

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
(повна назва інституту/факультету)
КАФЕДРА інформатики та програмної інженерії
(повна назва кафедри)

КУРСОВА РОБОТА
з дисципліни “Бази даних”
(назва дисципліни)

на тему: База даних служби таксі

Студентки 2 курсу ІІІ-22 групи
спеціальності 121 «Інженерія програмного забезпечення»
Семенової Є. О.
(прізвище та ініціали)

Керівник ст. вик. Марченко О. І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка ECTS _____

Члени комісії

_____	_____
<small>(підпис)</small>	<small>(вчене звання, науковий ступінь, прізвище та ініціали)</small>
_____	_____
<small>(підпис)</small>	<small>(вчене звання, науковий ступінь, прізвище та ініціали)</small>
_____	_____
<small>(підпис)</small>	<small>(вчене звання, науковий ступінь, прізвище та ініціали)</small>

Київ – 2023 рік

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Інформатики та програмної інженерії
(повна назва)

Дисципліна Бази даних

Курс 2 Група ІІ-22 Семестр 3

**З А В Д А Н Н Я
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Семеновій Єлизаветі Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи База даних служби таксі

керівник роботи ст. вик. Марченко Олена Іванівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 29.12.2023

3. Вихідні дані до роботи Потреба провести аналіз та розробити зрозумілу та якісну модель та надійну базу даних для служби таксі

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у форматі обраної системи управління базою даних

5) Створення користувачів бази даних

6) Імпорт даних з використанням засобів СУБД в створену базу даних

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 08.11.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	12.11.2023	
2	Побудова ER-моделі	01.12.2023	
3	Побудова реляційної схеми з ER-моделі	10.12.2023	
4	Створення бази даних, у форматі обраної системи управління базою даних	14.12.2023	
5	Створення користувачів бази даних	14.12.2023	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	18.12.2023	
7	Створення мовою SQL запитів	20.12.2023	
8	Оптимізація роботи запитів	22.12.2023	
9	Оформлення пояснювальної записки	28.12.2023	
10	Захист курсової роботи	30.12.2023	

Студент

(підпис)

Семенова Є. О.

(прізвище та ініціали)

Керівник роботи

(підпис)

Марченко О. І.

(прізвище та ініціали)

ЗМІСТ

ВСТУП.....	5
1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА.....	6
2 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ.....	11
3 ПОСТАНОВКА ЗАВДАННЯ.....	15
4 ПОБУДОВА ER-МОДЕЛІ.....	16
4.1 ER-модель.....	16
4.2 Даталогічна модель.....	22
5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ.....	28
5.1 Створення таблиць та їх заповнення.....	28
5.2 Створення ролей і користувачів.....	37
6 РОБОТА З БАЗОЮ ДАНИХ.....	39
6.1 Тексти генераторів.....	39
6.2 Тексти збережених процедур/функцій.....	40
6.3 Тексти тригерів.....	49
6.4 Тексти представлень.....	58
6.5 SQL-запити.....	61
6.6 Результати оптимізації.....	80
ВИСНОВКИ.....	82
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	83
ДОДАТКИ.....	84

ВСТУП

У будь-якому сучасному бізнесі необхідне використання інформаційних технологій. Служба таксі щодня має справу з великим об'ємом даних. База даних допоможе як керівникам, працівникам, так і клієнтам служби ефективно та безпечно зберігати, додавати та переглядати інформацію про роботу служби.

Робота з базою даних забезпечує швидкість та зручність роботи сервісу - надзвичайно важливі критерії у сфері обслуговування. За допомогою бази даних користувачі сервісу зможуть швидко і коректно подавати замовлення, уникаючи невизначеностей та непорозумінь, а також залишати відгуки щодо роботи сервісу. Працівники служби зможуть відстежувати нові замовлення та фіксувати інформацію про виконані поїздки. Керівники компанії зможуть відстежувати списки персоналу та транспорту, а також робити висновки щодо ефективності роботи служби та щодо задоволеності потреб користувачів.

Метою даної курсової роботи є розробка та реалізація бази даних для служби таксі, спрямованої на оптимізацію управління та підвищення якості обслуговування клієнтів. Основними завданнями роботи є проектування структури бази даних, її реалізація із забезпеченням збереження цілісності даних та реалізацією актуальних запитів.

Для реалізації бази даних служби таксі обрано Систему Управління Базами Даних PostgreSQL. Перевагами PostgreSQL є надійність, масштабованість, можливість використання Pearl і Python, а за встановлення розширень ще Java, JavaScript (V8), R та Rust, для написання функцій. Крім того, PostgreSQL підтримує географічні дані завдяки розширенню PostGIS. PostGIS дозволяє ефективно зберігати, опрацьовувати та аналізувати географічні дані, а також має багато корисних функцій, як от розрахунок відстані між точками. Крім цього PostGIS має зручну візуалізацію для географічних точок у базі даних, що дозволяє орієнтуватись у даних краще. Вибір PostgreSQL із розширенням PostGIS відповідає вимогам модернізованої та технологічно орієнтованої служби таксі, забезпечуючи не лише надійність та швидкість, але й можливість ефективно працювати з географічними аспектами.

1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Об'єктом дослідження є служба таксі, яка є ключовим елементом в галузі перевезень пасажирів. У контексті бази даних служби таксі, об'єктом є взаємодія між клієнтами, водіями та системою управління для забезпечення ефективного та швидкого обслуговування.

Система передбачатиме створення акаунтів користувачів, які включатимуть у себе інформацію про домашню адресу та номер телефону клієнта, а також зберігає логін та пароль у системі. Кожен логін є унікальним. Пароль повинен мати довжину в межах від 6 до 30 символів.

Клієнт:

- логін;
- пароль;
- домашня адреса;
- номер телефону.

Кожен клієнт може створювати необмежену кількість замовлень. Замовлення включають в себе інформацію про дату та час створення, місце відправлення та призначення, обраний тариф, обраний тип оплати, а також може включати додаткові послуги та знижки. Час замовлення за замовчуванням встановлюється на поточний, але клієнт може змінити його за бажанням. Координати точок початку й кінця поїздки передаються з графічного інтерфейсу, де клієнт може встановлювати їх за допомогою введення у пошукову строку або вибір на карті. Тариф та додаткові послуги обираються із заздалегідь визначених компанією списків. Знижка задається системою автоматично за виконання умов.

Замовлення:

- клієнт;
- тариф;
- знижка;
- час замовлення;
- місце відправлення;
- місце призначення;
- спосіб оплати.

Компанія має список визначених тарифів, які призначаються транспортним засобам в залежності від їх характеристик. У службі таксі є 6 тарифів: Економ, Стандарт, Комфорт, Бізнес, Універсал, Мінібус.

Тариф:

- назва тарифу;
- вартість тарифу (грн. за км).

Кількість додаткових послуг необмежена; в межах одного замовлення може надана одна знижка. Знижка в розмірі 5% надається на кожне 10-те замовлення. Знижка в розмірі 15% для постійних клієнтів (клієнтів, що протягом останніх трьох місяців скористались послугами служби менше 15 разів). Якщо замовлення створене в день святкування Нового року, то замовленню надається спеціальна святкова знижка в розмірі 24%.

Знижка:

- назва знижки;
- розмір знижки.

Додаткова послуга обирається з таблиці послуг, що надаються службою. Кожна послуга має заздалегідь визначену ціну, яка не залежить від тарифу поїздки.

Типові послуги:

- назва послуги;
- вартість послуги.

Представники компанії служби таксі можуть додавати нових водіїв після успішної співбесіди. У систему вноситься така інформація про кожного водія: прізвище, ім'я та номер посвідчення водія. Після того, як водій виконав кілька поїздок і отримав відгуки, система розраховує рейтинг водія. Рейтинг розраховується як середнє значення усіх оцінок, що були надані водію. Рейтинг постійно оновлюється з кожним створенням та видаленням відгуків на поїздки, виконані водієм. Рейтинг представляється у вигляді числа з одним значенням після коми.

Водій:

- прізвище;
- ім'я;
- посвідчення водія;
- номер телефону;
- рейтинг.

Також компанія веде облік транспорту, яким вона володіє. У системі фіксується модель, номерний знак та колір транспортного засобу, а також тариф, якому він відповідає. При цьому кожен номерний знак є унікальним.

Транспортний засіб:

- модель;
- номерний знак;
- колір.

Водій компанії може переглядати нові замовлення та обирати серед них ті, що співпадають за тарифами з транспортним засобом, який він наразі використовує. Також може визначати за допомогою інтерфейсу яке з цих замовлень розташоване ближче до його поточного місцезнаходження.

Оскільки у компанії немає встановленої відповідності водій-транспортний засіб (різні водії можуть використовувати один і той самий транспортний засіб у різний час; один водій може обирати різні транспортні засоби в залежності від поточних замовлень), то у системі немає зв'язку між водієм та транспортним засобом. Таким чином забезпечується взаємозамінність як водіїв так і транспорту, а отже і гнучкість роботи служби.

Після початку обробки замовлення, водій може створити на основі цього замовлення запис про поїздку. У записі зберігається інформація про дату та час початку перевезення, відстань та вартість поїздки, а також транспортний засіб (тариф транспортного засобу має відповідати тарифу, обраному в замовленні). Вартість поїздки вираховується автоматично за такою формулою:

$$\text{вартість} = (\text{відстань} \cdot \text{тариф} + [\text{сума вартостей додаткових послуг}]) \cdot (1 - \text{розмір знижки})$$

Поїздка:

- замовлення;
- водій;
- транспортний засіб;
- час поїздки;
- відстань;
- вартість.

Клієнт може залишити відгук до поїздки. У відгуці користувач обов'язково надає оцінку в межах від 1 до 5, а також за бажанням може залишити коментар. Користувач може додавати декілька відгуків до однієї поїздки.

Відгук:

- поїздка;
- оцінка;
- коментар.

Використання даних з цієї бази даних може включати, та не обмежується:

1. Аналіз якості роботи водіїв за їх оцінкою користувачами послуг служби таксі, кількістю та довжиною проведених ними поїздок, для визначення найефективніших працівників компанії, присвоєння премій водіям, що виконали найбільше замовлень;
2. Розгляд працівників, з якими розривається контракт. Контракт розривається якщо водій не виконує замовлення більше 3 тижнів без попередження керівництва та має рейтинг менше 4 балів;
3. Автоматизація процесу обчислення заробітної плати для водіїв. Заробітна плата становить 20% від суми вартостей виконаних водієм поїздок. Також у значенні враховується штраф, отриманий водієм за місяць. Штраф складає 2 грн. за кожну хвилину очікування клієнта після перших 10 хвилин;
4. Збереження даних водіїв, що отримували високий рейтинг навіть при звільненні, для використання у аналізі та можливій подальшій співпраці;
5. Аналіз прибутковості за різні періоди. Визначення більш прибуткових місяців та днів тижня. Обрахунок виручки компанії за рік;
6. Визначення популярних тарифів та послуг для використання у маркетингу та наданні знижок;
7. Аналіз стану автопарку. Визначення найбільш використаних транспортних засобів для першочергового техогляду, перелік нових автомобілів для використання при замовленні святкувань;
8. Аналіз уподобань користувачів та тенденцій у замовленнях, таких як: середня вартість поїздки користувача, відстань між точкою початку поїздки та домашньою адресою, кількість клієнтів, що замовляють поїздки пізніше 23:00.

Система передбачає багато користувачів з різними дозволами. Клієнт служби таксі може редагувати дані у власному акаунті, створювати замовлення та додавати до нього послуги, додавати відгуки на поїздки, виконані за його замовленням, а також переглядати будь-яку інформацію, що потребується для створення замовлення, історію

поїздок, транспорт та водія, що його обслуговує. Водій може переглядати деталі нових замовлень, відгуки на виконані ним поїздки, а також про транспорт наявний у автопарку компанії. Водій може додавати записи про виконані ним поїздки поїздки. Адміністратор інформаційної системи служби таксі може переглядати інформацію з усіх таблиць, редагувати та додавати записи про транспортні засоби, водіїв, знижки, тарифи, типові послуги.

Модель користувачів бази даних:

1. Клієнт

Права:

- редагувати дані клієнта;
- створювати замовлення;
- додавати послуги до замовлень, відгуки;
- переглядати інформацію щодо типів послуг, тарифів, знижок, поїздок, транспортного засобу та водіїв.

2. Водій:

Права:

- створювати поїздки;
- переглядати інформацію щодо послуг, тарифів, поїздок, транспортних засобів, відгуків, замовлень.

3. Адміністратор системи:

Права:

- додавати транспортні засоби;
- додавати та змінювати записи про водіїв, знижки, послуг, тарифи;
- переглядати інформацію про водіїв, транспортні засоби, клієнтів, замовлення та поїздки, відгуки, а також послуги, тарифи, знижки.

Для збереження цілісності даних вводимо обмеження на обов'язкове заповнення полів, а також такі окремі обмеження:

1. Унікальний логін клієнта;
2. Довжина паролю клієнта від 6 до 30 символів;
3. Номери телефонів клієнтів та водіїв складаються з 13 символів, де перші 4 відповідають шаблону '+380';
4. Розмір знижки подається як десятковий дріб від 0 до 1;
5. До одного замовлення створюється одна поїздка;
6. Оцінка у відгуку є цілим числом від 1 до 5.

2 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ

Ринок програмних застосунків для сервісів таксі не є новим та був в Україні ще з 2010 року [1]. Обираємо додатки на основі кількості населених пунктів, які вони покривали станом на початок 2022 року [2], а також на основі їх рейтингу у Google store. Для аналізу було відібрано додатки служби таксі Uklon та OnTaxi, що займають провідні позиції у рейтингу безкоштовних додатків у категорії “Подорожі та місцева інформація”, а також є розповсюдженими на всій території України.

Uklon – перший український онлайн-сервіс замовлення авто, створений у 2010 році [3]. Його основними перевагами є можливість редагувати активне замовлення з автоматичним перерахунком вартості, оплата через Apple Pay та Google Pay, можливість додавати нові адреси, яких немає на карті Uklon, а також постійні оновлення як додатку так і доступних послуг.

При реєстрації у систему заноситься номер телефону користувача. У особистому профілі користувач може за бажанням заповнити такі особисті дані: ім'я, дата народження, стать, електронна адреса, а також за потреби змінити номер телефону. Також клієнт має можливість персоналізувати подальші замовлення за допомогою опитування. У опитуванні можна вказати чи клієнт має дітей, що потребують дитячого крісла, домашнього улюбленця або власне авто. Користувач може додавати обрані адреси, серед по одній адресі для категорій Дім та Робота, а також необмежена кількість обраних адрес, в яких користувач самостійно задає назву. Після поїздки користувач може додати водія у чорний список, який відображається в особистому профілі.

При створенні замовлення задається адреса початкової та кінцевої точок, які можна задати у стрічці пошуку, як точку на карті або як одну зі збережених адрес. Після цього потрібно обрати тариф з такого переліку: стандарт, комфорт, prime, експрес, універсал, бізнес, інклюзивний, green, мікроавтобус, та автомобіль. Спосіб обирається між готівкою, картою або Google Pay. Також можливо замовити додаткові послуги з переліку: кондиціонер, поїздка з твариною, додатковий багаж, англomовний водій, водій, що не палить, тиша в салоні, зустріч з табличкою. За замовчанням замовлення створюється на поточний час, але також є можливість задати запланований час або вказати інтервал очікування. Також з початком війни у замовленні є можливість додати донат для різних зборів для ЗСУ, зокрема у Благодійний фонд Сергія Притули.

Крім звичайного перевезення Uklon пропонує такі послуги: доставка, драйвер та міжміст. Доставка може здійснюватись як в межах міста, так і в інші міста з обмеженням у вазі до 20 кг, а також додатковою послугою доставки від дверей до дверей. Драйвер - це послуга за якої водій відвозить клієнта на замовлену адресу на авто клієнта. Міжміст - це міжміське перевезення клієнтів.

Клієнт може переглядати історію своїх поїздок, та передивитись скільки з них було виконано, а скільки скасовано. Окрім того значною перевагою Uklon є можливість редагувати запропоновану ціну.

Окрім додатка для клієнтів Uklon також пропонує програмне забезпечення для водіїв. Система допомагає користувачам заробляти більше та має багато функцій, що полегшують життя працівників таксі.

У додатку є рейтинг водіїв - "Карма". Рейтинг залежить від багатьох параметрів, зокрема від оцінок клієнтів та кількості скасованих замовлень. Рейтинг працює як критерій відбору водія для замовлення у системі. Чим більший рейтинг, тим швидше працівник отримує замовлення. Така функція спонукає водіїв виконувати поїздки якісно та регулярно.

Для максимізації вигоди водіїв було розроблено кілька функцій. Так у "непопулярні" години працюються бонусні програми. Таким чином працюючим водіям компенсується зменшена через нестачу клієнтів оплата. Також в Uklon тариф прив'язаний до точки висадки: якщо кінцева точка є популярною, то тариф є меншим, адже більше вірогідність, що водій отримає наступне замовлення неподалік від висадки попереднього клієнта.

Uklon для водіїв автоматизує процеси, які раніше працівники виконували самостійно. Система розраховує вартість простоювання. Відлік починається після 3 хвилин очікування та програма включає вартість очікування у суму, що сплачує клієнт. Також водій отримує автоматичні пропозиції наступного замовлення, стартова точка якого буде неподалік точки висадки попереднього клієнта.

OnTaxi заснований 2016 року в Харкові. Сьогодні сервіс присутній у 28 містах України за прямою моделлю, 32 містах – за моделлю франшизи, а також у Азербайджані та країнах ЄС. Як повідомив Forbes CEO компанії Володимир Кузьменко, наразі сервісом співпрацюють понад 300 000 водіїв [4].

При реєстрації клієнт теж повинен вказати номер телефону, але також і ім'я. В особистому профілі клієнт може змінити ім'я, номер телефону, а також прив'язати акаунт Google або Facebook. Також можна вказати дату народження та стать і

відповісти на питання для персоналізації замовлень. Також можна вказати свої уподобання в поїздки та зазначити чи вам потрібен водій без музики або без спілкування. Клієнт може зберігати адреси дому, роботи, а також інших точок без обов'язкового вказання назви для збережених адрес.

При замовленні клієнт обирає точку початку й кінця поїздки за допомогою пошуку, точки на карті або вибору зі списку збережених адрес. Клас авто (тариф) обирається з запропонованого списку: економ, стандарт, комфорт, бізнес, універсал, мінівен, а також доставка для посилок не більше 20 кг вагою. У інтерфейсі детально описано що входить у тариф та як вираховується вартість поїздки. Замовити таксі можна як для себе, так і для іншої людини за її номером телефону. В замовлення можна включити такі додаткові послуги: дитяче крісло, кондиціонер, тверезий водій (водій для вашого авто), перевезення тварини, англомовний водій, підзарядка, та вибір електромобілю. Оплатити можна готівкою, картою або за допомогою Google Pay або Apple Pay.

В OnTaxi можна створювати шаблони поїздок, що можуть знадобитись, якщо користувач часто замовляє поїздки з однаковим набором послуг та однаковим маршрутом.

OnTaxi також пропонує долучатись до благодійності та має можливості як одноразового так і регулярного пожертвування.

В застосунку OnTaxi клієнт може редагувати маршрут активного замовлення, але потребує підтвердження від водія [5]. Завершені замовлення зі зміною маршруту перераховуються через службу підтримки. Про зміну маршруту необхідно повідомити оператора служби підтримки, який перерахує вартість замовлення. Дзвонити потрібно із розділу “Служба підтримки” у застосунку.

Платне очікування клієнта вмикається вручну кнопкою “простій”.

У системі OnTaxi рейтинг надається як водіям, так і клієнтам. Відгуки допомагають відстежити причини конфліктів, заблокувати клієнтів, що погіршують умови роботи водіїв, а також блокувати водіїв, що виконують свою роботу неякісно.

За скасування замовлення у цій системі водій отримує штраф, але це не впливає на його рейтинг.

Отже, розглянувши два популярних програмних продукти, можна зробити висновок, що їх суть схожа, а відмінності проявляються лише в деталях. Так в Uklon більшість процесів автоматизовані, в той час як в OnTaxi роблять ставку на персоналізацію як для клієнтів так і водіїв. У більш загальних аспектах робота систем

не відрізняється. У профілі клієнта зберігаються ідентичні дані, замовлення створюються за схожим принципом. Робота водіїв має відмінності. Працівники, що користуються інформаційними сервісами Uklon більше піклуються про свій рейтинг, адже від нього наряду залежить їх заробіток. Система робить за них більшість роботи щодо розрахунку вартості та вибору наступного замовлення, а також робить їх роботу більш ефективною і вигідною. Водії, що користуються OnTaxi повинні самостійно відслідковувати набагато більше процесів, але також мають перевагу - можливість дізнатись рейтинг клієнта перед поїздкою та відмовитись від перевезення потенційно конфліктного користувача.

Розглянуті програмні продукти Uklon та OnTaxi демонструють схожість у базових принципах роботи. Ці системи можуть слугувати прикладами під час розробки бази даних у цій роботі.

3 ПОСТАНОВКА ЗАВДАННЯ

Розглядаючи розділи “Опис предметного середовища” та “Аналіз існуючих програмних продуктів”, можна визначити основні завдання курсової роботи.

У роботі буде створено 11 таблиць, що забезпечуватимуть збереження необхідних даних, валідність зв'язків між іншими таблицями та винесення даних, що повторюються.

Першим кроком буде розробка ER-моделі на основі аналізу предметного середовища. Перед початком моделювання буде проведено структурування та детальний опис атрибутів та зв'язків між сутностями. Після створення ER-моделі, на її основі буде створено даталогічну модель для полегшення переходу від абстрактного уявлення про задану систему до створення бази даних.

З використанням результатів моделювання ER-моделі та даталогічної моделі створюємо фізичну базу даних за допомогою СУБД PostgreSQL. Після створення таблиць, враховуємо всі обмеження та зв'язки, що були визначені раніше у розділі “Опис предметного середовища” та даталогічних таблицях. Заповнюємо таблиці даними за допомогою команди INSERT, вставки з файлів CSV та генерації випадкових даних.

За описом моделі користувачів бази даних створюємо ролі та користувачів у базі даних таксі та присвоюємо їм відповідні права.

Для зручного заповнення таблиць та забезпечення унікальності первинних ключів застосовуємо генератори послідовності ключів.

Формулюємо практичні та логічні запити, формули та представлення, після чого реалізуємо їх та тестуємо.

Створюємо тригери для автоматизації розрахунків аргументів таблиць, перевірок валідності даних та попередження втраті інформації.

Оптимізуємо роботу запитів з використанням індексів та проводимо дослідження для визначення необхідних індексів.

4 ПОБУДОВА ER-МОДЕЛІ

4.1 ER-модель

Проаналізувавши предметне середовище та бізнес-правила, можна визначити такі основні сутності та їх атрибути у системі служби таксі.

Клієнт - інформація, що зберігається у профілі клієнта служби таксі.

Замовлення - замовлення на поїздку.

Тип послуги - типові додаткові послуги компанії.

Послуга - список послуг для конкретного замовлення, зв'язок між типом послуги та замовленням.

Тариф - ціна за кілометр для певної категорії транспорту

Знижка - знижки, що надає компанія

Модель - модель транспортних засобів, що є в автопарку компанії

Транспортний засіб - транспортний засіб, що наявний у автопарку компанії

Водій - водій, що працює у службі таксі.

Поїздка - поїздка, що була виконана за замовленням.

Відгук - відгук на поїздку.

У Таблицях 4.1-4.11 наведено опис атрибутів сутностей системи служби таксі.

Таблиця 4.1

Клієнт	
Атрибут	Опис
логін	унікальне ім'я користувача у системі
пароль	пароль для захисту акаунту клієнта
адреса	місцезнаходження домашньої адреси клієнта
номер телефону	номер телефону клієнта, що використовується для реєстрації та зв'язку при замовленнях

Таблиця 4.2

Замовлення	
Атрибут	Опис
час замовлення	час, на який потрібна поїздка; за замовчуванням - поточний час
місце відправки	точка місцезнаходження початку маршруту
місце призначення	точка місцезнаходження кінця маршруту
тариф	обраний клієнтом тариф поїздки
знижка	знижка, що призначається системою автоматично при виконанні певних умов
тип оплати	оплата карткою чи готівкою
клієнт	клієнт, що створив замовлення

Таблиця 4.3

Послуга	
Атрибут	Опис
замовлення	замовлення, в межах якого надається послуга певного типу
тип послуги	тип послуги, яка надається у замовленні

Таблиця 4.4

Тип послуги	
Атрибут	Опис
назва послуги	назва або коротка характеристика послуги
ціна послуги	стала ціна послуги

Таблиця 4.5

Знижка	
Атрибут	Опис
назва знижки	назва або короткий опис знижки
розмір знижки	розмір знижки у вигляді десяткового дробу

Таблиця 4.6

Тариф	
Атрибут	Опис
назва тарифу	назва або короткий опис тарифу
вартість	вартість перевезення за тарифом за 1 км

Таблиця 4.7

Модель	
Атрибут	Опис
марка	назва марки
назва моделі	повна назва моделі
рік випуску	рік випуску моделі

Таблиця 4.8

Транспортний засіб	
Атрибут	Опис
модель	посилання на модель транспорту
тариф	посилання на тариф, що відповідає транспортному засобу

Продовження Таблиці 4.8

номерний знак	номерний знак транспортного засобу
колір	колір транспортного засобу

Таблиця 4.9

Водій	
Атрибут	Опис
прізвище	прізвище водія
ім'я	ім'я водія
посвідчення водія	серійний номер посвідчення подія
номер телефону	номер телефону водія для комунікації під час роботи
рейтинг	рейтинг водія, що розраховується як середнє значення наданих водію оцінок

Таблиця 4.10

Поїздка	
Атрибут	Опис
замовлення	посилання на замовлення, на основі якого створюється запис про поїздку
транспортний засіб	посилання на транспортний засіб, яким було здійснено поїздку
водій	посилання на водія, що виконав поїздку
час поїздки	дата та час початку поїздки; за замовчуванням поточний час

Продовження таблиці 4.10

відстань	відстань, на яку було здійснено поїздки розраховується як відстань між початковою та кінцевою точками місцезнаходження
вартість	вартість поїздки розраховується як сума вартості пройденої відстані за обраним тарифом і загальної суми усіх додаткових послуг, помноженої на відсоток урахуванням знижки

Таблиця 4.11

Відгук	
Атрибут	Опис
поїздка	посилання на поїздку, на яку залишають відгук
оцінка	оцінка поїздки в межах від 1 до 5
коментар	розлогий коментар, щодо поїздки

Після опису сутностей переходимо до моделювання зв'язками. Згідно з описом предметного середовища та бізнес-правил, зазначених раніше у роботі, складаємо Таблицю 4.12 та детально описуємо зв'язки між сутностями.

Таблиця 4.12

Первинна сутність	Вторинна сутність	Зв'язок
Клієнт	Замовлення	1:М Клієнт може робити багато замовлень. Замовлення робиться одним клієнтом.
Замовлення	Послуга	1:М В одне замовлення може входити багато послуг.

Продовження Таблиці 4.12

Первинна сутність	Вторинна сутність	Зв'язок
Послуга	Тип послуги	M:1 Може надаватись багато послуг одного типу.
Замовлення	Знижка	M:1 В замовлення може бути включена лише одна знижка. Однакова знижка може надаватись за багатьом замовленням.
Замовлення	Тариф	M:1 Замовлення створюється за одним тарифом. За тарифом може бути створено багато замовлень.
Замовлення	Поїздка	1:1 За одним замовленням створюється одна поїздка.
Поїздка	Транспортний засіб	M:1 Поїздка проводиться на одному транспортному засобі. Один транспортний засіб проводить багато поїздок.
Поїздка	Водій	M:1 Поїздка виконується одним водієм. Один водій виконує багато поїздок.
Поїздка	Відгук	1:M Відгук надається за однією поїздкою. За поїздкою може бути надано багато відгуків.
Транспортний засіб	Тариф	M:1 Транспортний засіб оцінюється за одним тарифом. Тариф може відповідати багатьом транспортним засобам.
Транспортний засіб	Модель	M:1 Транспортний засіб має лише одну модель. Може існувати багато транспортних засобів однієї моделі.

На основі опису сутностей, їх атрибутів та зв'язків між сутностями проектуємо ER-модель системи (Рис. 4.1).

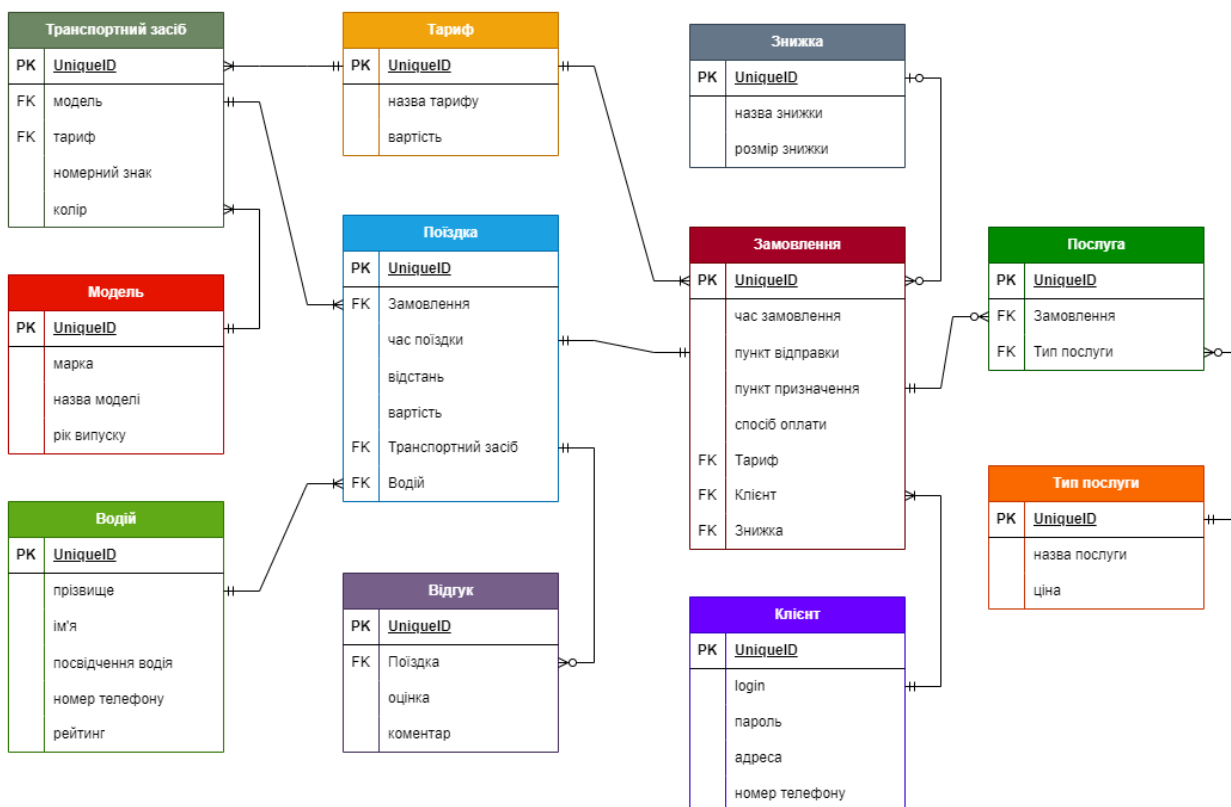


Рис. 4.1. ER-модель системи служби таксі

4.2 Даталогічна модель

На основі описів сутностей та їх зв'язків створюємо таблиці з детальним описом типів і обмежень атрибутів (Таблиці 4.13-4.23).

Таблиця 4.13 Таблиця сутності Клієнт

Client				
Атрибут		Ключ	Тип	Обмеження
—	client_id	PK	INTEGER	
логін	client_login		VARCHAR(20)	UNIQUE NOT NULL
пароль	client_password		VARCHAR(30)	NOT NULL 6<=length

Продовження таблиці 4.13

адреса	client_address		GEOGRAPHY	
номер телефону	client_pnumber		CHAR(13)	NOT NULL

Таблиця 4.14 Таблиця сутності Водій

Driver				
Атрибут		Ключ	Тип	Обмеження
—	driver_id	PK	INTEGER	
прізвище	last_name		VARCHAR(20)	NOT NULL
ім'я	first_name		VARCHAR(20)	
посвідчення водія	driving_license		VARCHAR(10)	NOT NULL
номер телефону	driver_pnumber		CHAR(13)	NOT NULL
рейтинг	driver_rating		NUMERIC(2, 1)	

Таблиця 4.15 Таблиця сутності Знижка

Discount				
Атрибут		Ключ	Тип	Обмеження
—	discount_id	PK	INTEGER	
назва знижки	discount_name		VARCHAR(20)	NOT NULL
розмір знижки	discount_size		NUMERIC(3, 2)	NOT NULL 0<=discount_size<=1

Таблиця 4.16 Таблиця сутності Тип послуг

Service_type				
Атрибут		Ключ	Тип	Обмеження
—	st_id	PK	INTEGER	
назва послуги	service_name		VARCHAR(20)	NOT NULL
ціна послуги	price		NUMERIC(10, 2)	NOT NULL

Таблиця 4.17 Таблиця сутності Тариф

Tariff				
Атрибут		Ключ	Тип	Обмеження
—	tariff_id	PK	INTEGER	
назва тарифу	tariff_name		VARCHAR(20)	NOT NULL
вартість	price		NUMERIC(10, 2)	NOT NULL

Таблиця 4.18 Таблиця сутності Замовлення

Orders				
Атрибут		Ключ	Тип	Обмеження
—	order_id	PK	INTEGER	
час замовлення	order_time		TIMESTAMP	DEFAULT NOW
місце відправки	departure_point		GEOGRAPHY	NOT NULL

Продовження таблиці 4.18

місце призначення	destination_point		GEOGRAPHY	NOT NULL
тип оплати	payment		ENUM	NOT NULL
тариф	tariff_id	FK	INTEGER	NOT NULL
клієнт	client_id	FK	INTEGER	NOT NULL
знижка	discount_id	FK	INTEGER	

Таблиця 4.19 Таблиця сутності Послуга

Service				
Атрибут		Ключ	Тип	Обмеження
—	service_id	PK	INTEGER	
тип послуги	st_id	FK	INTEGER	NOT NULL
замовлення	order_id	FK	INTEGER	NOT NULL

Таблиця 4.20 Таблиця сутності Модель

Model				
Атрибут		Ключ	Тип	Обмеження
—	model_id	PK	INTEGER	
марка	brand		VARCHAR(20)	NOT NULL
назва моделі	model_name		VARCHAR(30)	NOT NULL
рік випуску	manufacturing year		INTEGER	

Таблиця 4.21 Таблиця сутності Транспортний засіб

Transport				
Атрибут		Ключ	Тип	Обмеження
—	transport_id	PK	INTEGER	
модель	model_id	FK	INTEGER	NOT NULL
тариф	tariff_id	FK	INTEGER	NOT NULL
номерний знак	plate_number		VARCHAR(8)	UNIQUE NOT NULL 3<=length
колір	color		VARCHAR(10)	

Таблиця 4.22 Таблиця сутності Поїздка

Ride				
Атрибут		Ключ	Тип	Обмеження
—	ride_id	PK	INTEGER	
замовлення	order_id	FK	INTEGER	UNIQUE NOT NULL
водій	driver_id	FK	INTEGER	NOT NULL
час поїздки	ride_time		TIMESTAMP	DEFAULT NOW
відстань	distance		NUMERIC(10, 2)	
вартість	total_price		NUMERIC(10, 2)	
транспортний засіб	transport_id	FK	INTEGER	NOT NULL

Таблиця 4.23 Таблиця сутності Відгук

Feedback				
Атрибут		Ключ	Тип	Обмеження
—	feedback_id	PK	INTEGER	
поїздка	ride_id	FK	INTEGER	NOT NULL
оцінка	rating		INTEGER	NOT NULL $1 \leq \text{rating} \leq 5$
коментар	commentary		VARCHAR(8)	TEXT

5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

5.1 Створення таблиць та їх заповнення

На основі розробленої моделі створюємо фізичну реалізацію бази даних у СУБД PostgreSQL. Створюємо базу даних Taxi за допомогою наступного DDL скрипта:

```
CREATE DATABASE Taxi;
```

Перед створенням таблиць було визначено типи PHONE_NUMBER для запису номеру телефону, та PAYMENT_METHOD для вибору способу оплати, що будуть застосовуватись у атрибутах, а також під'єднано додаткове розширення PostGIS для роботи з геоданими.

```
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;
```

Створений тип даних PHONE_NUMBER - строка зі сталою кількістю символів - 13, а також обмеженням: перші 4 символи відповідають значенню '+380'. Тобто номер телефону задається за таким шаблоном: '+380502103565'

```
CREATE DOMAIN PHONE_NUMBER AS CHAR(13)
CHECK (VALUE LIKE '+380%');
```

Створений тип даних - PAYMENT_METHOD є переліком методів оплати, де можливі варіанти 'cash' - готівка та 'card' - картка.

```
CREATE TYPE PAYMENT_METHOD AS ENUM ('cash', 'card');
```

Створення таблиці Client за описом у таблиці 4.13

```
CREATE TABLE Client(
client_id SERIAL PRIMARY KEY,
client_login VARCHAR(20) UNIQUE NOT NULL,
client_password VARCHAR(30) NOT NULL CHECK(length(client_password)
>=6),
client_address GEOGRAPHY(POINT),
client_pnumber PHONE_NUMBER NOT NULL
);
```

Перевірка: `SELECT * FROM Client;`

client_id [PK] integer	client_login character varying (20)	client_password character varying (30)	client_pnumber character	client_address geography
1	aritelli0	hO6{?R{Y~	+380653548402	0101000020E6100000789CA223B9DC3E40EEBC039232A4940
2	clesley1	al0+Pd	+380143008229	0101000020E6100000A245B6F3FD543E40F38E537424474940
3	kmustoo2	aG8==,)2	+380063223237	0101000020E61000005F984C158C8A3E401EA7E8482E2F4940
4	amcdermot3	yB2'9B3	+380318980375	0101000020E61000006E3480B740C23E400B24287E8C494940
5	clillee4	tT9_L'Dt&	+380128515473	0101000020E6100000E7FBA9F1D28D3E409A779CA223394940
6	ostearn5	wL8`T/h!	+380347690124	0101000020E610000023DBF97E6A6C3E40736891ED7C474940
7	amarte6	wF4+e<	+380701834866	0101000020E61000009BE61DA7E8C83E400A68226C783A4940
8	jsimonin7	rA2/\<C}	+380428573769	0101000020E6100000545227A089903E40FAEDEBC0394B4940
9	mroocroft8	oB0*T.\ZK	+380631352654	0101000020E6100000A60A462575D23E40637FD93D79304940
10	lmacduff9	nC2{TJ}	+380904149292	0101000020E6100000645DDC4603883E4027A089B0E1314940
11	aanthona	kY4~*Z<U	+380857442917	0101000020E61000007A36AB3E57CB3E40910F7A36AB464940
12	tjansab	wF6\$XVx'Q	+380642007826	0101000020E61000008B6CE7FBA9913E408104C58F31374940

Рис. 5.1.1. Таблиця Client

Розташування адрес клієнтів на карті

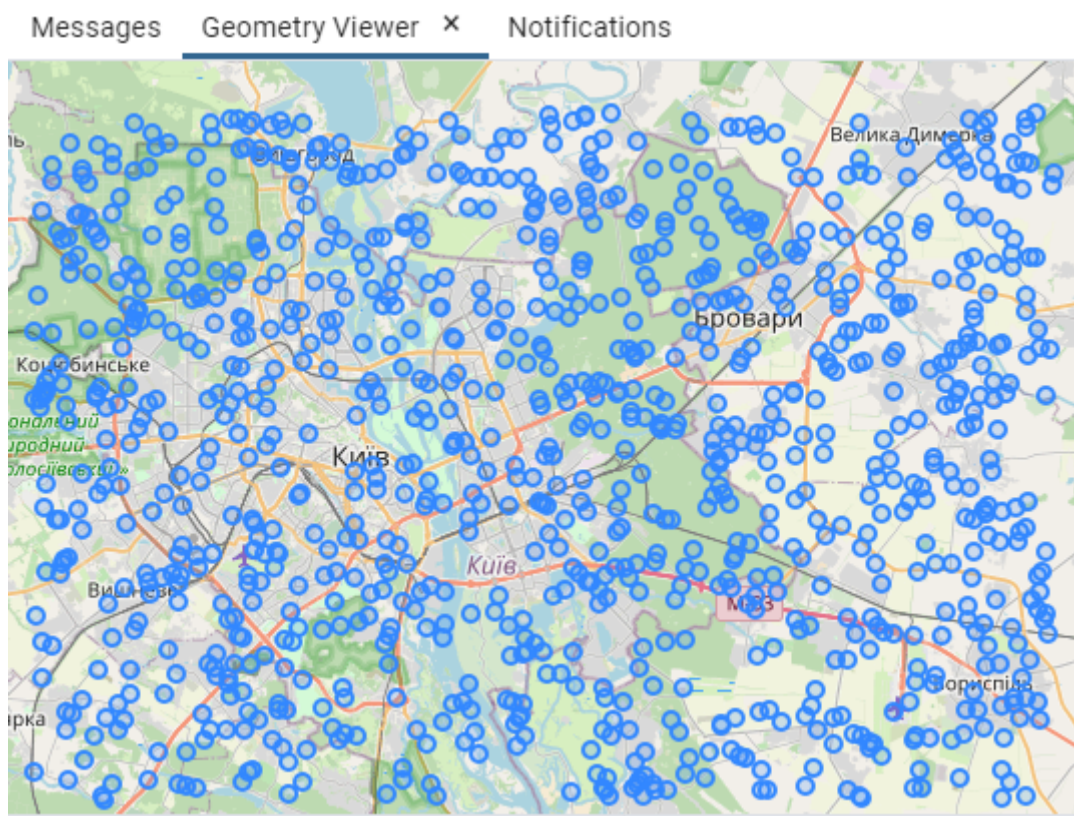


Рис. 5.1.2. Візуалізація геоданих таблиці Client

Створення таблиці Driver за описом у таблиці 4.14

```
CREATE TABLE Driver(
    driver_id SERIAL PRIMARY KEY,
    last_name VARCHAR(20) NOT NULL,
    first_name VARCHAR(20),
    driving_license VARCHAR(10) NOT NULL,
    driver_pnumber PHONE_NUMBER NOT NULL,
    driver_rating NUMERIC(2, 1)
);
```

Перевірка: `SELECT * FROM Driver;`

	driver_id [PK] integer	last_name character varying (20)	first_name character varying (20)	driving_license character varying (10)	driver_pnumber character	driver_rating numeric (2,1)
1	258	Piwall	Davy	5083652341	+380406408458	2.5
2	74	Scandred	Essie	4630319425	+380689908110	2.5
3	9	Grieves	Leena	7935170346	+380957514909	2.4
4	440	Birkhead	Demetri	1580522404	+380598994401	2.7
5	299	Swalle	Hymie	4077997484	+380123902918	2.4
6	403	Gherardi	Ricki	3521357783	+380227215276	2.6
7	250	Lait	Clarey	8809050343	+380541409014	2.7
8	39	Brislan	Elston	1876150109	+380893647593	2.6
9	540	Schreiner	Lemuel	3517838914	+380404978864	2.6
10	396	Scutt	Constantino	2927129620	+380713690586	2.6
11	268	MacKnocker	Nicko	8266759541	+380564993848	2.4
12	692	Huws	Zelma	1683695302	+380162499969	2.5
13	693	Wank	Netta	6425301548	+380948819259	2.6
14	295	O'Kynsillaghe	Nonie	6284549070	+380630592400	2.4
15	373	Stichall	Cornv	1605289550	+380336825476	2.5

Рис. 5.1.3. Таблиця Driver

Створення таблиці Discount за описом у таблиці 4.15

```
CREATE TABLE Discount (
    discount_id SERIAL PRIMARY KEY,
    discount_name VARCHAR(20) NOT NULL,
    discount_size NUMERIC(3, 2) NOT NULL CHECK (discount_size >= 0
AND discount_size <= 1)
);
```

Перевірка: `SELECT * FROM Discount;`

	discount_id [PK] integer	discount_name character varying (20)	discount_size numeric (3,2)
1	1	Regular client	0.15
2	2	10th order	0.05
3	3	New year	0.24
4	4	Weekend Special	0.10
5	5	Holiday Promo	0.18
6	6	Early Bird Offer	0.12
7	7	Summer Savings	0.15
8	8	Student Discount	0.08

Рис. 5.1.4. Таблиця Discount

Створення таблиці Service_type за описом у таблиці 4.16

```
CREATE TABLE Service_type (
    st_id SERIAL PRIMARY KEY,
    service_name VARCHAR(20) NOT NULL,
    price NUMERIC(10, 2) NOT NULL
);
```

Перевірка: `SELECT * FROM Service_type;`

	st_id [PK] integer	service_name character varying (20)	price numeric (10,2)
1	1	Child Seat	20.00
2	2	Towing Service	50.00
3	3	Pet Transportation	30.00
4	4	Event Transportation	50.00
5	5	Delivery	20.00
6	6	Airport Shuttle	25.00
7	7	Luxury Car Upgrade	20.00

Рис. 5.1.5. Таблиця Service_type

Створення таблиці Tariff за описом у таблиці 4.17

```
CREATE TABLE Tariff(
    tariff_id SERIAL PRIMARY KEY,
    tariff_name VARCHAR(20) NOT NULL,
    price NUMERIC(10, 2) NOT NULL
);
```

Перевірка: **SELECT * FROM Tariff;**

	tariff_id [PK] integer	tariff_name character varying (20)	price numeric (10,2)
18	1	Economy	10.00
19	2	Standard	15.00
20	3	Comfort	20.00
21	4	Business	25.00
22	5	Universal	18.00
23	6	Minibus	30.00

Рис. 5.1.6. Таблиця Tariff

Створення таблиці Orders за описом у таблиці 4.18

```
CREATE TABLE Orders(
    order_id SERIAL PRIMARY KEY,
    date_time TIMESTAMP DEFAULT NOW(),
    departure_point GEOGRAPHY(POINT) NOT NULL,
    destination_point GEOGRAPHY(POINT) NOT NULL,
    payment PAYMENT_METHOD,
    tariff_id INTEGER REFERENCES Tariff(tariff_id) NOT NULL,
    client_id INTEGER REFERENCES Client(client_id) NOT NULL,
    discount_id INTEGER REFERENCES Discount(discount_id)
);
```

Перевірка: **SELECT * FROM Orders;**

order_id [PK] integer	date_time timestamp without time zone	payment payment_method	tariff_id integer	client_id integer	discount_id integer	departure_point geography	destination_point geography
2001	2023-09-16 02:37:00	card	2	929	[null]	0101000020E610000031992A1895B43E401361C3D32B454940	0101000020E6100000B7D100DE02A93E407FFB3A70CE284940
2002	2023-09-18 22:47:00	cash	3	278	[null]	0101000020E6100000F241CF66D5E73E409FABADD85F4649...	0101000020E61000001F85E851B88E3E40D6C56D3480374940
2003	2023-08-23 21:00:00	card	2	9	[null]	0101000020E61000004F1E166A4D533E40D712F241CF264940	0101000020E6100000F241CF66D5973E4014AE47E17A3C4940
2004	2023-07-28 23:25:00	card	6	64	[null]	0101000020E610000096B20C71ACCB3E40C7293A92CB4749...	0101000020E61000004FAF946588833E40598638D6C53D4940
2005	2023-09-15 03:35:00	cash	4	491	[null]	0101000020E6100000EBE2361AC0FB3E403333333334B4940	0101000020E610000061545227A0F93E400F9C3A2B4474940
2006	2023-05-09 13:12:00	cash	6	143	[null]	0101000020E6100000F931E6AE25E43E40BE30992A182D4940	0101000020E61000001FF46C567D6E3E4095D40968223C4940
2007	2023-10-05 06:11:00	cash	5	780	[null]	0101000020E610000068226C787AE53E4061545227A0394940	0101000020E610000097FF907EFB8A3E40ED9E3C2CD4324940
2008	2023-07-22 14:59:00	card	2	718	[null]	0101000020E610000089D2DEE00B833E4097900F7A36334940	0101000020E6100000FADEB0C39C33E404B598638D6454940
2009	2023-08-23 19:54:00	cash	4	222	[null]	0101000020E6100000D6C56D3480A73E40C7BAB88D064049...	0101000020E6100000744694F6064F3E407DAEB6627F314940
2010	2023-11-01 14:49:00	card	6	477	[null]	0101000020E6100000E9B7AF03E7FC3E406DE7FBA9F12A49...	0101000020E6100000CE1951DA1BCC3E40E6AE25E4833E4940
2011	2023-06-11 15:03:00	cash	3	899	[null]	0101000020E61000005DFAE43FAED5B3E40401361C3D3434940	0101000020E61000009D11A5BDC1573E40D200DE02093A49...
2012	2023-12-15 21:27:00	card	4	993	[null]	0101000020E61000003EE8D9ACFADC3E400C022B87164149...	0101000020E6100000A60A462575623E400C93A98251314940
2013	2023-08-26 00:22:00	cash	3	527	[null]	0101000020E61000003A92CB7F48EF3E40C1A8A44E404349...	0101000020E6100000AB3E575BB19F3E4003098A1F63364940
2014	2023-07-23 21:04:00	cash	6	123	[null]	0101000020E61000009CC420B072F83E4031992A18952C4940	0101000020E610000024B9FC87F48B83E40F853E3A59B444940
2015	2023-09-15 08:42:00	card	5	113	[null]	0101000020E6100000BBB8D06F0C63E40B4C876BE9F3A49...	0101000020E6100000287E8CB96BD93E407CF2B0506B3A4940
2016	2023-11-08 02:11:00	cash	3	295	[null]	0101000020E61000009FCDAAACFD5663E40D95F764F1E3649...	0101000020E610000076711B0DE08D3E40AF25E4839E354940
2017	2023-12-05 17:07:00	card	1	678	[null]	0101000020E6100000E25817B7D1903E40A4703D0AD73349...	0101000020E6100000B459F5B9DAAA3E40228E75711B454940
2018	2023-06-09 11:14:00	cash	3	855	[null]	0101000020E6100000A167B3EA73853E40A4703D0AD72B49...	0101000020E61000003F355EBA49DC3E40ADFA5C6DC53E49...
2019	2023-06-14 11:46:00	cash	5	655	[null]	0101000020E61000001AC05B2041513E40E5F21FD26F3F4940	0101000020E610000023DBF97E6A7C3E405227A089B0414940
2020	2023-11-10 02:37:00	card	4	878	[null]	0101000020E6100000000000000000C03E40CDDCCCCCC344...	0101000020E6100000C217265305B33E40CE1951DA1B344940
2021	2023-10-15 14:06:00	card	1	752	[null]	0101000020E610000058A835CD3B4E3E40CF66D5E76A2B49...	0101000020E6100000CAC342AD695E3E4052499D0263A4940

Рис. 5.1.7. Таблиця Orders

Розташування точок відправлення на карті

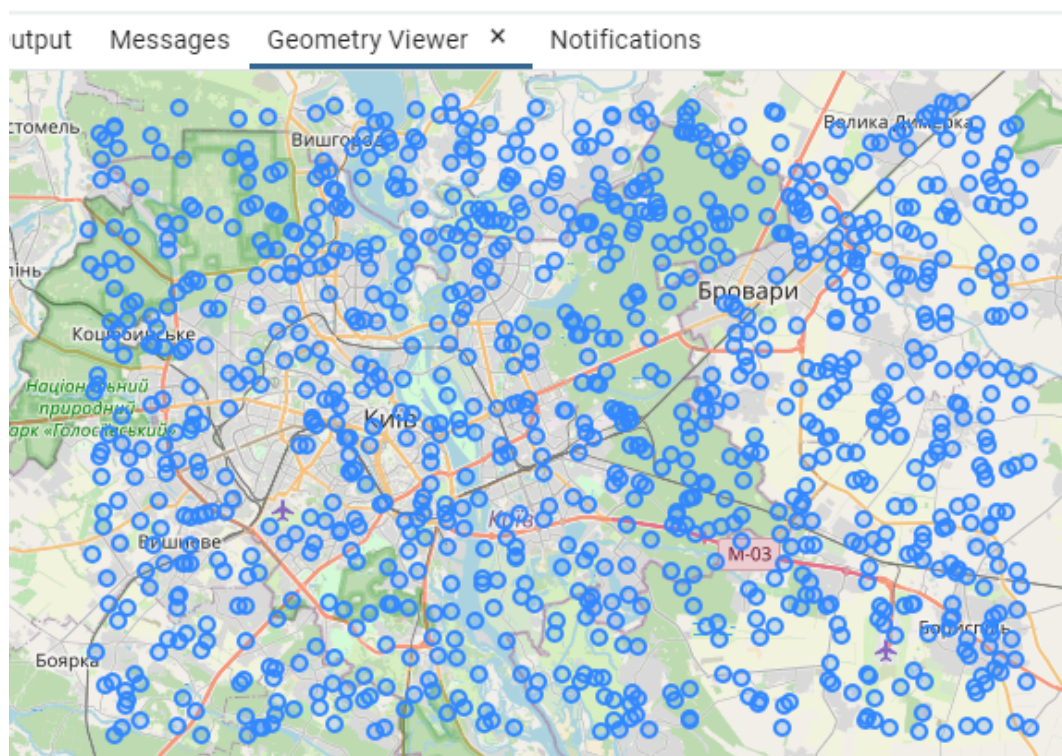


Рис. 5.1.8. Візуалізація геоданих таблиці Orders

Створення таблиці Service за описом у таблиці 4.19

```
CREATE TABLE Service(
service_id SERIAL PRIMARY KEY,
st_id INTEGER REFERENCES Service_type(st_id) NOT NULL,
order_id INTEGER REFERENCES Orders(order_id) NOT NULL
);
```

Перевірка: `SELECT * FROM Service;`

	service_id [PK] integer	st_id integer	order_id integer
1	1	14	52
2	2	6	1115
3	3	17	1059
4	4	15	1614
5	5	14	1667
6	6	4	844
7	7	19	1541
8	8	11	87

Рис. 5.1.9. Таблиця Service

Створення таблиці Model за описом у таблиці 4.20

```
CREATE TABLE Model(
    model_id SERIAL PRIMARY KEY,
    brand VARCHAR(20) NOT NULL,
    model_name VARCHAR(30) NOT NULL,
    manufacturing_year INTEGER
);
```

Перевірка: `SELECT * FROM Model;`

	model_id [PK] integer	brand character varying (20)	model_name character varying (30)	manufacturing_year integer
1	1	Chevrolet	Astro	2002
2	2	Subaru	Brat	1985
3	3	Lincoln	Continental	1995
4	4	Nissan	Pathfinder	2002
5	5	Jeep	Wrangler	1993
6	6	Acura	RL	2012
7	7	Ford	Explorer	1995

Рис. 5.1.10. Таблиця Model

Створення таблиці Transport за описом у таблиці 4.21

```
CREATE TABLE Transport(
    transport_id SERIAL PRIMARY KEY,
    model_id INTEGER REFERENCES Model(model_id) NOT NULL,
    tariff_id INTEGER REFERENCES Tariff(tariff_id) NOT NULL,
    plate_number VARCHAR(8) UNIQUE NOT NULL
CHECK(length(plate_number) >= 3),
    color VARCHAR(10)
);
```

Перевірка: `SELECT * FROM Transport;`

	transport_id [PK] integer	model_id integer	tariff_id integer	plate_number character varying (8)	color character varying (10)
1	1	33	5	WA5547IW	Maroon
2	2	63	3	FL4412MO	Maroon
3	3	83	4	RZ3697QZ	Khaki
4	4	21	1	FN3090HC	Goldenrod
5	5	83	3	CK3534LN	Aquamarine
6	6	88	5	TS0127JX	Yellow
7	7	103	5	DE3633BW	Turquoise

Рис. 5.1.11. Таблиця Transport

Створення таблиці Ride за описом у таблиці 4.22

```
CREATE TABLE Ride(
    ride_id SERIAL PRIMARY KEY,
    order_id INTEGER REFERENCES Orders(order_id) UNIQUE NOT NULL,
    driver_id INTEGER REFERENCES Driver(driver_id) NOT NULL,
    ride_time TIMESTAMP DEFAULT NOW(),
    distance NUMERIC(10, 2),
    total_price NUMERIC(10, 2),
    transport_id INTEGER REFERENCES Transport(transport_id)
);
```

Перевірка: `SELECT * FROM Ride;`

	ride_id [PK] integer	order_id integer	driver_id integer	distance numeric (10,2)	total_price numeric (10,2)	transport_id integer	ride_time timestamp without time zone
1	1	1	10	11.07	199.26	255	2023-10-19 01:53:00
2	2	2	571	44.45	860.10	236	2023-08-19 08:25:00
3	3	3	461	22.18	463.60	144	2023-08-02 05:08:00
4	4	4	99	18.39	314.85	13	2023-08-09 08:43:00
5	5	5	137	22.00	330.00	35	2023-09-19 19:34:00
6	6	6	648	20.13	397.34	134	2023-11-09 18:41:00
7	7	7	676	32.85	841.25	204	2023-04-15 01:07:00
8	8	8	88	25.04	500.80	153	2023-08-30 06:19:00
9	9	9	591	21.19	473.80	206	2023-11-18 09:45:00
10	10	10	484	7.06	138.60	179	2023-06-07 10:50:00
11	11	11	575	28.33	728.25	173	2023-09-01 09:02:00
12	12	12	226	15.66	426.50	23	2023-05-17 00:53:00
13	13	13	191	50.43	937.74	45	2023-03-27 18:21:00
14	14	14	363	6.73	140.95	252	2023-09-12 14:43:00
15	15	15	106	22.81	684.30	227	2023-12-04 07:16:00
16	16	16	141	9.61	240.25	240	2023-05-29 04:07:00

Рис. 5.1.12. Таблиця Ride

Створення таблиці Feedback за описом у таблиці 4.23

```
CREATE TABLE Feedback(
    feedback_id SERIAL PRIMARY KEY,
    ride_id INTEGER REFERENCES Ride(ride_id) NOT NULL,
    rating INTEGER NOT NULL CHECK (rating >= 1 AND rating <= 5),
    commentary TEXT
);
```

Перевірка: **SELECT * FROM Feedback;**

	feedback_id [PK] integer	ride_id integer	rating integer	commentary text
33	4	1209	3	In sagittis dui vel nisl. Duis ac nibh.
34	5	793	4	Duis consequat dui nec nisi volutpat eleifend.
35	6	762	2	[null]
36	7	1283	2	Vestibulum rutrum rutrum neque. Aenean auctor gravida sem.
37	8	788	5	Duis mattis egestas metus. Aenean fermentum.
38	9	1581	1	[null]
39	10	1296	4	Vivamus metus arcu, adipiscing molestie, hendrerit at, vulputate vitae, nisl.

Рис. 5.1.13. Таблиця Feedback

Геруємо ER-діаграму створеної бази даних у PgAdmin4.

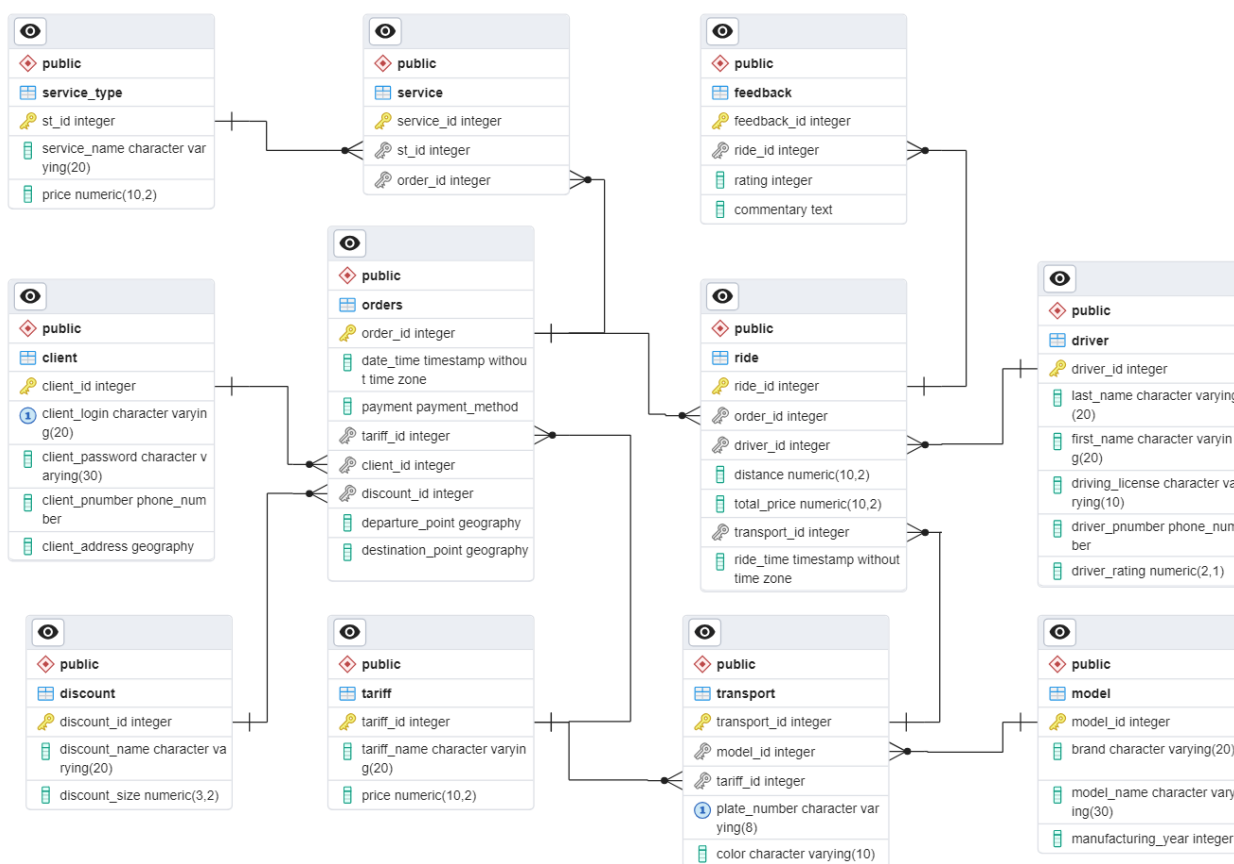


Рис. 5.1.14. ER-діаграма згенерована у PgAdmin4

Заповнюємо таблиці даними за допомогою генераторів CSV файлів, випадкової генерації значень у циклі та з використанням спеціально розроблених тригерів, а також вставки даних через звичайну команду INSERT ([Додаток 1](#)).

5.2 Створення ролей і користувачів

У розділі “Опис предметного середовища було” наведено модель користувачів бази даних. Для створення багатокористувацького інтерфейсу створюємо ролі та визначаємо їх права.

Створення ролей

```
CREATE ROLE taxi_admin_role;
CREATE ROLE driver_role;
CREATE ROLE client_role;
```

Присвоєння прав

```
GRANT SELECT ON TABLE Client, Orders, Service, Feedback,
Service_type, Tariff, Transport, Ride TO client_role;
GRANT INSERT, DELETE ON TABLE Orders, Service, Feedback TO
client_role;
GRANT UPDATE ON TABLE Client, Orders TO client_role;
GRANT USAGE, SELECT ON SEQUENCE orders_order_id_seq,
service_service_id_seq, feedback_feedback_id_seq TO client_role;

GRANT SELECT ON TABLE Client, Orders, Service, Feedback,
Service_type, Tariff, Transport, Model, Driver, Ride, Discount TO
driver_role;
GRANT INSERT ON TABLE Ride TO driver_role;
GRANT USAGE, SELECT ON SEQUENCE ride_ride_id_seq TO driver_role;

GRANT INSERT ON TABLE Transport, Model, Driver, Discount,
Service_type, Tariff TO taxi_admin_role;
GRANT UPDATE ON TABLE Driver, Discount, Service_type, Tariff TO
taxi_admin_role;
GRANT SELECT ON TABLE Client, Orders, Service, Feedback,
Service_type, Tariff, Transport, Model, Driver, Ride, Discount TO
taxi_admin_role;
GRANT USAGE, SELECT ON SEQUENCE transport_transport_id_seq,
model_model_id_seq, driver_driver_id_seq,
discount_discount_id_seq, service_type_st_id_seq,
tariff_tariff_id_seq TO taxi_admin_role;
```

Створення користувачів з паролями

```
CREATE USER "db_admin" WITH PASSWORD 'long_password';  
CREATE USER "driver1" WITH PASSWORD 'medium_password';  
CREATE USER "client1" WITH PASSWORD 'short_password';
```

Присвоєння ролей користувачів

```
GRANT taxi_admin_role TO "db_admin";  
GRANT driver_role TO "driver1";  
GRANT client_rode TO "client1";
```

6 РОБОТА З БАЗОЮ ДАНИХ

6.1 Тексти генераторів

При створенні таблиць було використано генератори з автоінкрементом типу даних `SERIAL` для первинних ключів таблиць.

```
client_id SERIAL PRIMARY KEY - PK таблиці Client
driver_id SERIAL PRIMARY KEY - PK таблиці Driver
discount_id SERIAL PRIMARY KEY - PK таблиці Discount
st_id SERIAL PRIMARY KEY - PK таблиці Service_type
tariff_id SERIAL PRIMARY KEY - PK таблиці Tariff
order_id SERIAL PRIMARY KEY - PK таблиці Orders
service_id SERIAL PRIMARY KEY - PK таблиці Service
model_id SERIAL PRIMARY KEY - PK таблиці Model
transport_id SERIAL PRIMARY KEY - PK таблиці Transport
ride_id SERIAL PRIMARY KEY - PK таблиці Ride
feedback_id SERIAL PRIMARY KEY - PK таблиці Feedback
```

На Рисунок 6.1 продемонстровано створені генератори послідовностей.

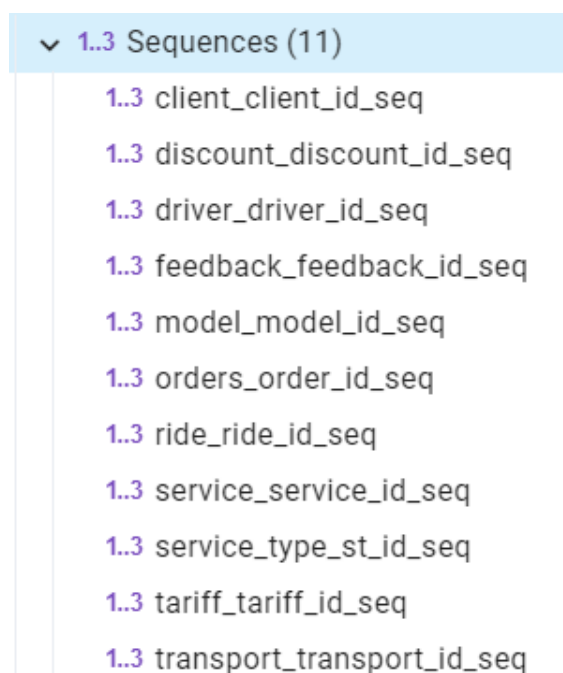


Рис. 6.1.1. Генератори бази даних Taxi

6.2 Тексти збережених процедур/функцій

Призначення: Визначення прибутку компанії за певний рік.

Опис: Сума вартостей усіх поїздок, виконаних за заданий рік.

```
CREATE OR REPLACE FUNCTION Revenue_Year(work_year INTEGER)
RETURNS NUMERIC(10, 2)
LANGUAGE plpgsql AS $$
DECLARE
    total_revenue NUMERIC(10, 2);
BEGIN
    SELECT SUM(total_price) INTO total_revenue
    FROM Ride
    WHERE EXTRACT(YEAR FROM ride_time) = work_year;

    RETURN total_revenue;
END $$;
```

Демонстрація: `SELECT Revenue_Year(2023) AS total_revenue`

	total_revenue numeric
1	27198876.16

Рис. 6.2.1. Демонстрація роботи функції Revenue_Year()

Призначення: Аналіз оцінок роботи водія для перевірки його професійності; виведення відгуків про водія для ознайомлення користувачами.

Опис: Виведення відгуків, що надавались на поїздки заданого водія з сортуванням за спаданням оцінки

```
CREATE OR REPLACE FUNCTION Drivers_Feedback(given_driver INTEGER)
RETURNS SETOF Feedback AS
$$
    SELECT f.*
    FROM Feedback f
    JOIN Ride r ON f.ride_id = r.ride_id
    WHERE r.driver_id = given_driver
    ORDER BY f.rating DESC;
$$ LANGUAGE SQL;
```


Демонстрація: `SELECT * FROM Drivers_Feedback(11);`

	feedback_id integer	ride_id integer	rating integer	commentary text
1	19457	51308	4	[null]
2	20012	36456	4	[null]
3	61672	47271	4	[null]
4	20111	45726	4	[null]
5	36277	15157	4	[null]
6	20540	29406	4	[null]
7	59139	57029	4	[null]
8	49592	59808	4	[null]
9	22077	36347	4	[null]
10	6833	41122	4	[null]
11	22370	28659	4	[null]

Рис. 6.2.2. Демонстрація роботи функції `Drivers_Feedback()`

Призначення: Визначення більш популярного способу оплати для подальшого планування фінансів.

Опис: Порівняння кількості замовлень зі способом оплати картою та готівкою та виведення повідомлення про те, що є більш популярним.

```
CREATE OR REPLACE PROCEDURE Cash_vs_Card()
AS $$
DECLARE
    cash_count INTEGER;
    card_count INTEGER;
BEGIN
    SELECT COUNT(*) INTO cash_count
    FROM Orders
    WHERE payment = 'cash';
    SELECT COUNT(*) INTO card_count
    FROM Orders
    WHERE payment = 'card';
    IF card_count > cash_count THEN
        RAISE NOTICE 'More used cards for payment: %', card_count;
    ELSE
        RAISE NOTICE 'More used cash for payment: %', cash_count;
    END IF;
END $$ LANGUAGE plpgsql;
```

Демонстрація: `CALL Cash_vs_Card();`

Data Output	Messages	Notifications
	ПОВІДОМЛЕННЯ: More used cards for payment: 30139 CALL	

Рис. 6.2.3. Демонстрація роботи функції `Cash_vs_Card()`

Призначення: Визначити штраф водія за заданий місяць для використання цієї інформації при нарахуванні заробітної плати.

Опис: Знаходимо поїздки водія за заданий місяць поточного року, у яких різниця між часом замовлення і часом початку поїздки складає більше 10 хвилин. Знаходимо суму затримок від очікуваного часу (сумарний час затримок за місяць) та множимо на розмір штрафу за хвилину - 2 грн.

```
CREATE OR REPLACE FUNCTION Driver_Fine(driver INTEGER, work_month
INTEGER)
RETURNS NUMERIC(10, 2)
LANGUAGE plpgsql AS $$
DECLARE
    fine NUMERIC(10, 2);
BEGIN
    SELECT
        SUM ((EXTRACT(EPOCH FROM (r.ride_time - o.date_time)) / 60)-10)*2
        INTO fine
    FROM Ride r
    JOIN Orders o ON o.order_id = r.order_id
    WHERE EXTRACT(MONTH FROM ride_time) = work_month
        AND EXTRACT(YEAR FROM ride_time) = EXTRACT(YEAR FROM
CURRENT_DATE)
        AND driver_id = driver
        AND EXTRACT(EPOCH FROM (r.ride_time - o.date_time)) /
60 > 10;

    RETURN fine;
END $$;
```

Демонстрація: `SELECT Driver_Fine(6, 7) AS drivers_fine;`

	drivers_fine numeric
1	162.00

Рис. 6.2.4. Демонстрація роботи функції Driver_Fine()

Призначення: Автоматичне розрахування заробітної плати водія за місяць з урахуванням його штрафів

Опис: Знаходимо поїздки водія за заданий місяць поточного року, та обраховуємо суму їх вартостей, після чого множимо отримане значення на 0.2 (20%). Від отриманого значення віднімаємо штраф водія за цей місяць.

```
CREATE OR REPLACE FUNCTION Driver_Salary(driver INTEGER,
work_month INTEGER)
RETURNS NUMERIC(10, 2)
LANGUAGE plpgsql AS $$
DECLARE
    salary NUMERIC(10, 2);
BEGIN
    SELECT SUM(total_price)*0.2 INTO salary
    FROM Ride
    WHERE EXTRACT(MONTH FROM ride_time) = work_month
          AND EXTRACT(YEAR FROM ride_time) = EXTRACT(YEAR
FROM CURRENT_DATE)
          AND driver_id = driver;
    RETURN salary - Driver_Fine(driver, work_month);
END $$;
```

Демонстрація: `SELECT Driver_Fine(6, 7) AS drivers_salary;`

	drivers_salary numeric
1	1094.67

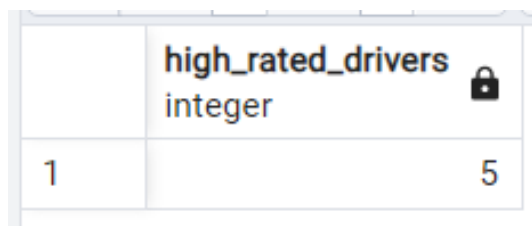
Рис. 6.2.5. Демонстрація роботи функції Driver_Fine()

Призначення: Визначення кількості водіїв, що мають рейтинг вище певного порогу для звітності про якість роботи служби.

Опис: Кількість водіїв, рейтинг яких вище заданого

```
CREATE OR REPLACE FUNCTION High_Rated_Drivers(min_rating
NUMERIC(2, 1))
RETURNS INTEGER AS $$
BEGIN
    RETURN (SELECT COUNT(*) FROM Driver WHERE driver_rating >=
min_rating);
END;
$$ LANGUAGE plpgsql;
```

Демонстрація: `SELECT High_Rated_Drivers(3) AS high_rated_drivers;`



	high_rated_drivers integer
1	5

Рис. 6.2.6. Демонстрація роботи функції High_Rated_Drivers()

Призначення: Визначення найпопулярнішої додаткової послуги за рік для планування знижок та маркетингу.

Опис: Обираємо послуги, що надавались у поточному році, сортуємо їх за спаданням за кількістю використань та повертаємо лише перший рядок - найбільш вживана додаткова послуга.

```
CREATE OR REPLACE FUNCTION Most_Used_Service()
RETURNS VARCHAR(20) AS $$
BEGIN
    RETURN (SELECT st.service_name
            FROM Service s
            JOIN Service_type st ON s.st_id = st.st_id
            JOIN Orders o ON o.order_id = s.order_id
            WHERE EXTRACT(YEAR FROM o.date_time) = EXTRACT(YEAR
FROM NOW()))
    GROUP BY st.service_name
    ORDER BY COUNT(s.order_id) DESC
    LIMIT 1);
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

Демонстрація: `SELECT EXTRACT(YEAR FROM NOW()) AS current_year,`
`Most_Used_Service() AS most_used_service;`

	current_year numeric	most_used_service character varying
1	2023	Business Upgrade

Рис. 6.2.7. Демонстрація роботи функції `Most_Used_Service()`

Призначення: Виведення списку додаткових послуг замовлення для створення чеків або звітності.

Опис: Виводимо номер заданого замовлення. Створюємо вибірку додаткових послуг та за допомогою циклу проходимо по всіх рядках та виводимо назву послуги.

```
CREATE OR REPLACE PROCEDURE Order_Services_Info(order_i INTEGER)
AS $$
```

```
DECLARE
```

```
    service_name_i TEXT;
```

```
BEGIN
```

```
    RAISE NOTICE 'Order ID: %', order_i;
```

```
    FOR service_name_i IN SELECT st.service_name
```

```
        FROM Service s
```

```
        JOIN Service_type st ON s.st_id = st.st_id
```

```
        WHERE s.order_id = order_i
```

```
    LOOP
```

```
        RAISE NOTICE 'Service: %', service_name_i;
```

```
    END LOOP;
```

```
END $$ LANGUAGE plpgsql;
```

Демонстрація: `CALL Order_Services_Info(2005);`

```
ПОВІДОМЛЕННЯ: Order ID: 2005
ПОВІДОМЛЕННЯ: Service: Business Upgrade
ПОВІДОМЛЕННЯ: Service: Multilingual Driver
CALL
```

Рис. 6.2.8. Демонстрація роботи функції `Order_Services_Info()`

Призначення: Підтримання актуальної інформації про водіїв та їх важливі документи.

Опис: Оновлення значення коду водійського посвідчення для заданого водія. Виведення повідомлення про успішну заміну та нове значення водійського посвідчення

```
CREATE OR REPLACE PROCEDURE Update_Driving_Licennse(driver_i
INTEGER, new_driving_license VARCHAR(20))
AS $$
BEGIN
    UPDATE Driver
    SET driving_license = new_driving_license
    WHERE driver_id = driver_i;
    RAISE NOTICE 'New driver license is %', new_driving_license;
END;
$$ LANGUAGE plpgsql;
```

Демонстрація:

До оновлення:

	driver_id [PK] integer	last_name character varying (20)	first_name character varying (20)	driving_license character varying (10)	driver_pnumber character	driver_rating numeric (2,1)
1	4	Corkell	Phillip	2872801027	+380095971365	2.3

Рис. 6.2.9. Запис у Driver з driver_id = 4 до виконання функції

```
CALL Update_Driving_Licennse(4, '00777890');
```

Після оновлення:

```
ПОВІДОМЛЕННЯ: New driver licence is 00777890
CALL
```

Рис. 6.2.10. Повідомлення про оновлення даних

	driver_id [PK] integer	last_name character varying (20)	first_name character varying (20)	driving_license character varying (10)	driver_pnumber character	driver_rating numeric (2,1)
1	4	Corkell	Phillip	00777890	+380095971365	2.3

Рис. 6.2.11. Запис у Driver з driver_id = 4 після виконання функції

Призначення: Відміна замовлення клієнтом до початку поїздки.

Опис: Якщо замовлення вже має створену по ньому поїздку, тобто обслуговування розпочалось, то клієнт не може видалити замовлення та виводиться виключення з повідомленням про неможливість видалення замовлення. Інакше проводиться видалення всіх додаткових послуг, що були включені у замовлення, а також самого замовлення, після чого виводиться повідомлення про успішне видалення замовлення.

```
CREATE OR REPLACE PROCEDURE Delete_Order(order_i INTEGER)
AS $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Orders WHERE order_id = order_i)
    THEN
        RAISE EXCEPTION 'Order with specified order_id does not
exist.';
    END IF;

    IF EXISTS (SELECT 1 FROM Ride WHERE order_id = order_i) THEN
        RAISE EXCEPTION 'Cannot delete order with associated
ride.';
    ELSE
        DELETE FROM Service WHERE order_id = order_i;
        DELETE FROM Orders WHERE order_id = order_i;
        RAISE NOTICE 'Order %% deleted', order_i;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Демонстрація:

```
CALL Delete_Order(65000);
```

```
ERROR: Order with specified order_id does not exist.
CONTEXT: Функція PL/pgSQL delete_order(integer) рядок 4 в RAISE
```

Рис. 6.2.12. Виведення повідомлення про відсутність замовлення

```
CALL Delete_Order(6000);
```

```
ERROR: Cannot delete order with associated ride.  
CONTEXT: Функция PL/pgSQL delete_order(integer) рядок 8 в RAISE
```

Рис. 6.2.13. Виведення повідомлення про неможливість
видалення замовлення

```
CALL Delete_Order(60001);
```

```
ПОВІДОМЛЕННЯ: Order #60001 deleted  
CALL
```

Рис. 6.2.13. Виведення повідомлення про
видалення замовлення

6.3 Тексти тригерів

Тригер призначення знижки до нового замовлення за виконання умов отримання знижки

Призначення: Автоматизація призначення знижок для зменшення вірогідності помилок або маніпуляцій при наданні знижок водіями, а також відстеження користувачів, що є постійними клієнтами служби таксі.

Опис додаткової функції: Функція перевірки для знижки №1 “Постійний клієнт”. Знижка постійного клієнта надається, якщо створене замовлення є п’ятнадцятим, або більше, протягом останніх 3 місяців для користувача. Функція виконує перевірку на виконання умови та повертає результат як булеве значення.

```
CREATE OR REPLACE FUNCTION condition_discount1(client INTEGER)
RETURNS BOOLEAN AS $$
BEGIN
    RETURN (
        EXISTS (
            SELECT 1
            FROM Orders
            WHERE
                client_id = client
                AND date_time >= current_date - interval '3 months'
            GROUP BY client_id
            HAVING COUNT(*)+1 >= 15
        )
    );
END;
$$ LANGUAGE plpgsql;
```

Опис додаткової функції: Функція перевірки для знижки №2 “Десяте замовлення”. Знижка надається на кожне десяте замовлення користувача. Функція виконує перевірку на виконання умови та повертає результат як булеве значення.

```
CREATE OR REPLACE FUNCTION condition_discount2(client INTEGER)
RETURNS BOOLEAN AS $$
DECLARE
    oder_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO oder_count
    FROM Orders
    WHERE client_id = client;

    RETURN (oder_count + 1) % 10 = 0;
END;
```

```
$$ LANGUAGE plpgsql;
```

Опис основної функції та триггеру: Функція призначення знижки з попередньою перевіркою умов. У функції умови надання знижок перевіряються поступово, починаючи зі знижки з найбільшим розміром і закінчуючи найменшим. Таким чином спочатку проводиться перевірка на святкову знижку “Новий рік”, де перевіряється чи дата співпадає з 31.12, потім відбувається виклик функцій для перевірки виконання умов для знижок “Постійний клієнт” та “Десяте замовлення”. Функція викликається тригером.

```
CREATE OR REPLACE FUNCTION set_discount_function()
RETURNS TRIGGER AS $$
DECLARE
    discount_id_value INTEGER;
BEGIN
    IF (EXTRACT(MONTH FROM NEW.date_time) = 12
        AND EXTRACT(DAY FROM NEW.date_time) = 31) THEN
        discount_id_value := 3;
    ELSIF condition_discount1(NEW.client_id) THEN
        discount_id_value := 1;
    ELSIF condition_discount2(NEW.client_id) THEN
        discount_id_value := 2;
    ELSE
        discount_id_value := NULL; -- Не відповідає жодній умові
    END IF;

    NEW.discount_id := discount_id_value;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Тригер, що викликається перед додаванням запису у таблицю Orders

```
CREATE TRIGGER set_discount_trigger
BEFORE INSERT ON Orders
FOR EACH ROW
EXECUTE FUNCTION set_discount_function();
```

Тригер призначення перерахунку рейтингу водія після додавання нового відгуку на поїздку, яку він виконав.

Призначення: Підтримання актуальності рейтингу водія для аналізу ефективності його роботи та для перегляду відповідної інформації клієнтами.

Опис: Функція обрахунку та оновлення рейтингу. Для доданого відгуку знаходиться водій, який виконував поїздку, на яку залишили цей відгук. Розраховується середнє значення всіх відгуків знайденого водія. Розраховане середнє значення призначається новим значенням рейтинга водія.

```
CREATE OR REPLACE FUNCTION update_driver_rating()
RETURNS TRIGGER AS $$
DECLARE
    avg_rating NUMERIC(2, 1);
    update_driver_id INTEGER;
BEGIN
    SELECT r.driver_id
    INTO update_driver_id
    FROM Feedback f
    JOIN Ride r ON f.ride_id = r.ride_id
    WHERE f.feedback_id = NEW.feedback_id;

    SELECT AVG(f.rating)::NUMERIC(2, 1)
    INTO avg_rating
    FROM Feedback f
    JOIN Ride r ON f.ride_id = r.ride_id
    WHERE r.driver_id = update_driver_id;

    UPDATE Driver
    SET driver_rating = avg_rating
    WHERE driver_id = update_driver_id;

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

Тригер, що викликається після додавання запису у таблицю Feedback

```
CREATE TRIGGER update_driver_rating_trigger
AFTER INSERT ON Feedback
FOR EACH ROW
EXECUTE FUNCTION update_driver_rating();
```

Тригер для обрахунку відстані між точками початку та кінця поїздки

Призначення: Автоматизація розрахунку відстані для подальшого обрахунку вартості поїздки, а також аналізу довжини поїздки.

Опис: Функція призначення відстані для створеного запису про поїздку з використанням функції ST_Distance з розширення PostGIS. Знайдена відстань ділиться на 1000 для переведення з метрів у кілометри. Оновлення запису про поїздку з обрахованою відстанню

```
CREATE OR REPLACE FUNCTION calculate_distance()
RETURNS TRIGGER AS $$
DECLARE
    calc_distance NUMERIC(10, 2);
BEGIN
    SELECT ST_Distance(o.departure_point,
o.destination_point)/1000
    INTO calc_distance
    FROM Orders o
    WHERE o.order_id = NEW.order_id;

    UPDATE Ride
    SET distance = calc_distance
    WHERE ride_id = NEW.ride_id;

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

Тригер, що викликається після додавання запису у таблицю Ride

```
CREATE TRIGGER calculate_distance_trigger
AFTER INSERT ON Ride
FOR EACH ROW
EXECUTE FUNCTION calculate_distance();
```

Тригер, для обрахунку вартості поїздки за наступною формулою:

вартість = (відстань · тариф + [сума вартостей додаткових послуг]) [· (1 – розмір знижки)]

Призначення: Автоматизація підрахунку вартості поїздки, спрощення роботи таксистів т

Опис: Функція обрахунку вартості. Для створеної поїздки визначається відстань, тариф, за наявності сума вартостей додаткових послуг та розмір знижки. Після чого розраховане значення призначається як вартість поїздки.

```
CREATE OR REPLACE FUNCTION calculate_total_price()
RETURNS TRIGGER AS $$
DECLARE
    calc_distance NUMERIC(10, 2);
    tariff_price NUMERIC(10, 2);
    services_sum NUMERIC(10, 2);
    discount_s NUMERIC(3, 2);

BEGIN
    SELECT ST_Distance(o.departure_point,
o.destination_point)/1000 INTO calc_distance
    FROM Orders o
    WHERE o.order_id = NEW.order_id;

    SELECT t.price INTO tariff_price
    FROM Orders o
    JOIN Tariff t ON t.tariff_id = o.tariff_id
    WHERE o.order_id = NEW.order_id;

    SELECT COALESCE(SUM(st.price), 0) INTO services_sum
    FROM Orders o
    JOIN Service s ON o.order_id = s.order_id
    JOIN Service_type st ON s.st_id = st.st_id
    WHERE o.order_id = NEW.order_id;

    SELECT COALESCE(d.discount_size, 0) INTO discount_s
    FROM Orders o
    LEFT JOIN Discount d ON d.discount_id = o.discount_id
    WHERE o.order_id = NEW.order_id;

    UPDATE Ride
    SET total_price = (calc_distance*tariff_price +
services_sum)*(1-discount_s)
    WHERE ride_id = NEW.ride_id;
```

```

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

Тригер, що викликається після додавання запису у таблицю Ride

```

CREATE TRIGGER calculate_total_price_trigger
AFTER INSERT ON Ride
FOR EACH ROW
EXECUTE FUNCTION calculate_total_price();

```

Тригер перевірки відповідності тарифів замовлення та транспорту, що використовують для виконання цього замовлення

Призначення: Запобігти виконання замовлень на транспорті невідповідного тарифу, для запобігання конфліктів з клієнтами, що отримали не ту якість поїздки, на яку розраховували.

Опис: Функція перевірки відповідності тарифів. За невідповідності викликається виняток з повідомленням про помилку та запис не буде додано у таблицю Ride

```

CREATE OR REPLACE FUNCTION check_tariff_match()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT tariff_id FROM Orders WHERE order_id =
NEW.order_id) !=
    (SELECT tariff_id FROM Transport WHERE transport_id =
NEW.transport_id) THEN
        RAISE EXCEPTION 'Tariff mismatch between order and
transport';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

Тригер, що виконується перед додаванням запису у таблицю Ride

```

CREATE TRIGGER check_tariff_match_trigger
BEFORE INSERT ON Ride
FOR EACH ROW
EXECUTE FUNCTION check_tariff_match();

```

Тригер перевірки рейтингу водія, якого намагаються видалити

Призначення: Запобігти видаленню водіїв з високим рейтингом, для збереження інформації про визначних працівників служби таксі.

Опис: Функція перевірки рейтингу водія. Якщо рейтинг більше дорівнює 4, то викликається виключення з повідомленням про неможливість видалення водія з високим рейтингом. Інакше проводиться каскадне видалення, де спочатку видаляються усі відгуки на поїздки водія, потім його поїздки і після цього сам запис про водія.

```
CREATE OR REPLACE FUNCTION
prevent_delete_high_rated_driver_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT driver_rating FROM Driver WHERE driver_id =
OLD.driver_id) >= 4.0 THEN
        RAISE EXCEPTION 'Cannot delete driver with high rating';
    ELSE
        DELETE
        FROM Feedback f
        WHERE f.ride_id IN(
            SELECT r.ride_id
            FROM Ride r
            WHERE r.driver_id = OLD.driver_id);

        DELETE FROM Ride WHERE driver_id = OLD.driver_id;
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Тригер, що виконується перед видаленням з таблиці Driver

```
CREATE TRIGGER prevent_delete_high_rated_driver
BEFORE DELETE ON Driver
FOR EACH ROW
EXECUTE FUNCTION prevent_delete_high_rated_driver_trigger();
```

Тригер оновлення рейтингу водія після видалення відгуку на його поїздку

Призначення: Підтримання актуальності рейтингу водія та запобігання ситуаціям, коли випадково залишений відгук впливає на рейтинг водія.

Опис: Функція переобрахунку рейтингу після видалення. Для всіх відгуків на поїздки обраховується середнє значення та призначається як новий рейтинг водія

```
CREATE OR REPLACE FUNCTION del_update_driver_rating()
RETURNS TRIGGER AS $$
DECLARE
    avg_rating NUMERIC(2, 1);
    update_driver_id INTEGER;
BEGIN
    SELECT r.driver_id
    INTO update_driver_id
    FROM Feedback f
    JOIN Ride r ON f.ride_id = r.ride_id
    WHERE f.feedback_id = OLD.feedback_id;

    SELECT AVG(f.rating)::NUMERIC(2, 1)
    INTO avg_rating
    FROM Feedback f
    JOIN Ride r ON f.ride_id = r.ride_id
    WHERE r.driver_id = update_driver_id;

    UPDATE Driver
    SET driver_rating = avg_rating
    WHERE driver_id = update_driver_id;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Тригер, що виконується після видалення з таблиці Feedback

```
CREATE TRIGGER del_update_driver_rating_trigger
AFTER DELETE ON Feedback
FOR EACH ROW
EXECUTE FUNCTION del_update_driver_rating();
```


Тригер для перевірки відповідності довжини нового паролю

Призначення: Забезпечення безпеки персональних акаунтів користувачів системи.

Опис: Функція перевірки нового паролю. Якщо довжина паролю менше 6 символів, викликається виключення з повідомленням, що пароль закороткий. При невалідності пароль не замінюється на новий.

```
CREATE OR REPLACE FUNCTION validate_client_password_length()  
RETURNS TRIGGER AS $$  
BEGIN  
    IF LENGTH(NEW.client_password) < 6 THEN  
        RAISE EXCEPTION 'Client password must be at least 6  
characters long.';  
    END IF;  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

Тригер, що виконується перед оновленням значень Client

```
CREATE TRIGGER validate_new_client_password  
BEFORE UPDATE ON Client  
FOR EACH ROW  
EXECUTE FUNCTION validate_client_password_length();
```

6.4 Тексти представлень

Опис: Представлення суми вартостей поїздок за місяцем для поточного року.

Призначення: Визначення більш прибуткових місяців та подальшого фінансового планування компанії.

```
CREATE OR REPLACE VIEW Ride_Cost_Sum_PerMonth AS
SELECT
    EXTRACT(MONTH FROM ride_time) AS month_,
    ROUND(SUM(total_price), 2) AS average_ride_cost
FROM
    Ride
WHERE
    EXTRACT(YEAR FROM ride_time) = EXTRACT(YEAR FROM
CURRENT_DATE)
GROUP BY
    month_
ORDER BY
    month_;
```

Демонстрація: `SELECT * FROM Ride_Cost_Sum_PerMonth;`

	month_ numeric	average_ride_cost numeric
1	3	654959.83
2	4	3005197.21
3	5	3032897.40
4	6	3002680.08
5	7	3008897.45
6	8	2968145.42
7	9	3032925.47
8	10	3109683.26
9	11	3028961.79
10	12	2354528.25

Рис. 6.4.1. Виручка за кожен місяць поточного року

Опис: Представлення водія, що виконав найбільше поїздок за поточний рік

Призначення: Визначення водія, що виконав найбільшу кількість поїздок за рік для присвоєння йому премії.

```
CREATE OR REPLACE VIEW Driver_Max_Rides AS
SELECT
    d.last_name AS driver_last_name,
    COUNT(r.ride_id) AS total_rides
FROM
    Driver d
    JOIN Ride r ON d.driver_id = r.driver_id
WHERE EXTRACT(YEAR FROM r.ride_time) = EXTRACT(YEAR FROM
CURRENT_DATE)
GROUP BY
    d.last_name
ORDER BY
    total_rides DESC
LIMIT 1;
```

Демонстрація: `SELECT * FROM Driver_Max_Rides;`

	driver_last_name character varying (20) 🔒	total_rides bigint 🔒
1	Gershon	187

Рис. 6.4.2. Водій з найбільшою кількістю поїздок

Опис: Представлення водіїв, що не виконували замовлення більше 3 тижні та мають рейтинг нижче 4

Призначення: Визначення водіїв, що не виконують замовлення протягом останніх 3 тижнів і мають низький рейтинг для розгляду розірвання контракту.

```
CREATE OR REPLACE VIEW Firing_List AS
SELECT *
FROM Driver
WHERE driver_id NOT IN
(SELECT DISTINCT driver_id FROM Ride WHERE ride_time >= NOW() -
INTERVAL '3 week')
AND (driver_rating < 4 OR driver_rating IS NULL);
```

Демонстрація: `SELECT * FROM Firing_List;`

	driver_id integer	last_name character varying (20)	first_name character varying (20)	driving_license character varying (10)	driver_pnumber character	driver_rating numeric (2,1)
1	227	Benoy	Odette	3946492982	+380260350444	2.7

Рис. 6.4.3. Водії, що не виконували замовлення більше 3 тижнів

6.5 SQL-запити

Призначення: Визначення найдовших затримок водіїв для аналізу якості роботи водіїв та усунення можливих проблем, що викликають затримки.

Опис: Об'єднуємо таблиці водіїв, поїздок та замовлень, групуємо записи за прізвищем водія та його ID. Для кожного водія виводимо прізвище та максимальну різницю між часом початку поїздки та часом замовлення.

```
SELECT d.last_name,
       MAX(r.ride_time - date_time) AS arriving_time
FROM Driver d
JOIN Ride r ON r.driver_id=d.driver_id
JOIN Orders o ON o.order_id = r.order_id
GROUP BY d.driver_id, d.last_name;
```

Демонстрація:

	last_name character varying (20) 🔒	arriving_time interval 🔒
1	Shovel	00:25:00
2	McKea	00:25:00
3	Teeney	00:25:00
4	Clelland	00:25:00
5	Quinnette	00:25:00
6	Collete	00:25:00
7	McLeese	00:25:00
8	Casbourne	00:25:00
9	Leeds	00:25:00
10	Lofty	00:25:00
11	Lethibridge	00:25:00
12	Gligori	00:25:00

Рис. 6.5.1. Запит максимальної затримки водіїв

Призначення: Аналіз тенденцій замовлень користувачів: в якій середній відстані від домашньої адреси клієнт робить замовлення

Опис: У вибірці клієнтів та відповідних їм замовлень, що угруповуються за логіном клієнта. Виводимо логін клієнта та середнє значення відстані від точки відправлення у замовленні до точки домашньої адреси у кілометрах.

```
SELECT
    c.client_login,
    ROUND(CAST(AVG(COALESCE(ST_Distance(o.departure_point,
c.client_address), 0))/1000 AS numeric), 2) AS distance_from_home
FROM
    Client c
JOIN
    Orders o ON o.client_id = c.client_id
GROUP BY
    c.client_login;
```

Демонстрація:



	client_login character varying (20) 	distance_from_home numeric 
1	nstallionhd	21.83
2	lwhiteheadiy	19.22
3	nandraschh	25.11
4	jrodenburgbj	17.54
5	jroullierq9	26.03
6	jerrickera3	23.67
7	fblasettinz	13.89
8	lstych51	26.90
9	cstarlingm3	17.38
10	nmuffc5	12.86
11	ajurczak6a	23.36
12	obeaves5j	24.89
13	mtasselere5	24.51

Рис. 6.5.2. Вибірка середньої відстані між точкою відправлення та домашньою адресою клієнта

Призначення: Топ 5 водіїв у службі таксі, що отримують премію в кінці року.

Опис: Об'єднання водіїв та поїздок, що вони виконували, угруповане за ID водія та відсортоване за рейтингом водія, кількістю виконаних поїздок та загальною довжиною усіх поїздок за спаданням. Вибірка виводить 5 перших водіїв.

```
SELECT
    d.driver_rating, d.last_name,
    r.total_distance, r.total_rides
FROM Driver d
JOIN (
    SELECT
        driver_id,
        SUM(distance) AS total_distance,
        COUNT(*) AS total_rides
    FROM Ride
    GROUP BY driver_id
) r ON d.driver_id = r.driver_id
WHERE d.driver_rating IS NOT NULL
ORDER BY d.driver_rating DESC, total_rides DESC, total_distance
DESC
LIMIT 5;
```

Демонстрація:

	driver_rating numeric (2,1) 🔒	last_name character varying (20) 🔒	total_distance numeric 🔒	total_rides bigint 🔒
1	4.6	Renals	2139.59	96
2	4.6	Lunge	1660.64	80
3	4.5	Hardwidge	1863.79	87
4	4.2	Hessle	1718.12	79
5	3.8	Aires	2565.76	106

Рис. 6.5.3. Вибірка топ 5 водіїв

Призначення: Визначити уподобання кожного працівника щодо транспортних засобів, на яких виконуються замовлення. Такий аналіз допоможе створити комфортні умови роботи для водіїв.

Опис: Об'єднуємо записи про водіїв, транспорт, який вони використовували для виконання замовлень. Сортуюмо записи за прізвищем водія. Виводимо прізвище водія, марку, модель та номерний знак транспортного засобу, яким він користувався.

```
SELECT
    d.last_name,
    m.brand,m.model_name,
    t.plate_number
FROM Driver d
JOIN Ride r ON d.driver_id = r.driver_id
JOIN Transport t ON r.transport_id = t.transport_id
JOIN Model m ON t.model_id = m.model_id
ORDER BY d.last_name;
```

Демонстрація:

	last_name character varying (20) 🔒	brand character varying (20) 🔒	model_name character varying (30) 🔒	plate_number character varying (8) 🔒
1	Abercromby	Jaguar	X-Type	WG6007TK
2	Abercromby	Lincoln	Town Car	AZ4078NI
3	Abercromby	Honda	CR-V	IS0117NJ
4	Abercromby	Toyota	RAV4	QO2787MC
5	Abercromby	GMC	Vandura 3500	JG3819WJ
6	Abercromby	Hummer	H1	WA5547IW
7	Abercromby	Cadillac	Escalade EXT	JC7848JI
8	Abercromby	Dodge	Intrepid	PS8519QW
9	Abercromby	Buick	Park Avenue	RC0390WY
10	Abercromby	Chevrolet	SSR	IB7942RU
11	Abercromby	Buick	Park Avenue	VM8723LA
12	Abercromby	Volkswagen	Touareg	TC7304GY
13	Abercromby	Bentley	Azure	RQ9623FN
14	Abercromby	Volkswagen	Golf	UZ9563BF
15	Abercromby	Jeep	Compass	WD4531NF
16	Abercromby	Kia	Spectra	BI0568UK
17	Abercromby	Volkswagen	Golf	ZI6280QS
18	Abercromby	Ford	Econoline E150	XA5378UB
19	Abercromby	Jaguar	X-Type	OO4363KE

Рис. 6.5.4. Вибірка водіїв та використаного ними транспорту

Призначення: Визначити більш прибуткових клієнтів.

Опис: Об'єднання клієнтів, з їх замовленнями та відповідними поїздками з умовою, що вартість поїздки більше за середнє значення вартостей усіх поїздок. Виводимо логін та номер телефону клієнта.

```
SELECT DISTINCT c.client_login, c.client_pnumber
FROM Client c
JOIN Orders o ON o.client_id = c.client_id
JOIN Ride r ON o.order_id = r.order_id
WHERE r.total_price > (SELECT AVG(total_price) FROM Ride);
```

Демонстрація:

	client_login character varying (20) 🔒	client_pnumber character 🔒
1	cbloomcs	+380355973078
2	jthonasonpv	+380333138560
3	lcinelli7f	+380924026724
4	abredeeeag	+380193937558
5	ftrewnnrdqn	+380679117823
6	erielly8w	+380214505857
7	meliet34	+380349581868
8	rdullingham2	+380182729539
9	otwentyman9i	+380074021716
10	bbullentq4	+380188774702
11	bpetrello9n	+380444381165
12	mshevell3k	+380777993651
13	qjirasek2d	+380448255410

Рис. 6.5.5. Вибірка клієнтів, що замовляли поїздки з вартістю більше середньої

Призначення: Визначення найбільш використаного транспортного засобу для надання першочергового техогляду.

Опис: Вибірка транспорту та поїздок, виконаних на ньому, згрупованих за маркою, моделлю та номерним знаком та відсортованих за кількістю використань за спаданням. Виводиться лише перший запис - транспорт з максимальною кількістю використань.

```
SELECT m.brand,
       m.model_name,
       t.plate_number,
       COUNT(*) as usage_count
FROM Ride r
JOIN Transport t ON r.transport_id = t.transport_id
JOIN Model m ON t.model_id = m.model_id
WHERE EXTRACT(YEAR FROM ride_time) = EXTRACT(YEAR FROM
CURRENT_DATE)
GROUP BY m.brand,
         m.model_name,
         t.plate_number
ORDER BY usage_count DESC
LIMIT 1;
```

Демонстрація:

	brand character varying (20) 🔒	model_name character varying (30) 🔒	plate_number character varying (8) 🔒	usage_count bigint 🔒
1	Pontiac	Firebird	FB9447CW	312

Рис. 6.5.6. Транспортний засіб з найчастішим використанням

Призначення: Визначення найпопулярніших послуг за останні 3 місяці для планування знижок та маркетингу.

Опис: Об'єднуємо замовлення з послугами, що надані в рамках цих замовлень, де дата замовлення знаходиться в межах останніх 3 місяців. Групуємо за назвою послуги та сортуємо за кількістю використань за спаданням. Виводиться послуга та кількість використань для перших 3 рядків - 3 найбільш використані послуги.

```
SELECT st.service_name, COUNT(s.order_id) AS order_count
FROM Service s
JOIN Service_type st ON s.st_id = st.st_id
JOIN Orders o ON o.order_id = s.order_id
WHERE o.date_time >= CURRENT_DATE - INTERVAL '3 months'
GROUP BY st.service_name
ORDER BY order_count DESC
LIMIT 3;
```

Демонстрація:



	service_name character varying (20) 	order_count bigint 
1	Event Transportation	1000
2	Delivery	993
3	Cargo Box	990

Рис. 6.5.7. Топ 3 послуги за останні 3 місяці

Призначення: Визначення найновіших автомобілів у автопарку компанії для використання у святкових та люксових замовленнях.

Опис: Виводимо марку, модель та номерний знак транспорту, рік випуску якого був пізніше 2010. Записи відсортовано за роком випуску за спаданням.

```
SELECT m.brand,
       m.model_name,
       t.plate_number
FROM Model m
JOIN Transport t ON t.model_id = m.model_id
WHERE m.manufacturing_year >= 2010
ORDER BY m.manufacturing_year;
```

Демонстрація:

	brand character varying (20) 🔒	model_name character varying (30) 🔒	plate_number character varying (8) 🔒
1	Lamborghini	Murciélago	BK4247IS
2	Ford	Expedition	PH8170DC
3	Volvo	S60	LC2808MZ
4	Kia	Optima	JK3176OS
5	Cadillac	CTS-V	HN4291PJ
6	Toyota	Camry Hybrid	DK9406JN
7	Chevrolet	Aveo	LY5448YG
8	Cadillac	CTS-V	TB1704KF
9	Toyota	Camry Hybrid	KH7599AU
10	Toyota	Camry Hybrid	SK3312YB
11	BMW	7 Series	UR6311CD
12	Chevrolet	Aveo	IJ9228YK
13	Volvo	S60	AC7724YV
14	Volvo	S60	FC6426KU
15	Toyota	Camry Hybrid	EY6827RJ
16	Mercedes-Benz	GL-Class	HF6236PW
17	Lamborghini	Murciélago	XY0757BI

Рис. 6.5.8. Нові автомобілі у автопарку

Призначення: Визначення постійних клієнтів служби таксі для таргетованої реклами, розсилок та знижок.

Опис: Виведення логіну та номеру телефону клієнтів, у замовленнях яких було надано знижку постійного клієнта.

```
SELECT DISTINCT c.client_login,
               c.client_pnumber
FROM Orders o
JOIN Client c ON o.client_id = c.client_id
WHERE discount_id = 1;
```

Демонстрація:

	client_login character varying (20) 	client_pnumber character 
1	amccullaghdn	+380938915472
2	bhowoodk0	+380195087258
3	dsedworth1e	+380826872550
4	gpentycost1k	+380293170789
5	lfruser25	+380722652317
6	lisard148	+380111122233
7	mloselhk	+380618037201
8	mmaddick8s	+380187949770
9	oastill5h	+380304256420
10	Olligathor	+380123456789
11	pjedrasik2x	+380827373789
12	sfriel52	+380735717684
13	sisworth1m	+380928399841
14	swhether3t	+380669057751
15	yaroslaviynev	+380987654321

Рис. 6.5.9. Вибірка постійних клієнтів

Призначення: Визначення продуктивності водіїв під час літнього сезону

Опис: Вибираємо прізвище, ім'я та кількість поїздок, які вони виконали в часовому проміжку між 01.06.2023 та 31.08.2023, з групуванням за ім'ям та прізвищем маючи кількість поїздок більше 0

```
SELECT d.first_name, d.last_name, COUNT(*) as ride_count
FROM Ride r
JOIN Driver d ON d.driver_id = r.driver_id
WHERE r.ride_time BETWEEN '2023-06-01 00:01' AND '2023-08-31
23:59'
GROUP BY d.first_name, d.last_name
HAVING COUNT(*) > 0;
```

Демонстрація:

	first_name character varying (20)	last_name character varying (20)	ride_count bigint
1	Bliss	Greenhall	35
2	Donaugh	Beeden	25
3	Deerdre	Slater	30
4	Siana	Huby	29
5	Engracia	Franken	29
6	Katherina	Sybry	21
7	Sondra	Fawbert	24
8	Netti	Shallcroff	39
9	Mirabel	Bagley	28
10	Maridel	Fowlestone	31
11	Kati	Coey	32
12	Archer	Benito	25
13	Alano	Corradeschi	32
14	Netty	Moody	30
15	Far	Donnellan	26
16	Conni	Cromett	33
17	Silvano	Ure	31
18	Suzann	De Vere	24

Рис. 6.5.10. Вибірка водіїв, що працювали влітку

Призначення: Список водіїв, з якими потрібно сконтактувати щодо їх стану та готовності працювати.

Опис: Вибір водіїв, ID яких не присутні у результаті виконання підзапиту. Підзапит знаходить ID водіїв, які виконували замовлення, час яких знаходиться в межах останнього тижня.

```
SELECT *
FROM Driver
WHERE driver_id NOT IN
(SELECT DISTINCT driver_id FROM Ride WHERE ride_time >= NOW() -
INTERVAL '1 week');
```

Демонстрація:

	driver_id [PK] integer	last_name character varying (20)	first_name character varying (20)	driving_license character varying (10)	driver_pnumber character	driver_rating numeric (2,1)
252	182	Petyakov	Obidiah	8188313088	+380294695742	2.5
253	527	Natte	Becki	5569217733	+380640863944	2.5
254	568	Dorro	Jorey	8160072183	+380753517223	2.4
255	320	Shallcroff	Netti	8353193821	+380933811513	2.5
256	232	Gomme	Avrom	0942757060	+380978039988	2.4
257	419	Di Biaggi	Daniella	0956484508	+380578649344	2.5
258	123	Copestick	Hilly	5796625158	+380300646660	2.2
259	465	Streeton	Darwin	5398697449	+380113958468	2.5
260	583	MacNeillie	Shelley	4410975074	+380333875501	2.9
261	264	Stoakes	Kania	2568407692	+380562459666	2.3
262	328	Strooband	Kala	7623447493	+380057695661	2.5
263	52	Minnette	Brita	7360976708	+380024124196	2.3
264	339	Norquoy	Arvin	2505420530	+380900291122	2.5
265	530	Challinor	Bearnard	3610172238	+380692854632	2.5
266	525	Rushbury	Latrena	8963279573	+380304133349	2.3
267	47	Kinker	Burnaby	8166308589	+380357153766	2.6
268	311	Somerville	Edan	2834088254	+380734937169	2.6

Рис. 6.5.11. Водії, що не виконували замовлення більше тижня

Призначення: Визначення не актуальних тарифів у замовленнях клієнтів за останній місяць

Опис: Обираємо тарифи, яких не існує результату виконання підзапиту. Підзапит обирає замовлення з певним тарифом за попередній місяць

```
SELECT *
FROM Tariff t
WHERE NOT EXISTS (
    SELECT 1
    FROM Orders o
    WHERE o.tariff_id = t.tariff_id
        AND EXTRACT(MONTH FROM o.date_time) = EXTRACT(MONTH
FROM CURRENT_DATE)-1
        AND EXTRACT(YEAR FROM o.date_time) = EXTRACT(YEAR FROM
CURRENT_DATE)
);
```

Демонстрація:

	tariff_id [PK] integer	tariff_name character varying (20)	price numeric (10,2)
1	7	Premium	20.00
2	8	Luxury	30.00
3	9	Express	18.00
4	10	City	12.00
5	11	Night	14.00
6	12	Day	16.00
7	13	Weekend	22.00
8	14	Holiday	28.00
9	15	Family	32.00
10	16	Single	17.00
11	17	RoundTrip	26.00
12	18	Hourly	19.00
13	19	SuperSaver	8.00
14	20	UltraPremium	35.00
15	21	WeekdaySpecial	14.50
16	22	Adventurer	21.00
17	23	QuickRide	12.50

Рис. 6.5.12. Вибір не актуальних тарифів

Призначення: Визначення рейтингу користувачів за середньою вартістю їх поїздок для визначення користувачів, що платять більші суми за поїздки.

Опис: Вибір ID, логіну клієнту та середньої ціни поїздки, що вираховується за допомогою підзапиту. Середнє значення вираховується для записів з таблиці Ride, ID замовлення якого знаходяться у підзапиті, в якому обираються замовлення поточного клієнта. Результат основного запиту впорядковується за середньою ціною за спаданням.

```
SELECT client_id, client_login,
       ROUND(COALESCE((SELECT AVG(total_price) FROM Ride r WHERE
r.order_id IN(
SELECT o.order_id FROM Orders o WHERE o.client_id =
c.client_id)),0),2) AS avg_ride_cost
FROM Client c
ORDER BY avg_ride_cost DESC;
```

Демонстрація:

	client_id [PK] integer	client_login character varying (20)	avg_ride_cost numeric
1	4143	tkilpin3o	1304.70
2	3958	gsivellqb	1289.95
3	9530	zgimsonef	1200.50
4	2688	tgorvetteit	1132.50
5	1247	tseery6k	1132.00
6	6165	bnelm4a	1114.20
7	5268	lclynter75	1103.80
8	6004	gcotterillrl	1095.75
9	3781	dscownle	1094.90
10	1106	khughesdon2n	1076.50
11	4883	cantalffyo8	1059.90
12	9040	tpennigart	1054.50
13	3404	mcheltnamax	1051.75
14	5244	nhabercham6h	1037.50
15	3304	ksliney85	1029.48
16	2188	kredan4x	1026.81
17	7740	orobardleyk9	1003.50
18	5465	sarchanbaultcm	998.42

Рис. 6.5.13. Топ клієнтів за середньою вартістю замовлень

Призначення: Визначення водіїв, яким частіше залишаються відгуки

Опис: Визначаємо ID, прізвище та ім'я водія, та, за допомогою підзапиту, кількість залишених йому відгуків. У підзапиті знаходить кількість відгуків, у яких ID поїздки знаходиться у результаті підзапиту поїздок, ID водія яких співпадає з ID водія основного запиту

```
SELECT driver_id, last_name, first_name,
       (SELECT COUNT(*) FROM Feedback f WHERE f.ride_id IN (SELECT
r.ride_id FROM Ride r
       WHERE r.driver_id = d.driver_id)) AS feedback_count
FROM Driver d
ORDER BY feedback_count DESC;
```

Демонстрація:

	driver_id [PK] integer	last_name character varying (20)	first_name character varying (20)	feedback_count bigint
1	134	Renals	Ardyce	706
2	18	Lunge	Karrah	451
3	342	Hardwidge	Hermann	383
4	112	Hessle	Ophelia	298
5	682	Aires	Dorothy	236
6	181	Crevagh	Collie	137
7	366	Norville	Salim	129
8	145	Caughtry	Sheelagh	126
9	365	Maskelyne	Hermine	125
10	680	Farrall	Gabriel	125
11	683	Mackerness	Rosina	123
12	700	Alflat	Demetris	122
13	301	Thorold	Elset	122
14	441	MacSherry	Ramsay	121
15	554	Olifaunt	Gelya	120
16	663	Morillas	Jeromy	119
17	328	Strooband	Kala	118
18	396	Scutt	Constantino	118
19	70	Lofty	Alistair	118

Рис. 6.5.14. Рейтинг водіїв за кількістю відгуків

Призначення: Аналіз кількості користувачів, що замовляли послуги таксі на Новий рік та отримали знижку для планування кількості працівників, які працюватимуть на свято, та аналізу прибутковості та збитковості надання цієї знижки

Опис: Обрахунок кількості замовлень, в яких назва відповідної їм знижки - "New year"

```
SELECT COUNT(*) AS new_year_discounts
FROM Orders o
JOIN Discount d ON d.discount_id = o.discount_id
WHERE d.discount_name = 'New year';
```

Демонстрація:

	new_year_discounts
	bigint
1	12

Рис. 6.5.15. Кількість клієнтів, що зробили замовлення на Новий рік

Призначення: Визначення середнього рейтингу усіх аналізі загального іміджу компанії в очах користувачів та використанні у маркетингу

Опис: Вибір середнього значення усіх рейтингів водіїв служби таксі та кількості оцінених поїздок

```
SELECT
    ROUND(AVG(d.driver_rating), 1) AS average_rating,
    COUNT(DISTINCT r.ride_id) AS ratings
FROM
    Driver d
LEFT JOIN Ride r ON d.driver_id = r.driver_id
JOIN Feedback f ON r.ride_id = f.ride_id
WHERE
    d.driver_rating IS NOT NULL;
```

Демонстрація:

	average_rating	ratings
	numeric	bigint
1	2.6	38371

Рис. 6.5.16. Середній рейтинг усіх водіїв

Призначення: Визначення водія, що виконав найдовшу поїздку за попередній місяць, для присвоєння йому премії

Опис: Обирається водій, максимальна довжина поїздки для якого є максимальною з усіх за місяць. Максимальну довжину поїздки визначаємо у підзапиті, де обирається максимальне значення довжини для поїздок, що були проведені за попередній місяць, після чого знаходимо водія для якого довжина найдовшої поїздки буде дорівнювати результату підзапиту.

```
SELECT
    d.driver_id,
    d.last_name,
    MAX(r.distance) AS max_distance
FROM
    Ride r
JOIN
    Driver d ON r.driver_id = d.driver_id
WHERE
    EXTRACT(MONTH FROM r.ride_time) = EXTRACT(MONTH FROM
CURRENT_DATE)-1
    AND EXTRACT(YEAR FROM r.ride_time) = EXTRACT(YEAR FROM
CURRENT_DATE)
GROUP BY
    d.driver_id,
    d.last_name
HAVING MAX(r.distance) = (
    SELECT
        MAX(r_inner.distance)
    FROM
        Ride r_inner
    WHERE
        EXTRACT(MONTH FROM r_inner.ride_time) = EXTRACT(MONTH
FROM CURRENT_DATE)-1
        AND EXTRACT(YEAR FROM r_inner.ride_time) = EXTRACT(YEAR
FROM CURRENT_DATE));
```

Демонстрація:

	driver_id [PK] integer	last_name character varying (20)	max_distance numeric
1	219	Wilstead	55.29

Рис. 6.5.17. Водій, що виконав найдовшу поїздку за попередній місяць

Призначення: Аналіз у які дні тижня було більше замовлень та більше виручки.

Опис: Групуємо та сортуємо поїздки та замовлення за днем тижня. Виводимо день тижня (у PostgreSQL TIMESTAMP дні тижні позначаються числами від 0 до 6, де 0 - понеділок, 6 - неділя), кількість та сумарна вартість поїздок, що проводились у відповідний день тижня за поточний рік.

```
SELECT
    EXTRACT(DOW FROM o.date_time) AS day_of_week,
    COUNT(*) AS rides_on_weekdays,
    SUM(r.total_price) AS sum_on_weekdays
FROM Orders o
JOIN Ride r ON o.order_id = r.order_id
WHERE EXTRACT(YEAR FROM o.date_time) = EXTRACT(YEAR FROM
CURRENT_DATE)
GROUP BY
    day_of_week
ORDER BY
    day_of_week;
```

Демонстрація:

	day_of_week numeric	rides_on_weekdays bigint	sum_on_weekdays numeric
1	0	8754	3941565.26
2	1	8497	3824794.07
3	2	8314	3779800.58
4	3	8490	3871719.68
5	4	8780	3955189.90
6	5	8466	3888731.87
7	6	8699	3937074.80

Рис. 6.5.18. Кількість та сумарна вартість поїздок за кожен день тижня

Призначення: Проаналізувати кількість додаткових послуг для кожного клієнта для персоналізації пропозицій.

Опис: Вибірка з таблиць замовлень та типових послуг, що в них входять, згрупованих за ID клієнта та назвою послуги та з рахунком разів використання послуги клієнтом. Впорядковуються дані за ID клієнта та за кількістю використань послуги за спаданням.

```
SELECT
    c.client_id,
    st.service_name,
    COUNT(*) AS service_count
FROM
    Client c
    JOIN Orders o ON c.client_id = o.client_id
    JOIN Service s ON o.order_id = s.order_id
    JOIN Service_type st ON s.st_id = st.st_id
GROUP BY
    c.client_id,
    st.service_name
ORDER BY c.client_id, service_count DESC;
```

Демонстрація:

	client_id integer	service_name character varying (20)	service_count bigint
9	1	EV Upgrade	1
10	1	Inter-City Transfer	1
11	1	Towing Service	1
12	1	Business Upgrade	1
13	2	Grocery Delivery	1
14	2	Business Upgrade	1
15	2	Express Delivery	1
16	2	Airport Shuttle	1
17	2	Towing Service	1
18	2	VIP Escort	1
19	3	Parcel Courier	2
20	3	Bicycle Rack	1
21	3	Business Upgrade	1

Рис. 6.5.19. Вибірка використання послуг клієнтом

Призначення: Визначення кількості клієнтів, що замовляють поїздки пізніше 23:00 для аналізу попиту на пізні поїздки, та кількості водіїв, що працюють після 23:00, щоб з'ясувати чи достатня ця кількість водіїв та чи є можливість зменшити кількість водіїв чергових вночі.

Опис: Рахунок окремих клієнтів та водіїв, що подавали замовлення або виконували поїздки відповідно, де година з часу поїздки була більше дорівнює 23.

```
SELECT
  COUNT(DISTINCT o.client_id) AS late_clients,
  COUNT(DISTINCT r.driver_id) AS late_drivers
FROM
  Orders o
  LEFT JOIN Ride r ON o.order_id = r.order_id
WHERE
  EXTRACT(HOUR FROM r.ride_time) >= 23;
```

Демонстрація:



	late_clients bigint 	late_drivers bigint 
1	2158	679

Рис. 6.5.20. Вибірка кількості клієнтів та водіїв, що мають замовлення пізніше 23:00

6.6 Результати оптимізації

Для оптимізації запитів створюємо індекси для атрибутів, що з великою вірогідністю будуть унікальними. В нашій базі даних таких атрибути три: час замовлення, час початку поїздки та вартість поїздки після проведення випробувань і порівнянь часу виконання запиту без та з індексами з'ясувалось, що індекси покращують деякі запити для часу замовлення та часу початку поїздки, але час виконання збільшується для всіх створених запитів з індексом для вартості поїздки. Отже залишаємо лише перші 2 індекси.

Індекс часу замовлення

```
CREATE INDEX time_index ON Orders(date_time);
```

Запити до та після введення індексу

1.

```
EXPLAIN ANALYZE SELECT
    EXTRACT(DOW FROM o.date_time) AS day_of_week,
    COUNT(*) AS rides_on_weekdays,
    SUM(r.total_price) AS sum_on_weekdays
FROM Orders o
JOIN Ride r ON o.order_id = r.order_id
WHERE EXTRACT(YEAR FROM o.date_time) = EXTRACT(YEAR FROM CURRENT_DATE)
GROUP BY
    day_of_week
ORDER BY
    day_of_week;
```

Planning Time: 2.608 ms

Execution Time: 170.186 ms

Рис. 6.6.1. Запит 1 до оптимізації

Planning Time: 0.290 ms

Execution Time: 157.024 ms

Рис. 6.6.2. Запит 1 після оптимізації

2.

```
EXPLAIN ANALYZE SELECT st.service_name, COUNT(s.order_id) AS order_count
FROM Service s
JOIN Service_type st ON s.st_id = st.st_id
JOIN Orders o ON o.order_id = s.order_id
WHERE o.date_time >= CURRENT_DATE - INTERVAL '3 months'
```



```
GROUP BY st.service_name
ORDER BY order_count DESC
LIMIT 3;
```

Planning Time: 0.400 ms

Execution Time: 30.878 ms

Рис. 6.6.3. Запит 2 до оптимізації

Planning Time: 0.531 ms

Execution Time: 25.697 ms

Рис. 6.6.4. Запит 2 після оптимізації

Індекс часу початку поїздки

```
CREATE INDEX ride_time_index ON Ride(ride_time);
```

Запити до та після введення індексу

3.

```
EXPLAIN ANALYZE SELECT *
FROM Driver
WHERE driver_id NOT IN
(SELECT DISTINCT driver_id FROM Ride WHERE ride_time >= NOW() - INTERVAL
'1 week');
```

Planning Time: 0.213 ms

Execution Time: 16.916 ms

Рис. 6.6.5. Запит 3 до оптимізації

Planning Time: 0.201 ms

Execution Time: 1.113 ms

Рис. 6.6.6. Запит 3 після оптимізації

ВИСНОВКИ

У ході виконання курсової роботи було досліджено предметне середовище служби таксі та визначено його основні властивості, особливості, обмеження та потреби. Визначені чинники враховувались та були важливими при моделюванні майбутньої бази даних. База даних була створена на основі спроектованих моделей та передбачала усі зв'язки та обмеження, яких потребувало завдання. Були сформульовані та реалізовані запити, які відповідали потребам та очікуваним варіантам використання бази даних для аналізу роботи служби таксі.

При виконанні роботи було розроблено базу даних служби таксі, що передбачала створення замовлень клієнтами та опрацювання цих замовлень водіями, які ведуть звітність про поїздку. У роботі було автоматизовано багато процесів роботи з базою даних, зокрема обрахування рейтингу водія, відстані та вартості поїздки, призначення знижки за виконання умов. При роботі з базою даних було використано розширення PostGIS для роботи з географічними даними таблиці. При виконанні завдань було повністю покрито вимоги, що зазначались у розділі “Опис предметного середовища”.

Виконання курсової роботи дозволило мені покращити свої навички дослідження, аналізу та моделювання, отримати корисну практику в створенні повноцінних баз даних з нуля, навчитись виділяти основні потенційні потреби користувачів бази даних та передбачити їх у створених функціях, процедурах та представленнях, дізнатись як варто проводити дослідження на ефективність роботи запитів та як їх оптимізувати.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кацило Д. Uklon проклав дорогу в Україну техгігантам Uber і Bolt. Тепер компанія шукає свій блакитний океан – Forbes.ua.
URL:
<https://forbes.ua/company/u-poshukakh-blakitnogo-oceanu-02012021-814> (дата звернення: 25.12.2023).
2. Список міст України за покриттям застосунків таксі – Вікіпедія. *Vikimedia*. URL:
https://uk.wikipedia.org/wiki/Список_міст_України_за_покриттям_застосунків_таксі (дата звернення: 25.12.2023).
3. Uklon користуються 7,5 млн українців. 10 рішень від команди застосунків, які допомогли обігнати конкурентів. MC.today, Media for Creators. URL:
<https://mc.today/uk/uklon-koristuyutsya-7-5-mln-ukrayintsiv-10-rishen-vid-komandi-zastosunkiv-yaki-dopomogli-obignati-konkurentiv/> (дата звернення: 25.12.2023).
4. Тарасовський Ю. Український сервіс замовлення таксі OnTaxi з'явився у Празі – Forbes.ua.
URL:
<https://forbes.ua/news/ukrainskiy-servis-zamovlennya-taksi-ontaxi-pochav-pratsyuvati-u-prazi-29092023-16340> (дата звернення: 25.12.2023).
5. OnTaxi - Правила роботи водіїв. OnTaxi. URL: <https://ontaxi.com.ua/drivers-rule> (дата звернення: 26.12.2023).
6. Зелена книга "Ринок послуг таксі" / О. Гончарук та ін. 2018. 87 с. URL:
https://cdn.regulation.gov.ua/ec/b9/1e/88/regulation.gov.ua_Зелена%20книга%20Ринок%20послуг%20таксі.pdf (дата звернення: 25.12.2023).
7. PostgreSQL: documentation. PostgreSQL: The world's most advanced open source database. URL: <https://www.postgresql.org/docs/> (date of access: 27.12.2023).
8. Documentation. PostGIS. URL: <https://postgis.net/documentation/> (дата звернення: 22.12.2023).

ДОДАТКИ

Додаток 1 Тексти скриптів вставки записів

Функція для генерації випадкової точки місцезнаходження в межах міста Києва.

Точка задається між 30°20' та 31°00' сх. д, між 50°20' та 50°40' пн. ш. випадковим чином. Тобто в межах міста Києва. Функція повертає текст, який потім буде перетворено на тип GEOGRAPHY за допомогою функції ST_GeographyFromText(), тому точка задається за певним шаблоном:

```
'POINT(30.5632 50.4563)'
```

```
CREATE OR REPLACE FUNCTION Generate_Point()
RETURNS TEXT AS $$
BEGIN
    RETURN 'POINT(' || CAST(random() * (31.0 - 30.3) + 30.3 AS
NUMERIC(10, 4)) || ' ' ||
    CAST(random() * (50.6 - 50.3) + 50.3 AS NUMERIC(10, 4)) || ')';
END;
$$ LANGUAGE plpgsql;
```

Вставка даних у таблиці Discount, Service_type та Tariff

```
INSERT INTO Discount (discount_name, discount_size)
VALUES
('Regular client', 0.15), ('10th order', 0.05), ('New year',
0.24), ('Weekend Special', 0.10),
('Holiday Promo', 0.18), ('Early Bird Offer', 0.12), ('Summer
Savings', 0.15),
('Student Discount', 0.08), ('Family Package', 0.20), ('Senior
Citizen', 0.10),
('Birthday Bonus', 0.05), ('Referral Reward', 0.12), ('Spring
Fling', 0.15),
('Corporate Deal', 0.18), ('Valentines', 0.14),
('Back-to-School', 0.10),
('Frequent Rider', 0.15), ('Winter Wonderland', 0.20), ('Black
Friday', 0.25);
```

```

INSERT INTO Service_type (service_name, price)
VALUES
('Child Seat', 20.00), ('Towing Service', 50.00), ('Pet
Transportation', 30.00),
('Event Transportation', 50.00), ('Delivery', 20.00), ('Airport
Shuttle', 25.00),
('Luxury Car Upgrade', 20.00), ('Express Delivery', 12.00),
('Bicycle Rack', 8.00),
('Car Wash', 18.00), ('Night Surcharge', 15.00), ('VIP Escort',
40.00), ('EV Upgrade', 25.00),
('Grocery Delivery', 15.00), ('City Tour', 30.00), ('Inter-City
Transfer', 60.00),
('Business Upgrade', 35.00), ('Parcel Courier', 15.00),
('Multilingual Driver', 20.00),
('Cargo Box', 10.00);

```

```

INSERT INTO Tariff (tariff_name, price)
VALUES
('Economy', 10.00), ('Standard', 15.00), ('Comfort', 20.00),
('Business', 25.00),
('Universal', 18.00), ('Minibus', 30.00), ('Premium', 20.00),
('Luxury', 30.00),
('Express', 18.00), ('City', 12.00), ('Night', 14.00), ('Day',
16.00), ('Weekend', 22.00),
('Holiday', 28.00), ('Family', 32.00), ('Single', 17.00),
('RoundTrip', 26.00), ('Hourly', 19.00),
('SuperSaver', 8.00), ('UltraPremium', 35.00), ('WeekdaySpecial',
14.50),
('Adventurer', 21.00), ('QuickRide', 12.50);

```

Заповнення таблиці Client за допомогою звичайного запиту INSERT

```
INSERT INTO Client (client_login, client_password, client_address,
client_pnumber)
VALUES
    ('Olligathor', 'dR6*9K}z0Q',
ST_GeographyFromText(Generate_Point()), '+380123456789'),
    ('yaroslaviynev', 'oI0+pIX',
ST_GeographyFromText(Generate_Point()), '+380987654321'),
    ('Tcinn', 'cN3)sc(', ST_GeographyFromText(Generate_Point()),
'+380555555555'),
    ('lisard148', 'r07{Xw/3Y',
ST_GeographyFromText(Generate_Point()), '+380111122233'),
    ('irakum', 'tY7!q$kZk!Dw',
ST_GeographyFromText(Generate_Point()), '+380444433322'),
    ('renamed_user07665', 'gP6&U5T<x$',
ST_GeographyFromText(Generate_Point()), '+380999988877'),
    ('sky1609', 'iV86bbf', ST_GeographyFromText(Generate_Point()),
'+380777766655'),
    ('milrusy', 'pyA6+?Lk',
ST_GeographyFromText(Generate_Point()), '+380888877766'),
    ('DmitriyNizhnik', 'vM7=?@',
ST_GeographyFromText(Generate_Point()), '+38012348765'),
    ('Liliok9', 'xJ4!eaQC',
ST_GeographyFromText(Generate_Point()), '+38098761234');
```

Заповнення таблиці Client із згенерованого CSV файлу

```
COPY Client (client_login, client_password, client_pnumber)
FROM 'C:\taxi\client_generated.csv' WITH CSV HEADER DELIMITER ',';
```

Заповнення таблиці Driver із згенерованого CSV файлу

```
COPY Driver (last_name, first_name, driving_license,
driver_pnumber)
FROM 'C:\taxi\driver_generated.csv' WITH CSV HEADER DELIMITER ',';
```

Генерація випадкових даних до таблиці Orders

Атрибути `departure_point` та `destination_point` генеруються за попередньо визначеною функцією `Generate_Point()`; `payment` з рівномірною ймовірністю може набувати значень 'cash' та 'card'; `tariff_id` випадково вибирається з перших 6 значень у таблиці `Tariff`; `client_id` обирається випадково з усієї таблиці `Client`; `date_time` задається як випадкова дата в межах 9 до поточного часу. Створення 60000 записів.

```
INSERT INTO Orders (departure_point, destination_point, payment,
tariff_id, client_id, date_time)
SELECT
    ST_GeographyFromText(Generate_Point()),
    ST_GeographyFromText(Generate_Point()),
    CASE WHEN random() < 0.5 THEN 'cash'::payment_method ELSE
'card'::payment_method END,
    FLOOR(random() * 6) + 1,
    FLOOR(random() * (SELECT COUNT(*) FROM Client) + 1),
    date_trunc('minute', NOW() - random() * INTERVAL '9 month')
FROM generate_series(1, 60000);
```

Генерація випадкових даних до таблиці Service

Вибір випадкових записів з таблиць `Service_type` та `Orders`. Створення 50000 записів.

```
INSERT INTO Service (st_id, order_id)
SELECT
    FLOOR(random() * (SELECT COUNT(*) FROM Service_type) + 1),
    FLOOR(random() * (SELECT COUNT(*) FROM Orders) + 1)
FROM generate_series(1, 50000);
```

Заповнення таблиці Model із згенерованого CSV файлу

```
COPY Model (brand, model_name, manufacturing_year)
FROM 'C:\taxi\model_generated.csv' WITH CSV HEADER DELIMITER ',';
```

Заповнення таблиці Transport із згенерованого CSV файлу

```
COPY Transport (model_id, tariff_id, plate_number, color)
FROM 'C:\taxi\transport_generated.csv' WITH CSV HEADER DELIMITER
',';
```

Генерація записів у таблицю Ride

Нові створені поїздки послідовно поєднуються з замовленнями та їм призначається випадковий водій з таблиці Ride. Усі інші атрибути заповнюються за допомогою тригерів, функції яких зазначені після генерації.

```
INSERT INTO Ride (order_id, driver_id)
SELECT
    order_i,
    FLOOR(random() * (SELECT COUNT(*) FROM Driver) + 1)
FROM generate_series(1, 60000) AS order_i;
```

Функції

Розрахунок відстані між початковою та кінцевою точками замовлення за допомогою стандартної функції PostGIS ST_Distance().

```
CREATE OR REPLACE FUNCTION calculate_distance()
RETURNS TRIGGER AS $$
DECLARE
    calc_distance NUMERIC(10, 2);
BEGIN
    SELECT ST_Distance(o.departure_point,
        o.destination_point)/1000
        INTO calc_distance
    FROM Orders o
    WHERE o.order_id = NEW.order_id;
    UPDATE Ride
    SET distance = calc_distance
    WHERE ride_id = NEW.ride_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```


Розрахунок вартості з урахуванням відстані та тарифу поїздки, суми вартостей додаткових послуг та знижки.

```
CREATE OR REPLACE FUNCTION calculate_total_price()
RETURNS TRIGGER AS $$
DECLARE
    calc_distance NUMERIC(10, 2);
    tariff_price NUMERIC(10, 2);
    services_sum NUMERIC(10, 2);
    discount_s NUMERIC(3, 2);

BEGIN
    SELECT ST_Distance(o.departure_point,
o.destination_point)/1000 INTO calc_distance
    FROM Orders o
    WHERE o.order_id = NEW.order_id;

    SELECT t.price INTO tariff_price
    FROM Orders o
    JOIN Tariff t ON t.tariff_id = o.tariff_id
    WHERE o.order_id = NEW.order_id;

    SELECT COALESCE(SUM(st.price), 0) INTO services_sum
    FROM Orders o
    JOIN Service s ON o.order_id = s.order_id
    JOIN Service_type st ON s.st_id = st.st_id
    WHERE o.order_id = NEW.order_id;

    SELECT COALESCE(d.discount_size, 0) INTO discount_s
    FROM Orders o
    LEFT JOIN Discount d ON d.discount_id = o.discount_id
    WHERE o.order_id = NEW.order_id;

    UPDATE Ride
    SET total_price = (calc_distance*tariff_price +
services_sum)*(1-discount_s)
    WHERE ride_id = NEW.ride_id;

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

Встановлення випадкового часу початку поїздки в межах від 3 до 25 хвилин від замовлення

```
CREATE OR REPLACE FUNCTION calculate_time()
RETURNS TRIGGER AS $$
DECLARE
    order_date TIMESTAMP;
BEGIN
    SELECT o.date_time
    INTO order_date
    FROM Orders o
    WHERE o.order_id = NEW.order_id;
    UPDATE Ride
    SET ride_time = order_date + (FLOOR(random() * (25 - 3 + 1) +
3) || ' minutes')::INTERVAL
    WHERE ride_id = NEW.ride_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Вибір випадкового транспорту з відповідним тарифом

```
CREATE OR REPLACE FUNCTION choose_transport()
RETURNS TRIGGER AS $$
DECLARE
    order_tariff INTEGER;
    transport_num INTEGER;
BEGIN
    SELECT o.tariff_id
    INTO order_tariff
    FROM Orders o
    WHERE o.order_id = NEW.order_id;

    SELECT transport_id
    INTO transport_num
    FROM Transport
    WHERE tariff_id = order_tariff
    ORDER BY RANDOM() LIMIT 1;

    UPDATE Ride
    SET transport_id = transport_num
    WHERE ride_id = NEW.ride_id;

    RETURN NEW;
```

```
END;
$$ LANGUAGE plpgsql;
```

Заповнення таблиці Feedback із згенерованого CSV файлу

```
COPY Feedback (ride_id, rating, commentary)
FROM 'C:\taxi\feedback_generated.csv' WITH CSV HEADER DELIMITER
',';
SELECT COUNT(*) FROM Feedback;
```

Генерація записів у таблицю Feedback

ride_id обирається випадково з усіх записів поїздок; rating задається випадковим цілим значенням між 1 та 5; commentary не задається

```
INSERT INTO Feedback (ride_id, rating)
SELECT
    FLOOR(random() * (SELECT COUNT(*) FROM Ride) + 1),
    FLOOR(random() * (5 - 4) + 1)
FROM generate_series(1, 60000);
```