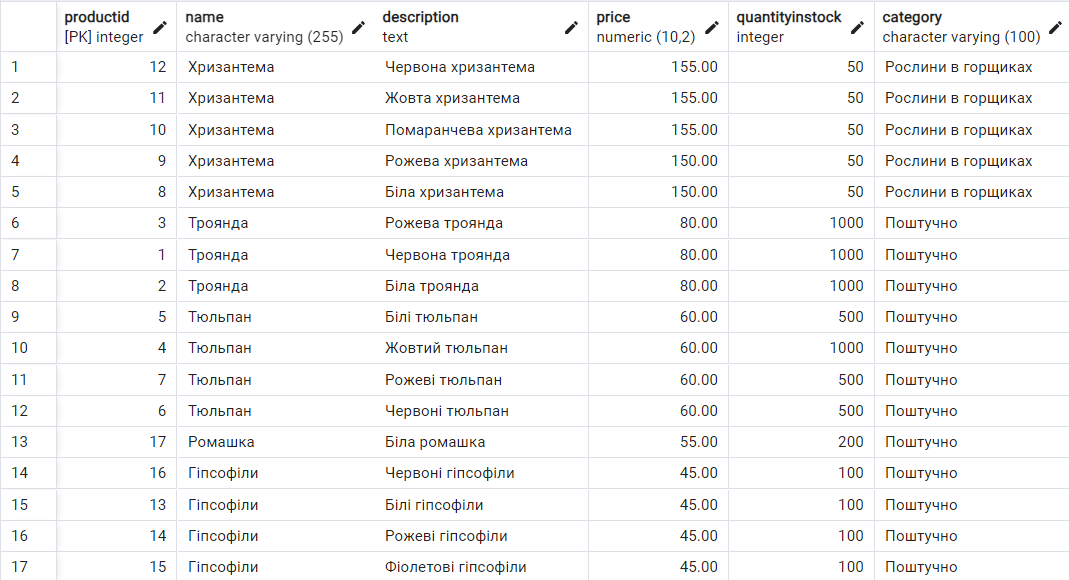


Рис 3.4.12 – Таблиця Операції ORDER BY

1. Цей запит використовується для підрахунку кількості записів у таблиці "Flowers" для кожної категорії

SELECT Category, COUNT(\*) FROM Flowers GROUP BY Category;

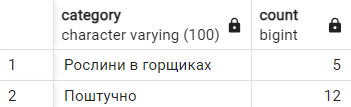


Рис 3.4.13 – Таблиця Операції GROUP BY

1. Цей SQL запит вирішує задачу знаходження максимальної ціни серед усіх квітів в базі даних.

SELECT MAX(Price) FROM Flowers;

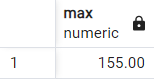


Рис 3.4.14 – Таблиця Операції MAX

1. Цей запит розширює схему таблиці "Квіти", дозволяючи зберігати інформацію про знижку, яка може бути застосована до цін квітів.

ALTER TABLE Flowers ADD Discount FLOAT;

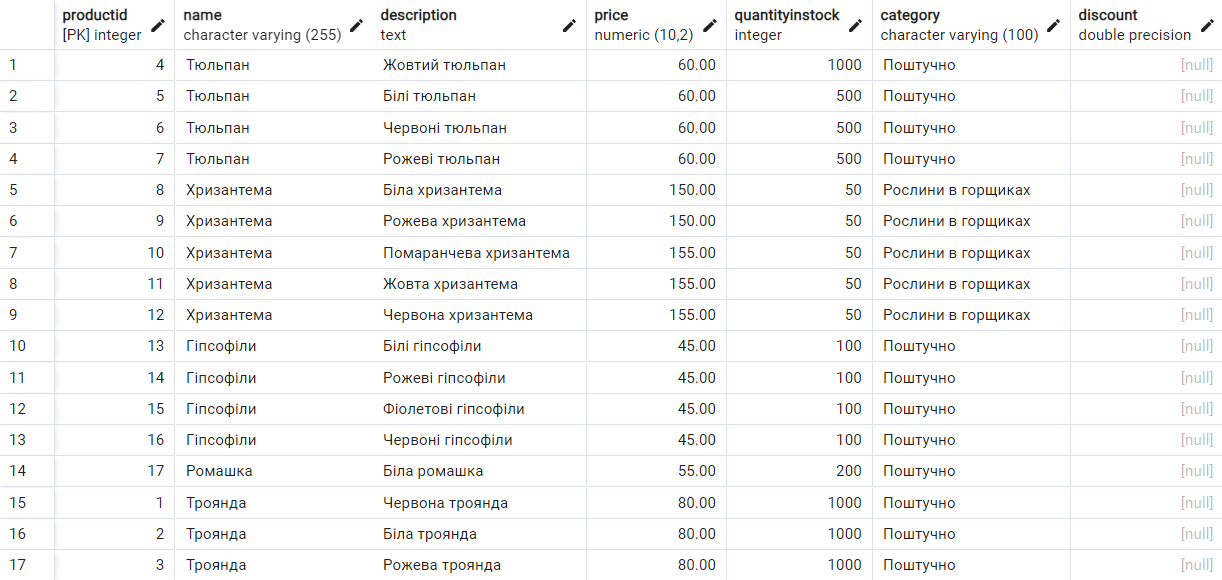


Рис 3.4.15 -Таблиця Операції ALTER TABLE

1. Цей блок SQL-запитів вирішує задачу оновлення ціни для квітки "Троянда" на 80.00 і вставки нової квітки "Лилия" з ціною 90.00 у таблицю "Квіти" в межах однієї транзакції, щоб зміни стали постійними.

BEGIN TRANSACTION;

UPDATE Flowers SET Price = 80.00 WHERE Name = 'Троянда';

INSERT INTO Flowers (ProductID, Name, Description, Price, QuantityInStock, Category)

VALUES (18, 'Лилия', 'Белая лилия', 90.00, 200, 'Поштучно');

COMMIT;

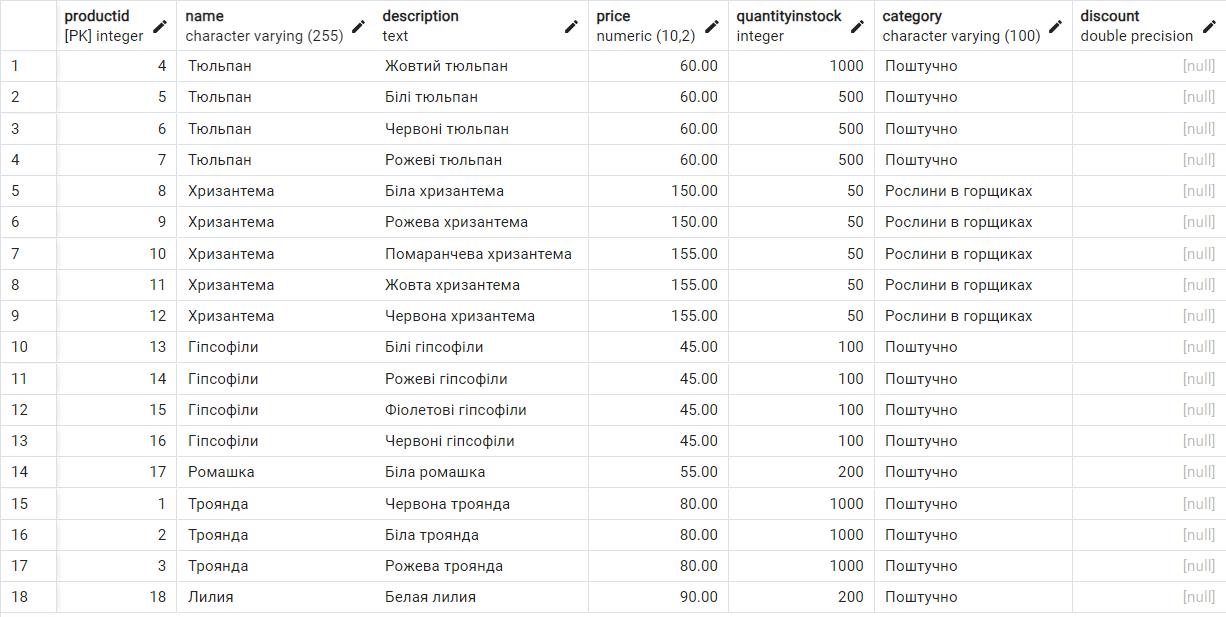


Рис 3.4.16 – Таблиця Транзакції

1. Цей блок SQL-запитів вибирає всі записи про квіти з таблиці, у яких ціна вища, ніж середнє значення ціни квітів у цій же таблиці.

SELECT \* FROM Flowers WHERE Price > (SELECT AVG(Price) FROM Flowers);

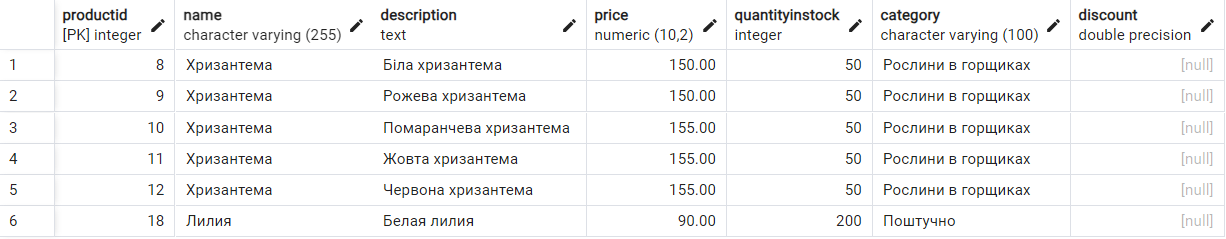


Рис 3.4.17 – Таблиця Використання підзапитів

1. Цей блок SQL-запитів вирішує задачу вибору перших 10 записів з таблиці "Квіти".

SELECT \* FROM Flowers LIMIT 10;

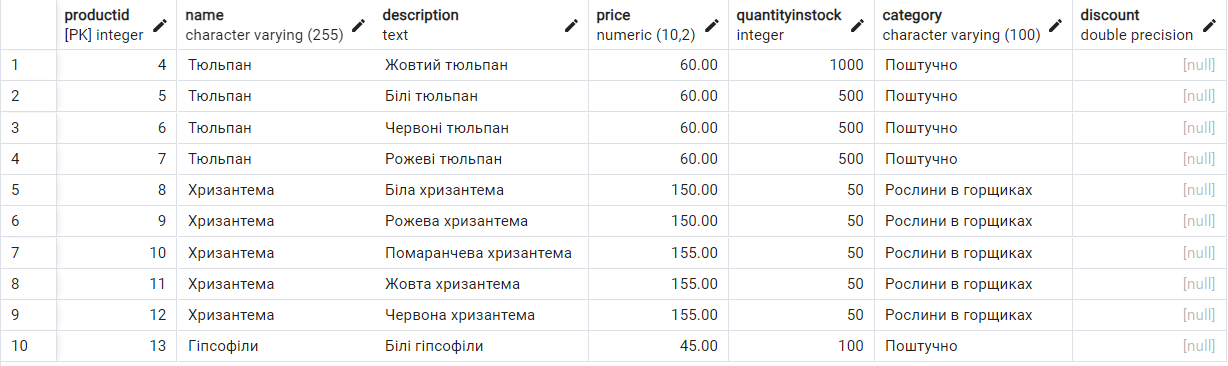


Рис 3.4.18 Таблиця Операції LIMIT

Таблиця Оплата:

1. Цей блок SQL-запитів вирішує задачу вибору всіх записів з таблиці " Оплата", де метод оплати (стовпець "PaymentMethod") дорівнює 'Готівка'.

SELECT \* FROM Payments WHERE PaymentMethod = 'Готівка'

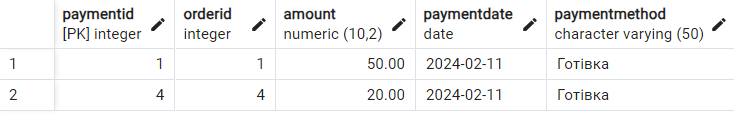


Рис 3.4.19 Таблиця Операції SELECT

1. Цей блок SQL-запитів вирішує задачу оновлення дати оплати для платежу з ідентифікатором 1 у таблиці " Оплата ". Він встановлює нову дату оплати ('2024-02-12') для платежу з ідентифікатором 1.

UPDATE Payments SET PaymentDate = '2024-02-12' WHERE PaymentID = 1

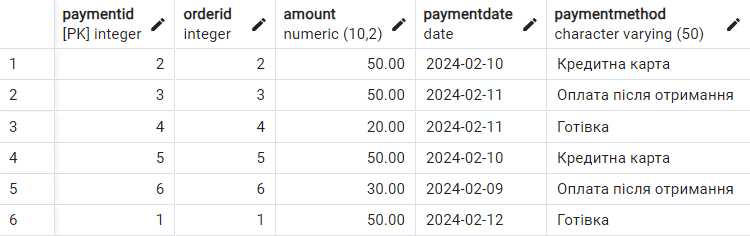


Рис 3.4.20 – Таблиця Операції UPDATE

1. Цей блок SQL-запитів вирішує задачу видалення запису з таблиці " Оплата ", де ідентифікатор платежу (стовпець "PaymentID") дорівнює 6.

DELETE FROM Payments WHERE PaymentID = 6

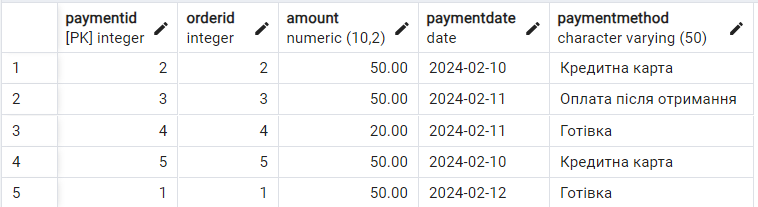


Рис 3.4.21 – Таблиця Операції DELETE

1. Цей блок SQL-запитів вирішує задачу підсумовування значень у стовпці "Сума" (**Amount**) з таблиці " Оплата ". Він обчислює загальну суму всіх платежів, що були здійснені.

SELECT SUM(Amount) FROM Payments;

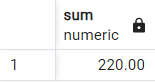


Рис 3.4.22 – Таблиця Операції SUM

Таблиця Поставок:

1. Цей блок SQL-запитів вирішує задачу вибору всіх записів з таблиці " Поставок", де дата відвантаження (**ShipmentDate**) більша за '2024-02-05'.

SELECT \* FROM Shipments WHERE ShipmentDate > '2024-02-05';

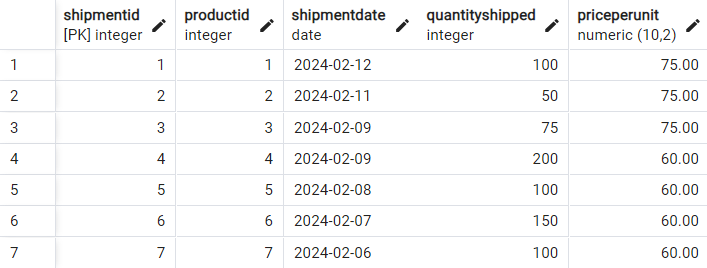


Рис 3.4.23 – Таблиця Операції SELECT

1. Цей блок SQL-запитів вирішує задачу оновлення кількості відвантажених одиниць для відвантаження з ідентифікатором 1 у таблиці "Поставок". Він змінює значення кількості відвантажених одиниць на 120 для відвантаження з ідентифікатором 1

UPDATE Shipments SET QuantityShipped = 120 WHERE ShipmentID = 1;

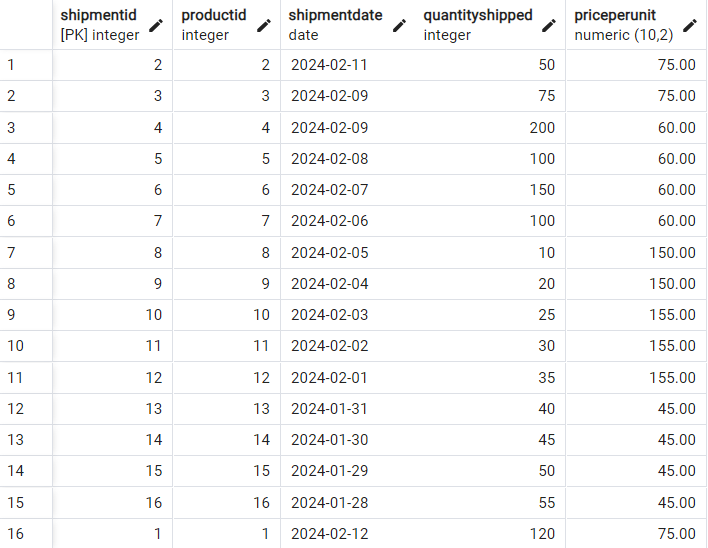


Рис 3.4.24 – Таблиця Операції UPDATE

1. Цей блок SQL-запитів вирішує задачу видалення запису з таблиці " Поставок ", де ідентифікатор відвантаження (стовпець "ShipmentID") дорівнює 16.

DELETE FROM Shipments WHERE ShipmentID = 16;

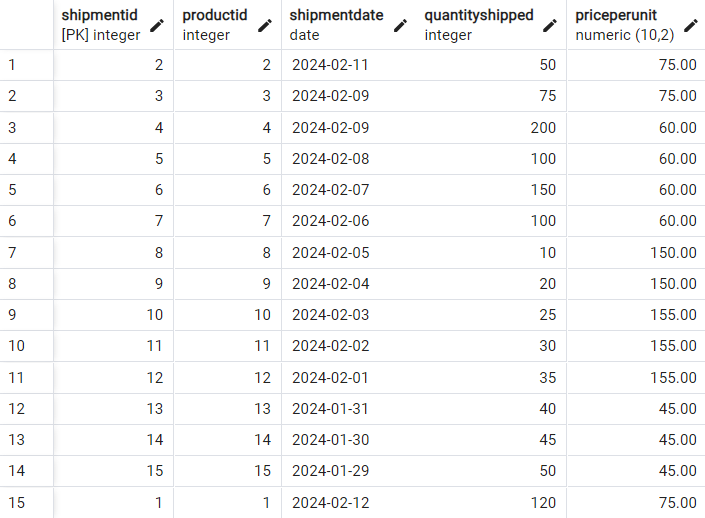


Рис 3.4.25 – Таблиця Операції DELETE

1. Цей блок SQL-запитів вирішує задачу обчислення середньої кількості відвантажених одиниць у всіх записах таблиці " Поставок ".

SELECT AVG(QuantityShipped) FROM Shipments;

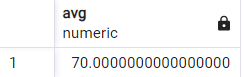


Рис 3.4.26 – Таблиця Операції AVG

ВИСНОВКИ

У ході виконання практики та індивідуального завдання було проведено дослідження ІТ інфраструктури та комп'ютеризації компанії "KeepSolid", що сприяло глибокому розумінню її організаційних, функціональних та структурних характеристик. Окрім того, була виконана практична частина завдань від компанії, де проводився аналіз ринкових потреб та розробка програм на основі принципу JTBD.

Щодо індивідуального завдання, був проведений огляд підходів та критеріїв вибору систем управління базами даних (СУБД) для створення інформаційних систем. Також була розроблена база даних для Квіткового магазину.

В результаті цієї роботи було підтверджено важливість глибокого розуміння інфраструктури підприємства для ефективного розв'язання його завдань у сфері інформаційних технологій. Робота дала змогу розвинути професійні навички та отримати практичний досвід у роботі з ІТ та інформаційними системами.

ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Реляційна база даних – UA5.org. *UA5.org – Матеріали з інформаційних технологій*. URL: <https://ua5.org/database/189-reljacjjna-baza-danikh.html> (дата звернення: 22.02.2024).
2. Як організувати дані у додатку: NoSQL-підходи у реляційних базах даних. *Highload.today - медіа для розробників*. URL: <https://highload.today/uk/blogs/yak-organizuvati-dani-u-dodatku-nosql-pidhodi-u-relyatsijnih-bazah-danih/#9>(дата звернення: 22.02.2024).
3. SQL чи NoSQL - ось в чому питання - Альтернативна Наука Віталія Сема. *Альтернативна Наука Віталія Сема*. URL: <http://surl.li/qummy> (дата звернення: 22.02.2024).
4. Реляційні бази даних: структура та застосування у практиці. *FoxmindEd*. URL: <https://foxminded.ua/reliatsiini-bazy-danykh/> (дата звернення: 22.02.2024).
5. 3.4. Критерії вибору субд при створенні інформаційних систем. *StudFiles*. URL: <https://studfile.net/preview/5118185/page:29/> (дата звернення: 22.02.2024).