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 для підтримки діяльності сервісного центру МВС

в частині прийому екзаменів, водійських посвідчень ㅤ

|  |
| --- |
| Студента ㅤ2ㅤ курсу ㅤІП-22ㅤ групи |
| спеціальності 121 «Інженерія програмного забезпечення» |
| ㅤㅤㅤㅤㅤㅤ Нижника Д.С. ㅤㅤㅤㅤㅤ |
| (прізвище та ініціали) |
| Керівник старший викладач Марченко О.І. |
| (посада, вчене звання, науковий ступінь, прізвище та ініціали) |
| Національна шкала ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ |
| Кількість балів: ㅤㅤ Оцінка ECTS ㅤㅤ |
|  |
|  |

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

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

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

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 ㅤ2ㅤ Група ㅤІП-22ㅤ Семестр ㅤ3ㅤ

**ЗАВДАННЯ**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

ㅤㅤㅤㅤㅤㅤㅤㅤ ㅤ Нижнику Дмитру Сергійовичу ㅤㅤㅤㅤㅤㅤㅤㅤ

(прізвище, ім’я, по батькові)

1. Тема роботи База даних для підтримки діяльності сервісного центру МВС

в частині прийому екзаменів, водійських посвідчень ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ

керівник роботи старший викладач Марченко Олена Іванівнаㅤㅤ ㅤㅤㅤㅤ

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи ㅤ29.12.2023 ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ

3. Вихідні дані до роботи завдання на розробку бази даних для підтримкиㅤ

діяльності сервісного центру МВС в частині прийому екзаменів, водійських посвідченьㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ

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)

ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ

6. Дата видачі завдання ㅤ31.11.2023**ㅤㅤ ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ ㅤㅤ**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища | 03.12.2023 |  |
| 2 | Побудова ER-моделі | 03.12.2023 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 03.12.2023 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 15.12.2023 |  |
| 5 | Створення користувачів бази даних | 15.12.2023 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 15.12.2023 |  |
| 7 | Створення мовою SQL запитів | 21.12.2023 |  |
| 8 | Оптимізація роботи запитів | 28.12.2023 |  |
| 9 | Оформлення пояснювальної записки | 29.12.2023 |  |
| 10 | Захист курсової роботи | 30.12.2023 |  |

|  |
| --- |
| Нижник Д.С. |

**Студент ㅤㅤㅤㅤㅤㅤㅤ**

(підпис ) (прізвище та ініціали)

|  |
| --- |
| Марченко О.І. |

**Керівник роботи ㅤㅤㅤㅤㅤㅤㅤ**

(підпис ) (прізвище та ініціали)

ЗМІСТ

[1. ВСТУП 2](#_Toc154826582)

[2. ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА 3](#_Toc154826583)

[3. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ 6](#_Toc154826584)

[4. ПОСТАНОВКА ЗАВДАННЯ 7](#_Toc154826585)

[5. ПОБУДОВА ERD ТА РЕЛЯЦІЙНОЇ МОДЕЛІ 7](#_Toc154826586)

[6. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ 21](#_Toc154826587)

[7. РЕАЛІЗАЦІЯ РОЛЕЙ ТА КОРИСТУВАЧІВ 26](#_Toc154826588)

[7.1. Створення ролей 26](#_Toc154826589)

[7.2. Надання привілеїв ролі applicant 26](#_Toc154826590)

[7.3. Надання привілеїв ролі inspector 26](#_Toc154826591)

[7.4. Надання привілеїв ролі chief\_manager 27](#_Toc154826592)

[7.5. Надання привілеїв ролі document\_exam\_manager 27](#_Toc154826593)

[7.6. Створення користувачів 28](#_Toc154826594)

[8. РЕАЛІЗАЦІЯ ТРИГЕРІВ 30](#_Toc154826595)

[9. РЕАЛІЗАЦІЯ ПРЕДСТАВЛЕНЬ 45](#_Toc154826596)

[10. РЕАЛІЗАЦІЯ ЗАПИТІВ 51](#_Toc154826597)

[11. РЕАЛІЗАЦІЯ ФУНКЦІЙ ТА ПРОЦЕДУР 76](#_Toc154826598)

[12. ОПТИМІЗАЦІЯ 94](#_Toc154826599)

[13. ВИСНОВКИ 101](#_Toc154826600)

[14. СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 102](#_Toc154826601)

[ДОДАТОК А 103](#_Toc154826602)

[ДОДАТОК Б 116](#_Toc154826603)

# ВСТУП

Ми живемо у той час, коли майже у кожної сім’ї є автомобіль, ми живемо у час великих відстаней та швидкого темпу життя. З кожним роком кількість машин збільшується, збільшується кількість вантажоперевезень та робочих місць, пов’язаних із роботою на транспорті. У таких умовах кожна людина має вміти керувати машиною. Згідно із чинним законодавством, для керування транспортом треба мати посвідчення водія із зазначеною категорією транспорту.

В Україні для того, щоб отримати водійське посвідчення, треба звернутись до сервісного центру МВС, пройти теоретичний іспит на знання Правил дорожнього руху та пройти практичний іспит на вміння водити машину. Тільки після цього громадянин може отримати посвідчення водія. Категорії транспорту можна відкривати поступово.

Для ефективної організації роботи сервісного центру в області складання іспитів та отримання водійських посвідчень, необхідна база даних, що забезпечить цю роботу. База даних має великі переваги над іншими способами зберігання інформації, адже гарантує швидкий та зручний доступ, розподіл привілеїв, надійне зберігання інформації.

База даних має відповідати вимогам роботи сервісного центру, пришвидшувати її та автоматизувати її, зберігати дані про іспити та посвідчення, забезпечувати безпеку даних та захист від підробки, гарантувати зручну роботу для всіх можливих ролей.

Мета цієї курсової роботи – сформувати вимоги до бази даних, проаналізувати їх, проаналізувати предметне середовище, існуючі програмні продукти, розробити ER-модель, розробити базу даних та забезпечити її правильну роботу.

Всі файли, створені в ході виконання курсової роботи, зберігаються на репозиторії в GitHub (<https://github.com/Matematik15878/DB-Coursework>).

# ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Предметна область бази даних для підтримки діяльності сервісного центру МВС в частині прийому екзаменів, водійських посвідчень охоплює широкий спектр понять і механізмів, вимагає глибокого розуміння принципів роботи сервісного центру МВС.

Задачі, що постають перед такою базою даних, включають в себе збір, зберігання та обробку великого обсягу інформації про кандидатів, їхні результати екзаменів, стан видачі водійських посвідчень, а також інші аспекти, пов'язані з роботою СЦ і отриманням водійських посвідчень.

Аналіз цієї предметної області дозволить розкрити необхідність оптимізації інформаційних процесів для забезпечення ефективної роботи сервісного центру та задоволення потреб громадян в здачі екзаменів та отриманні водійських посвідчень.

Для детального аналізу області важливо зазначити основні принципи роботи сервісного центру МВС в частині прийому екзаменів, водійських посвідчень. Основними особами, що беруть участь в процесі отримання посвідчень, виступають, з одного боку, інспектори МВС і, з іншого боку, кандидати, тобто особи, що складають іспит.

Для отримання водійського посвідчення кандидатові треба здати 2 іспити: теоретичний і практичний. Теоретичний екзамен перевіряє знання чинних Правил дорожнього руху, розуміння принципів організації дорожнього руху, вміння теоретично зчитувати ситуацію на дорозі, розуміння своїх прав та обов’язків, розуміння прав та обов’язків інших учасників руху, знання принципів роботи правоохоронних органів на дорозі. Теоретичний екзамен – це комп’ютерний тест, що складається з 20 теоретичних питань. На здачу іспиту виділяється до 20 хвилин від початку складання. Структура кожного тесту може відрізнятися від інших, але загалом вона така: задається питання і надається ілюстрація дорожньої ситуації або графічний малюнок елементу дорожнього руху. Відповісти можна, поставивши прапорець навпроти правильної або декількох правильних відповідей (варіантів може бути від 2 до 5). Як тільки користувач завершить тест, система порахує кількість правильних відповідей і оголосить кандидату результат, а також зафіксує його у базі даних. Тест пройдено, якщо правильних відповідей не менше 18. За проведення тесту, дотримання порядку і уникнення нечесного проходження іспиту відповідає інспектор теоретичного іспиту. Він також відповідальний за дотримання порядку іспиту, тобто за запрошення кожного кандидата із черги на проходження тесту, оголошення інструкції та надання технічної допомоги у випадку непередбачуваних ситуацій.

Практичний іспит приймає інспектор практичного іспиту. Його задача – перевірити навички управління транспортним засобом, практичного зчитування дорожньої ситуації, розуміння знаків, принципів організації дорожнього руху. Окрім того, інспектор має забезпечувати безпеку іспиту, тому має право втрутитись у керування ТЗ. Практичний іспит починається і закінчується біля СЦ. Інспектор має право сам обирати маршрут і змінювати його під час іспиту. Екзамен може складатись не лише з руху за маршрутом, а і з виконання різних маневрів, в т.ч. паркування (паралельне або методом заїзду в гараж), розвороту в декілька прийомів тощо. Протягом іспиту кандидат може помилятися, що може вплинути на результат іспиту. Іспит буде завершено не успішно, якщо буде допущено більше 2 некритичних помилок або 1 критична. Критичними вважаються втручання інспектора в керування ТЗ або ігнорування його наказу. Якщо особа, що здає іспит, пройшла маршрут і зробила до 2 некритичних помилок включно, вона склала практичний іспит і має право отримати посвідчення водія. Незалежно від успіху екзамену, результати практичного іспиту фіксуються адміністратором СЦ у базі даних.

Теоретичний екзамен спільний для всіх кандидатів, а практичний залежить від обраної категорії транспортного засобу. Кандидат має складати практичний іспит на тій категорії, на якій навчався. У водійському посвідчені буде вказано коробку передач (якщо вона є), на якій було здано екзамен. Якщо це АКПП, тобто автоматична коробка передач, то водій може мати право управління лише ТЗ, що мають АКПП. Якщо іспит було здано на МКПП, тобто механічній коробці передач, то водій може мати право на управління будь-якою КПП транспорту своєї категорії.

Окрім того, особа, що здає іспит, не повинна мати протипоказань до управління ТЗ. Такі протипоказання можуть бути зафіксовані у медичній довідці, яку роблять лікарі поза межами сервісного центру.

Сервісний центр веде облік медичних карток, результатів практичних та теоретичних іспитів, а також облік отриманих посвідчень. Докладніше про ці обліки буде розказано пізніше.

Сервісний центр МВС має власні транспортні засоби для проведення іспитів. При проведенні практичного іспиту кандидат може здавати його або на транспортному засобі сервісного центру МВС, або на транспортному засобі автошколи. Важливо, що сервісний центр може мати транспорт різних категорій і призначення. Відповідність між призначенням транспорту і його категорією наведено в чинних Правилах дорожнього руху і буде наведено у наступних розділах.

# АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ

Єдиний існуючий програмний продукт, що реалізовує базу даних заданої теми – це база даних реальних сервісних центрів МВС. Звісно, доступ до неї обмежено, більша частина бази даних СЦ МВС не доступна звичайним громадянам, але принцип її роботи можна зрозуміти через взаємодію із цією системою.

Для отримання посвідчення водія, треба пройти практичний та теоретичний іспит, після чого в сервісному центрі можна забрати водійське посвідчення. Дані практичних іспитів зберігаються в базі даних Сервісного центру і на основі цих даних видаються посвідчення водія.

Для здачі теоретичного іспиту треба зареєструватись за паспортними даними, після чого менеджер зможе записати кандидата на здачу теоретичного іспиту в Сервісному центрі. Він проходить в форматі електронного тесту. Для здачі практичного іспиту потрібно здати теоретичний іспит. Практичний іспит проходить на машині сервісного центру або автошколи за присутності інспектору практичного іспиту.

Перевагою такої системи є її надійність та безпека. Дані користувачів надійно збережені у базі даних, захищені від сторонніх очей. Окрім того, Сервісний центр має вільний та зручний доступ до цих даних.

Недоліком цієї системи є неприємна бюрократизація процесу запису на іспити та проходження їх, відсутність зручного доступу до необхідної інформації для тих, хто проходить іспит, та інспекторів (наприклад, до інформації про транспорт, на якому здається іспит).

# ПОСТАНОВКА ЗАВДАННЯ

Задача даної курсової роботи – проаналізувати предметне середовище, описати його, проаналізувати вже існуючі продукти, деталізувати бізнес-правила та описати їх, сформувати ER-модель, розробити базу даних засобами мови SQL, розробити всі необхідні тригери, функції та процедури для правильної роботи бази даних, корисні запити для легшої взаємодії. База даних має повністю вирішувати завдання сервісного центру МВС та забезпечувати ефективну її роботу.

Окрім створення тригерів, збережних функцій та процедур, треба провести невелике дослідження щодо оптимізації роботи запитів в створеній базі даних. Виконання курсової роботи завершується висновками.

# ПОБУДОВА ERD ТА РЕЛЯЦІЙНОЇ МОДЕЛІ

Для початку слід сформувати бізнес-правила майбутньої системи:

1. Не можна записати кандидата на теоретичний іспит, якщо він має медичні протипоказання на керування транспортним засобом.
2. Не можна записати кандидата на практичний іспит, якщо він не здав теоретичний іспит на позитивну оцінку. Тобто, якщо маркер успіху теоретичного іспиту = FALSE, то кандидат не може здавати практичний іспит.
3. Не можна додати водійське посвідчення на деяку категорію із деякою коробкою передач, якщо кандидат не здав практичний іспит на цю категорію на цій коробці передач. Тобто, водійське посвідчення можна отримати тільки після практичного іспиту і тільки на ті характеристики транспорту, на яких він цей іспит здавав.
4. Якщо водій визнається медично не спроможним водити автомобіль, його водійське посвідчення має бути видалено.
5. Маркер успіху іспиту залежить від оцінки (для теоретичного іспиту) та кількості помилок (для практичного іспиту). Не можна ставити невідповідний маркер успіху.
6. Автомобілі, кандидати та інспектори мають бути зареєстрованими, інакше екзамен проводити не можна.

На основі опису предметного середовища можна сформувати такі необхідні сутності:

1. transport – сутність, що представляє зареєстрований для іспитів транспорт.
2. type\_of\_transport – сутність, що представляє собою доступні типи транспорту
3. category\_of\_transport – сутність, що представляє категорії зареєстрованого транспорту.
4. practical\_exam\_inspector – сутність, що представляє інструктора практичного іспиту.
5. theoretical\_exam\_inspector – сутність, що представляє існтруктора теоретичного іспиту.
6. applicant – сутність, що представляє кандидата на отримання водійського посвідчення.
7. log\_of\_practical\_exams – сутність, що представляє список практичних іспитів.
8. log\_of\_theoretical\_exams – сутність, що представляє список теоретичних екзаменів.
9. medical\_certificates – сутність, що містить медичні показники кандидата: групу крові, резус-фактор та маркер здатності керувати ТЗ.
10. drivers\_license – сутність, що представляє водійське посвідчення.

Окрім того, можна сформувати такі зв’язки:

1. practical\_exam\_inspector - log\_of\_practical\_exams – зв’язок «один до багатьох», адже кожний практичний іспит може проводити лише 1 екзаменатор, але кожний екзаменатор може проводити багато іспитів.
2. theoretical\_exam\_inspector - log\_of\_theoretical\_exams – зв’язок «один до багатьох», адже кожний теоретичний іспит може проводити лише один екзаменатор, але кожний екзаменатор може проводити багато іспитів.
3. applicant - log\_of\_practical\_exams – зв’язок «один до багатьох», адже кожний практичний іспит може проводитись лише для 1 кандидата, але кожний кандидат може проходити декілька іспитів.
4. applicant - log\_of\_theoretical\_exams – зв’язок «один до багатьох», адже кожний теоретичний іспит може проводитись лише для 1 кандидата, але кожний кандидат може проходити декілька іспитів.
5. applicant - medical\_certificates – зв’язок «один до одного», адже кожний кандидат має 1 запис медичних даних, і кожний запис медичних даних посилається на одного кандидата.
6. transport - log\_of\_practical\_exams – зв’язок «один до багатьох», адже кожний практичний іспит може проводитись лише на 1 транспортному засобі, але на кожному ТЗ можна проводити багато іспитів.
7. transport - type\_of\_transport – зв’язок «один до багатьох», адже кожний ТЗ може бути лише одного типу, але кожен тип можуть представляти багато транспортних засобів
8. transport - category\_of\_transport – зв’язок «один до багатьох», адже кожний ТЗ може бути лише однієї категорії, але кожну категорію транспорту можуть представляти багато транспортних засобів
9. drivers\_license - category\_of\_transport – зв’язок «багато до багатьох», адже кожне водійське посвідчення може мати декілька категорій, і кожна категорія може бути вписана у багатьох посвідченнях. Далі цей зв’язок доведеться спростити, додавши таблицю received\_categories.

Вкажемо користувачів, яких треба реалізувати, і їхні права.

1. document\_exam\_manager – менеджер документів та екзаменів. Відповідає за додання інформації про кандидатів, записи на іспити, водійські посвідчення та отримані категорії, медичні довідки. Тому має доступ до перегляду всіх таблиць в базі даних, а також на вставку і видалення даних в таблицях medical\_certificates, log\_of\_practical\_exams, received\_categories, drivers\_license, applicant, а також на зміну даних в полях amount\_of\_mistakes, success\_marker таблиці log\_of\_practical\_exams, phone\_number, amount\_of\_attempts таблиці applicant, exam\_grade, success\_marker таблиці log\_of\_theoretical\_exams, ability\_to\_drive\_marker таблиці medical\_certificates.
2. chief\_manager – головний менеджер Сервісного центру. Він відповідає за додання та видалення інспекторів, а також менеджмент транспорту. Тому він має право виводити всі дані із всіх таблиць, додавати та видаляти записи в theoretical\_exam\_inspector, practical\_exam\_inspector, transport, type\_of\_transport, category\_of\_transport, а також зміна поля phone\_number таблиць practical\_exam\_inspector та theoretical\_exam\_inspector.
3. applicant – кандидат на отримання посвідчення. Він має право переглядати важливу для себе інформацію, тобто має доступ до запитів типу SELECT до таблиць log\_of\_practical\_exams, log\_of\_theoretical\_exams, practical\_exam\_inspector, theoretical\_exam\_inspector, transport, applicant.
4. inspector – роль інспектора. Має всі права ролі applicant, а ще й свої власні – перегляд додаткової важливої інформації про кандидата. Тобто, він має доступ до запитів типу SELECT до таблиць received\_categories та drivers\_license.

Скрипти створення ролей наведено в розділі 7. Окрім створення ролей, для роботи із базою даних в цьому розділі створено користувачів із паролями.

Перед реалізацією бази даних засобами мови SQL, слід створити ER-діаграму бази даних. Побудована діаграма у лапковій моделі (Crow’s Foot) і зображена на рисунку 5.1.

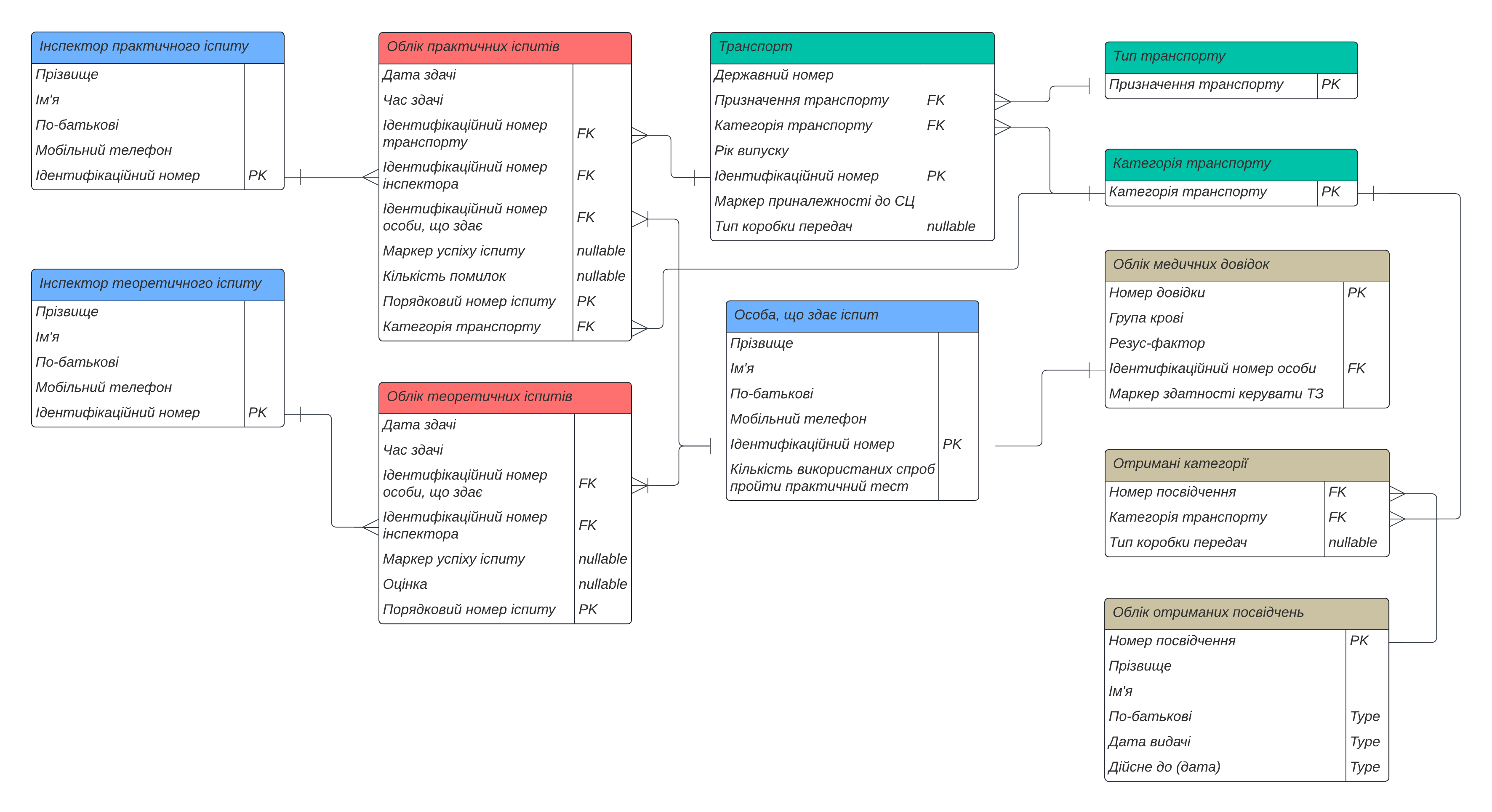


Рисунок 5.1 – ER діаграма до бази даних Сервісного центру МВС

Нижче наведено опис необхідних таблиць.

1. Інспектори. В таблицях «Інспектор практичного іспиту» та «Інспектор теоретичного іспиту» зберігаються дані про інспекторів.

Таблиця 5.1 – Інспектор практичного іспиту (practical\_exam\_inspector)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Прізвище | VARCHAR | 50 |  | Прізвище інспектора ПІ |
| Ім’я | VARCHAR | 50 |  | Ім’я інспектора ПІ |
| По-батькові | VARCHAR | 50 |  | По-батькові інспектора ПІ |
| Мобільний телефон | VARCHAR | 16 |  | Номер мобільного телефону інспектора ПІ |
| Ідентифікаційний номер | VARCHAR | 4 | PK | Ідентифікаційний номер інспектора, присвоюється СЦ |

Таблиця 5.2 – Інспектор теоретичного іспиту (theoretical\_exam\_inspector)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Прізвище | VARCHAR | 50 |  | Прізвище інспектора ТІ |
| Ім’я | VARCHAR | 50 |  | Ім’я інспектора ТІ |
| По-батькові | VARCHAR | 50 |  | По-батькові інспектора ТІ |
| Мобільний телефон | VARCHAR | 16 |  | Номер мобільного телефону інспектора ТІ |
| Ідентифікаційний номер | VARCHAR | 4 | PK | Ідентифікаційний номер інспектора, присвоюється СЦ |

1. Облік іспитів ведеться за допомогою таблиць «Облік практичних іспитів» та «Облік теоретичних іспитів».

Таблиця 5.3 – Облік практичних іспитів (log\_of\_practical\_exams)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Дата здачі | DATE |  |  | Дата проведення іспиту |
| Час здачі | TIME |  |  | Час проведення іспиту |
| Ідентифікаційний номер транспорту | VARCHAR | 4 | FK | Номер транспорту, на якому здається іспит |
| Ідентифікаційний номер інспектора | VARCHAR | 4 | FK | Ідентифікаційний номер інспектора, що приймає |
| Ідентифікаційний номер кандидата | VARCHAR | 4 | FK | Ідентифікаційний номер особи, що здає іспит |
| Маркер успіху іспиту | BOOL, nullable |  |  | Іспит складено успішно/не успішно |
| Кількість помилок | INTEGER, nullable |  |  | Кількість помилок |
| Порядковий номер іспиту | VARCHAR | 9 | PK | Порядковий номер у черзі |
| Категорія транспорту | VARCHAR | 2 | FK | Категорія, на яку здавала |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Дата здачі | DATE |  |  | Дата проведення іспиту |
| Час здачі | TIME |  |  | Час проведення іспиту |
| Ідентифікаційний номер інспектора | VARCHAR | 4 | FK | Ідентифікаційний номер інспектора, що приймає |
| Ідентифікаційний номер кандидата | VARCHAR | 4 | FK | Ідентифікаційний номер особи, що здає іспит |
| Маркер успіху іспиту | BOOL, nullable |  |  | Іспит складено успішно/не успішно |
| Оцінка | INTEGER, nullable |  |  | Кількість помилок |
| Порядковий номер іспиту | VARCHAR | 9 | PK | Порядковий номер у черзі |

Таблиця 5.4 – Облік теоретичних іспитів (log\_of\_theoretical\_exams)

Важливо, що спочатку створюється запис іспиту у таблиці, і лише після завершення в неї записується результат іспиту і оцінка.

1. Облік транспортних засобів зберігається у таблиці «Транспорт».

Таблиця 5.5 – Транспорт (transport)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Державний номер | VARCHAR | 8 |  | Державний номер ТЗ |
| Призначення транспорту | VARCHAR | 20 | FK | Призначення транспорту, зберігається у табличці «Тип транспорту» |
| Категорія транспорту | VARCHAR | 2 | FK | Категорія транспорту, зберігається у табличці «Категорія транспорту» |
| Рік випуску | VARCHAR | 4 |  | Рік випуску транспорту |
| Ідентифікаційний номер | VARCHAR | 4 | PK | Ідентифікаційний номер транспорту, надається СЦ |
| Маркер приналежності до СЦ | BOOL |  |  | True, якщо ТЗ належить СЦ і False, якщо ТЗ належить автошколі |
| Тип КПП | VARCHAR, nullable | 2 |  | КПП, на якій здавався іспит. Якщо у ТЗ нема КПП, то NULL. KP або AT |

Таблиця 5.6 – Тип транспорту (type\_of\_transport)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Призначення транспорту | VARCHAR | 20 | PK | Призначення транспорту. Може бути автобус, вантажівка, легковий автомобіль тощо. |

Таблиця 5.7 – Категорія транспорту (category\_of\_transport)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Категорія транспорту | VARCHAR | 2 | PK | Категорія транспорту. Прописано в ПДР |

У ПДР наведено такі категорії транспорту:

Категорія А - мотоцикли, в тому числі з боковим причепом, двигун яких понад 50 куб. см або електродвигун потужністю 4 кВт і більше.

Категорія А1 - мопеди, моторолери та інші 2-х колісні ТЗ: робочий об’єм до 50 куб. см або потужність електродвигуна до 4 кВт.

Категорія В – ТЗ з максимальною масою до 3.5 тон і з кількістю сидячих місць до 9 (легкові автомобілі, легкі вантажівки).

Категорія В1 – ТЗ з 4 колесами максимальною масою до 400 кг (квадроцикли).

Категорія С – вантажні автомобілі з максимальною масою більше 7.5 тон.

Категорія С1 – вантажівки з максимальною масою від 3.5 до 7.5 тон.

Категорія D – автобуси із кількістю сидячих місць більше 16 + місце водія.

Категорія D1 – мікроавтобуси із кількістю сидячих місць 16 + місце водія.

Категорія Т – трамваї та тролейбуси.

1. Інформація про кандидатів зберігається в табличці «Особа, що здає іспит»

Таблиця 5.8 – Особа, що здає іспит (applicant)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Прізвище | VARCHAR | 50 |  | Прізвище кандидата |
| Ім'я | VARCHAR | 50 |  | Ім’я кандидата |
| По-батькові | VARCHAR | 50 |  | По-батькові кандидата |
| Мобільний телефон | VARCHAR | 16 |  | Мобільний номер телефону кандидата |
| Ідентифікаційний номер | VARCHAR | 4 | PK | Ідентифікаційний номер кандидата, присвоюється СЦ |
| Кількість використаних спроб пройти практичний тест | INTEGER |  |  | Кількість використаних спроб скласти практичний іспит |

1. Інша інформація про кандидата зберігається у табличках «Облік медичних довідок», «Отримані категорії» та «Облік отриманих посвідчень»

Таблиця 5.9 – Облік медичних довідок (medical\_certificates)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Номер довідки | VARCHAR | 6 | PK | Номер медичної довідки |
| Група крові | INTEGER |  |  | Група крові людини. 1, 2, 3 або 4 |
| Резус-фактор | VARCHAR | 1 |  | Резус-фактор. + або - |
| Ідентифікаційний номер особи | VARCHAR | 4 | FK | Ідентифікаційний номер кандидата, присвоюється СЦ |
| Маркер здатності керувати ТЗ | BOOL |  |  | Маркер, чи може людина керувати ТЗ. True або False |

Таблиця 5.10 – Отримані категорії (received\_categories)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Номер посвідчення | VARCHAR | 9 | FK | Номер посвідчення водія,  3 букви і 6 цифр |
| Категорія транспорту | VARCHAR | 2 | FK | Категорія транспорту, яким може керувати людина |
| Тип коробки передач | VARCHAR, nullable | 2 |  | КПП, на якій здавався іспит. Якщо у ТЗ нема КПП, то NULL. KP або AT |

Таблиця 5.11 – Облік отриманих посвідчень (drivers\_license)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я колонки** | **Тип** | **Розмір** | **Ключ** | **Опис** |
| Номер посвідчення | VARCHAR | 9 | PK | Номер посвідчення водія,  3 букви і 6 цифр |
| Прізвище | VARCHAR | 50 |  | Прізвище водія |
| Ім'я | VARCHAR | 50 |  | Ім’я водія |
| По-батькові | VARCHAR | 50 |  | По-батькові водія |
| Дата отримання | DATE |  |  | Дата отримання посвідчення |
| Дійсне до | DATE |  |  | Дата, до якого це посвідчення дійсне |

Для реалізації бази даних обрано один із діалектів мови програмування SQL - PostgreSQL. Детальніше про реалізацію бази даних розказано у наступному розділі.

PostgreSQL є потужною та розширюваною системою управління базами даних, яка має кілька переваг порівняно з іншими діалектами СУБД, такими як MySQL, SQLite або Microsoft SQL Server.

По-перше, Postgre підтримує багато розширень. Це можуть бути запити, функції та процедури. Наприклад, даний діалект підтримує такі важливі розширення, як PostGIS (для роботи із географічними даними), hstore (для зберігання неканонічних даних у вигляді пар ключ-значення), pg\_trgm (для роботи з текстовими індексами) тощо. Більше того, ці розширення можна створювати самому, що може сильно розширити функціональність програми.

По-друге, Postgre має велику та дуже активну спільноту розробників, що гарантує швидке виявлення та виправлення помилок, а також розвиток нового функціоналу. Окрім того, велика спільнота розробників означає великий попит на програми, написані на цьому діалекті.

Код для створення бази даних для підтримки Сервісного центру МВС в області проведення екзаменів та видачі водійських посвідчень наведено в наступному розділі. На рисунку 5.2 показано ERD, створену засобами СУБД PostgreSQL.

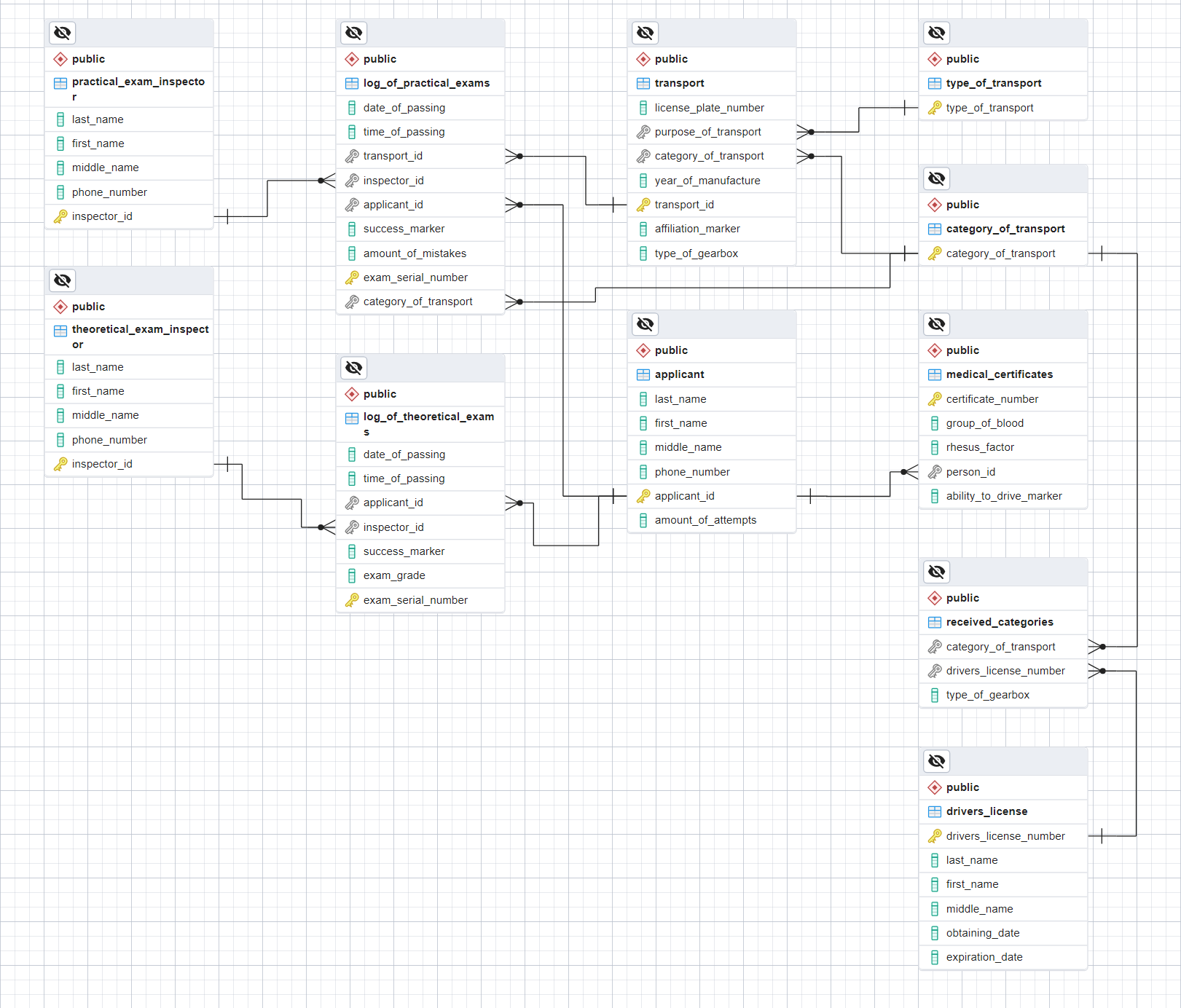


Рисунок 5.2 – Створена ERD засобами СУБД PostgreSQL

# РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

Результатом проєктування бази даних є її реалізація засобами мови програмування SQL. Нижче наведено скрипти, що забезпечують створення бази даних service\_center\_mia та реалізацію її таблиць.

Окрім того, в створених таблицях використовуються генератори (поля з типом даних SERIAL). Генератори використовуються у таких полях: exam\_serial\_number (таблиця log\_of\_practical\_exams) та exam\_serial\_number (таблиця log\_of\_theoretical\_exams).

CREATE DATABASE service\_center\_mia;

\c service\_center\_mia

CREATE TABLE type\_of\_transport (

type\_of\_transport VARCHAR(50) PRIMARY KEY NOT NULL

);

CREATE TABLE category\_of\_transport (

category\_of\_transport VARCHAR(2) PRIMARY KEY NOT NULL

);

CREATE TABLE transport (

license\_plate\_number VARCHAR(8),

purpose\_of\_transport VARCHAR(50) REFERENCES type\_of\_transport (type\_of\_transport) ON DELETE CASCADE NOT NULL,

category\_of\_transport VARCHAR(3) REFERENCES category\_of\_transport (category\_of\_transport) ON DELETE CASCADE NOT NULL,

year\_of\_manufacture VARCHAR(4) NOT NULL,

transport\_id VARCHAR(4) PRIMARY KEY NOT NULL,

affiliation\_marker BOOLEAN NOT NULL,

type\_of\_gearbox varchar(2));

CREATE TABLE practical\_exam\_inspector (

last\_name VARCHAR(50) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

middle\_name VARCHAR(50) NOT NULL,

phone\_number VARCHAR(16) NOT NULL,

inspector\_id VARCHAR(4) PRIMARY KEY NOT NULL

);

CREATE TABLE theoretical\_exam\_inspector (

last\_name VARCHAR(50) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

middle\_name VARCHAR(50) NOT NULL,

phone\_number VARCHAR(16) NOT NULL,

inspector\_id VARCHAR(4) PRIMARY KEY NOT NULL

);

CREATE TABLE applicant (

last\_name VARCHAR(50) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

middle\_name VARCHAR(50) NOT NULL,

phone\_number VARCHAR(16) NOT NULL,

applicant\_id VARCHAR(10) PRIMARY KEY NOT NULL,

amount\_of\_attempts INTEGER NOT NULL

);

CREATE TABLE log\_of\_practical\_exams (

date\_of\_passing DATE CHECK (date\_of\_passing >= '1900-12-24') NOT NULL,

time\_of\_passing TIME NOT NULL,

transport\_id VARCHAR(4) REFERENCES transport (transport\_id) ON DELETE CASCADE NOT NULL,

inspector\_id VARCHAR(4) REFERENCES practical\_exam\_inspector (inspector\_id) ON DELETE CASCADE NOT NULL,

applicant\_id VARCHAR(10) REFERENCES applicant (applicant\_id) ON DELETE CASCADE NOT NULL,

success\_marker BOOLEAN,

amount\_of\_mistakes SMALLINT,

exam\_serial\_number SERIAL PRIMARY KEY NOT NULL,

category\_of\_transport VARCHAR(3) REFERENCES category\_of\_transport (category\_of\_transport) ON DELETE CASCADE NOT NULL);

CREATE TABLE log\_of\_theoretical\_exams (

date\_of\_passing DATE CHECK (date\_of\_passing >= '1900-12-24') NOT NULL,

time\_of\_passing TIME NOT NULL,

applicant\_id VARCHAR(10) REFERENCES applicant (applicant\_id) ON DELETE CASCADE NOT NULL,

inspector\_id VARCHAR(4) REFERENCES theoretical\_exam\_inspector (inspector\_id) ON DELETE CASCADE NOT NULL,

success\_marker BOOLEAN,

exam\_grade SMALLINT,

exam\_serial\_number SERIAL PRIMARY KEY NOT NULL

);

CREATE TABLE medical\_certificates (

certificate\_number VARCHAR(6) PRIMARY KEY NOT NULL,

group\_of\_blood INTEGER NOT NULL,

rhesus\_factor VARCHAR(1) NOT NULL,

person\_id VARCHAR(10) REFERENCES applicant(applicant\_id) ON DELETE CASCADE NOT NULL,

ability\_to\_drive\_marker BOOLEAN NOT NULL

);

CREATE TABLE drivers\_license (

drivers\_license\_number VARCHAR(9) PRIMARY KEY NOT NULL,

last\_name VARCHAR(50) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

middle\_name VARCHAR(50) NOT NULL,

obtaining\_date DATE CHECK (obtaining\_date >= '1900-12-24') NOT NULL,

expiration\_date DATE CHECK (expiration\_date >= '1900-12-24') NOT NULL

);

CREATE TABLE received\_categories (

category\_of\_transport VARCHAR(2) REFERENCES category\_of\_transport (category\_of\_transport) NOT NULL,

drivers\_license\_number VARCHAR(9) REFERENCES drivers\_license (drivers\_license\_number) NOT NULL,

type\_of\_gearbox varchar(2),

CONSTRAINT uq\_recieved\_categorie UNIQUE (category\_of\_transport, drivers\_license\_number)

);

Код для заповнення бази даних даними наведено у Додатку А та Додатку Б.

Для перевірки правильності створення бази даних service\_center\_mia, можна скористатися командою \l у SQL Shell. Результат на рисунку 6.1.

Для перевірки правильності створення таблиць БД service\_center\_mia, слід скористатися командою \dt у SQL Shell. Результат на рисунку 6.2.



Рисунок 6.1 – Створена БД у SQL Shell

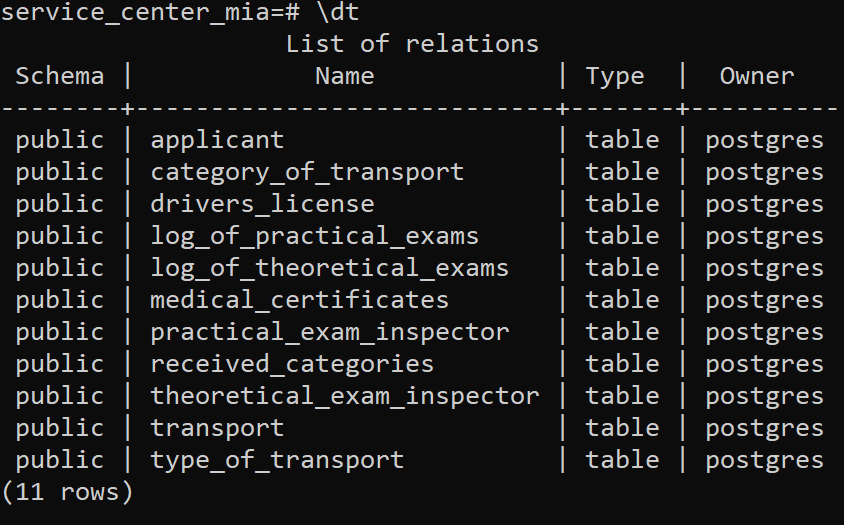


Рисунок 6.2 – Створені таблиці БД service\_center\_mia

# РЕАЛІЗАЦІЯ РОЛЕЙ ТА КОРИСТУВАЧІВ

Для забезпечення безпеки бази даних, встановлення обмежень на доступ до даних та ідентифікації користувачів створюються ролі та користувачі. Опис ролей та їх прав наведено у 5 розділі пояснювальної записки.

Нижче наведено код реалізації ролей та користувачів для бази даних service\_center\_mia та надання їм необхідних привілеїв.

## 7.1. Створення ролей

CREATE ROLE document\_exam\_manager;

CREATE ROLE chief\_manager;

CREATE ROLE applicant;

CREATE ROLE inspector;

## 7.2. Надання привілеїв ролі applicant

GRANT SELECT ON log\_of\_practical\_exams TO applicant;

GRANT SELECT ON log\_of\_theoretical\_exams TO applicant;

GRANT SELECT ON practical\_exam\_inspector TO applicant;

GRANT SELECT ON theoretical\_exam\_inspector TO applicant;

GRANT SELECT ON transport TO applicant;

GRANT SELECT ON applicant TO applicant;

## 7.3. Надання привілеїв ролі inspector

GRANT applicant TO inspector;

GRANT SELECT ON received\_categories TO inspector;

GRANT SELECT ON drivers\_license TO inspector;

## 7.4. Надання привілеїв ролі chief\_manager

GRANT SELECT ON ALL TABLES IN SCHEMA public TO chief\_manager;

GRANT INSERT, DELETE ON practical\_exam\_inspector, theoretical\_exam\_inspector TO chief\_manager;

GRANT INSERT, DELETE ON transport, type\_of\_transport, category\_of\_transport TO chief\_manager;

GRANT UPDATE (phone\_number) ON practical\_exam\_inspector, theoretical\_exam\_inspector TO chief\_manager;

## 7.5. Надання привілеїв ролі document\_exam\_manager

GRANT SELECT ON ALL TABLES IN SCHEMA public TO document\_exam\_manager;

GRANT INSERT, DELETE ON medical\_certificates TO document\_exam\_manager;

GRANT INSERT, DELETE ON log\_of\_practical\_exams, log\_of\_theoretical\_exams TO document\_exam\_manager;

GRANT INSERT, DELETE ON received\_categories TO document\_exam\_manager;

GRANT INSERT, DELETE ON drivers\_license TO document\_exam\_manager;

GRANT INSERT, DELETE ON applicant TO document\_exam\_manager;

GRANT USAGE, SELECT ON SEQUENCE log\_of\_practical\_exams\_exam\_serial\_number\_seq TO document\_exam\_manager;

GRANT USAGE, SELECT ON SEQUENCE log\_of\_theoretical\_exams\_exam\_serial\_number\_seq TO document\_exam\_manager;

GRANT UPDATE (amount\_of\_mistakes, success\_marker) ON log\_of\_practical\_exams TO document\_exam\_manager;

GRANT UPDATE (phone\_number, amount\_of\_attempts) ON applicant TO document\_exam\_manager;

GRANT UPDATE (exam\_grade, success\_marker) ON log\_of\_theoretical\_exams TO document\_exam\_manager;

GRANT UPDATE (ability\_to\_drive\_marker) ON medical\_certificates TO document\_exam\_manager;

## 7.6. Створення користувачів

CREATE USER chief\_manager\_user WITH PASSWORD 'admin';

GRANT chief\_manager TO chief\_manager\_user;

CREATE USER document\_exam\_manager\_user WITH PASSWORD 'password';

GRANT document\_exam\_manager TO document\_exam\_manager\_user;

CREATE USER inspector\_user WITH PASSWORD '244466666';

GRANT inspector TO inspector\_user;

CREATE USER applicant\_user WITH PASSWORD '987654321';

GRANT applicant TO applicant\_user;

Успішність створення ролей та користувачів можна перевірити за допомогою запитів. Перевірка ролей показана на рисунку 7.1, перевірка користувачів – на рисунку 7.2.

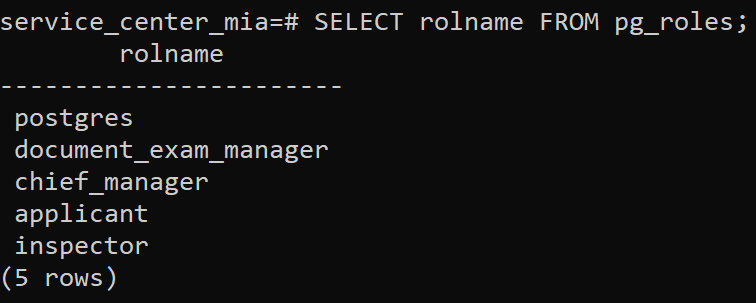


Рисунок 7.1 – Створені ролі бази даних service\_center\_mia

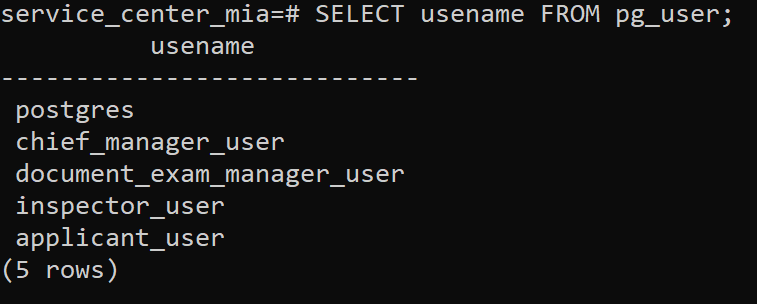


Рисунок 7.2 – Створені користувачі бази даних service\_center\_mia

# РЕАЛІЗАЦІЯ ТРИГЕРІВ

У базах даних тригери потрібні для автоматизації бізнес-логіки, забезпечення цілісності даних, забезпечення безпеки, виконання деяких операцій перед або після операцій з даними.

Нижче наведено реалізовані тригери, їхній словесний опис та суть, а також приклади роботи.

8.1. Тригер, що реагує на вставку практичного іспиту.

Цей тригер реалізує таке обмеження, як обов’язковість теоретичного іспиту перед складанням практичного. Іншими словами, неможливо записати кандидата на здачу практичного іспиту, якщо він не пройшов теоретичний. Таку логіку виконують тригер check\_theoretical\_exam\_trigger і тригерна функція check\_theoretical\_exam\_success. Нижче наведено їхній код.

CREATE OR REPLACE FUNCTION check\_theoretical\_exam\_success()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS (

SELECT 1 FROM log\_of\_theoretical\_exams

WHERE applicant\_id = NEW.applicant\_id AND success\_marker = TRUE

)

THEN RAISE EXCEPTION 'Applicant has not passed the theoretical exam! The insertion was aborted!';

ELSE RETURN NEW;

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_theoretical\_exam\_trigger

BEFORE INSERT ON log\_of\_practical\_exams

FOR EACH ROW

EXECUTE FUNCTION check\_theoretical\_exam\_success();

На рисунку 8.1. показано роботу тригера при вставці практичного іспиту для кандидата, що не пройшов теоретичний іспит. На рисунку 8.2. показано вставку практичного іспиту для кандидата, що пройшов теоретичний іспит.

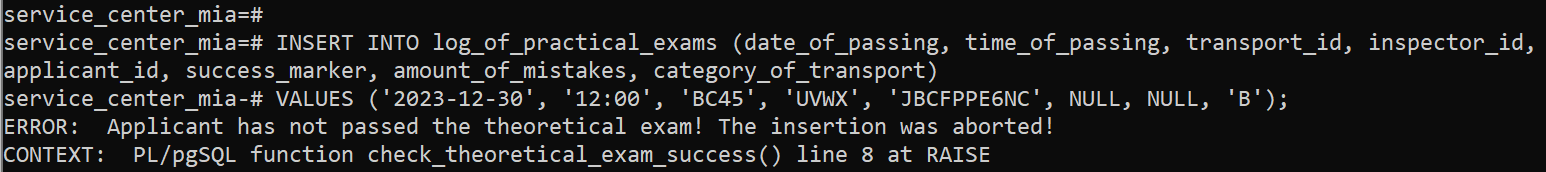


Рисунок 8.1 – Вставка практичного іспиту кандидата, що не склав теорію

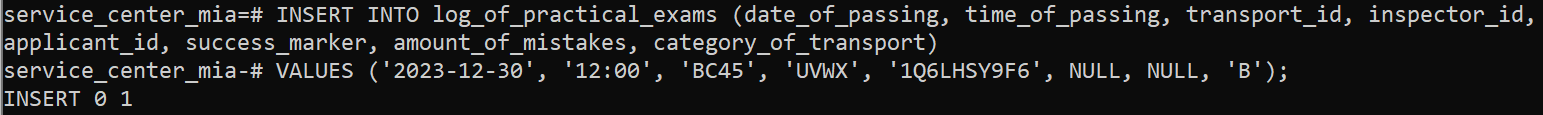


Рисунок 8.2 – Вставка практичного іспиту кандидата, що склав теорію

8.2. Тригер, що реагує на вставку теоретичного іспиту.

Цей тригер реалізує таке обмеження, як обов’язковість допуску до керування транспортним засобом перед складанням теоретичного іспиту. Іншими словами, неможливо записати кандидата на здачу теоретичного іспиту, якщо його не допущено до керуванням ТЗ при оформленні медичної довідки. Таку логіку виконують тригер check\_ability\_to\_drive\_trigger і тригерна функція check\_ability\_to\_drive. Нижче наведено їхній код.

CREATE OR REPLACE FUNCTION check\_ability\_to\_drive()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS (

SELECT 1

FROM medical\_certificates

WHERE person\_id = NEW.applicant\_id AND ability\_to\_drive\_marker = TRUE

)

THEN RAISE EXCEPTION 'Applicant cannot drive a vehicle or does not have a medical certificate! The insertion was aborted!';

ELSE RETURN NEW;

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_ability\_to\_drive\_trigger

BEFORE INSERT ON log\_of\_theoretical\_exams

FOR EACH ROW

EXECUTE FUNCTION check\_ability\_to\_drive();

На рисунку 8.3. показано роботу тригера при вставці теоретичного іспиту для кандидата, що не має права на керування транспортним засобом. На рисунку 8.4. показано вставку теоретичного іспиту для кандидата, що таке право має.

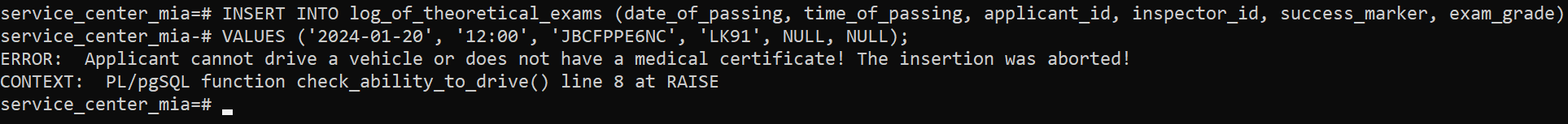


Рисунок 8.3 – Вставка теоретичного іспиту кандидата,

що не має права керувати ТЗ

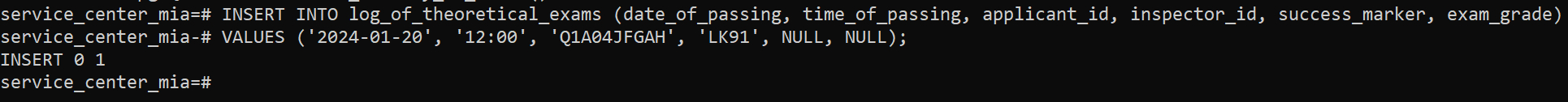


Рисунок 8.4 – Вставка теоретичного іспиту кандидата,

що має право керувати ТЗ

8.3. Тригер, що підраховує кількість спроб здати практичний іспит.

Цей тригер реалізує механізм підрахунку використаних кандидатом спроб на те, щоб здати практичний іспит. Кількість спроб збільшується при записі кандидата на практичний іспит і зменшується при видаленні запису. Цю логіку виконують тригер update\_amount\_of\_attempts\_trigger і тригерна функція update\_amount\_of\_attempts. Нижче наведено їхній код.

CREATE OR REPLACE FUNCTION update\_amount\_of\_attempts()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

UPDATE applicant

SET amount\_of\_attempts = amount\_of\_attempts + 1

WHERE applicant\_id = NEW.applicant\_id;

ELSIF TG\_OP = 'DELETE' THEN

UPDATE applicant

SET amount\_of\_attempts = amount\_of\_attempts - 1

WHERE applicant\_id = OLD.applicant\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER update\_amount\_of\_attempts\_trigger

AFTER INSERT OR DELETE ON log\_of\_practical\_exams

FOR EACH ROW

EXECUTE FUNCTION update\_amount\_of\_attempts();

На рисунку 8.5 показано роботу тригера при видаленні запису на практичний іспит. На рисунку 8.6 показано роботу тригера при доданні запису на практичний іспит. При видаленні запису кількість спроб (зберігається у колонці amount\_of\_attempts таблиці applicant) зменшилась на 1, а при доданні запису – збільшилась на 1.

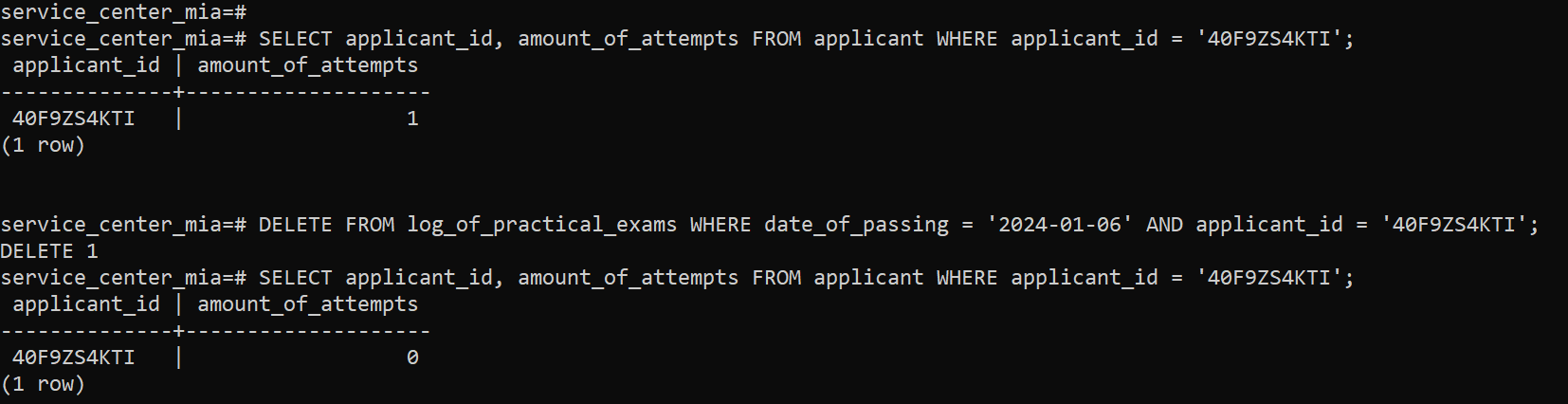


Рисунок 8.5 – Демонстрація роботи тригера при видаленні запису

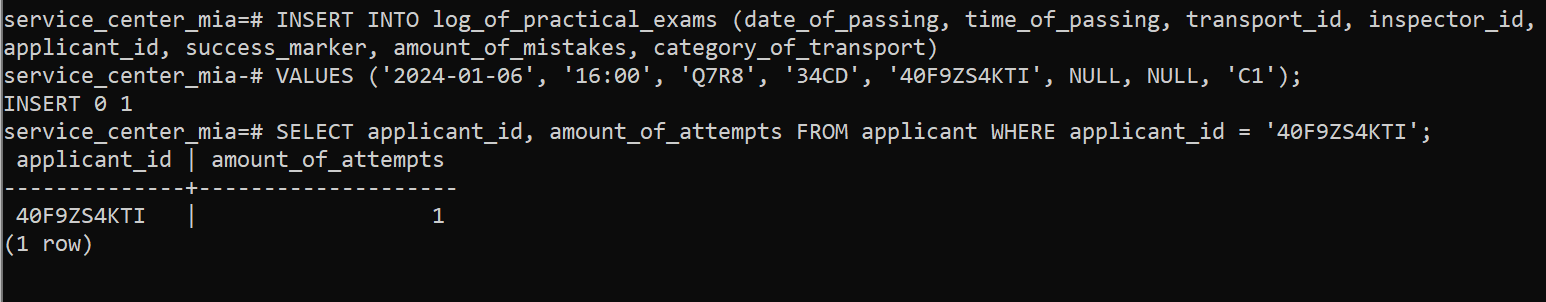


Рисунок 8.6 – Демонстрація роботи тригера при доданні запису

8.4. Тригер, що реагує на вставку водійського посвідчення.

Цей тригер реалізує таке обмеження вставки: не можна додавати водійське посвідчення тому, хто не здав практичний іспит. Цю логіку виконують тригер check\_practical\_exam\_trigger і тригерна функція check\_practical\_exam\_success. Нижче наведено їхній код.

CREATE OR REPLACE FUNCTION check\_practical\_exam\_success()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS (

SELECT 1 FROM log\_of\_practical\_exams pe

JOIN applicant a ON pe.applicant\_id = a.applicant\_id

WHERE a.last\_name = NEW.last\_name

AND a.first\_name = NEW.first\_name

AND a.middle\_name = NEW.middle\_name

AND pe.success\_marker = TRUE

)

THEN RAISE EXCEPTION 'Applicant has not passed the practical exam! The insertion was aborted!';

ELSE RETURN NEW;

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_practical\_exam\_trigger

BEFORE INSERT ON drivers\_license

FOR EACH ROW

EXECUTE FUNCTION check\_practical\_exam\_success();

На рисунку 8.7 продемонстровано роботу тригера при вставці водійського посвідчення тій особі, яка не здала практичний іспит. На рисунку 8.8 показано роботу тригера при вставці водійського посвідчення та отриманої категорії тій особі, яка практичний іспит здала.

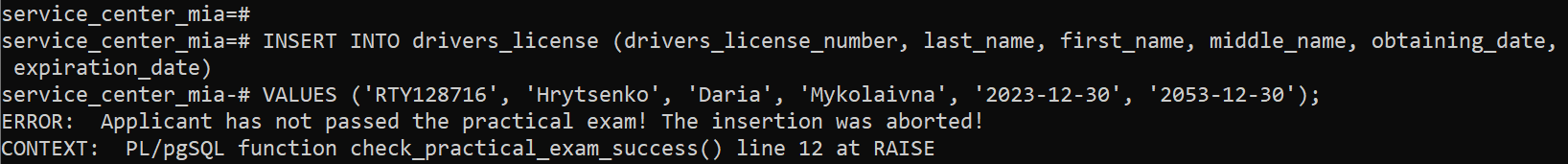


Рисунок 8.7 – Демонстрація роботи тригера при доданні посвідчення особі,

що не здала практичний іспит

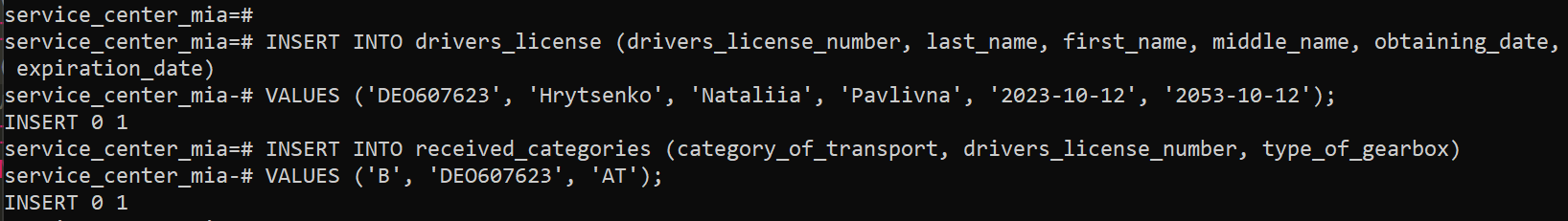


Рисунок 8.8 – Демонстрація роботи тригера при доданні посвідчення особі,

що здала практичний іспит

8.5. Тригер, що реагує на вставку отриманої категорії та коробки передач, на якій цю категорію було отримано.

Цей тригер реалізує таке обмеження вставки отриманої категорії: не можна додавати водійське посвідчення із зазначеною категорією та коробкою передач тому, хто не здав практичний іспит на цю категорію та коробку передач. Цю логіку виконують тригер check\_received\_categories\_trigger і тригерна функція check\_received\_categories. Нижче наведено їхній код.

CREATE OR REPLACE FUNCTION check\_received\_categories()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS (

SELECT 1

FROM log\_of\_practical\_exams pe

JOIN applicant a ON pe.applicant\_id = a.applicant\_id

JOIN transport t ON pe.transport\_id = t.transport\_id

WHERE a.last\_name = (SELECT last\_name FROM drivers\_license WHERE drivers\_license\_number = NEW.drivers\_license\_number)

AND a.first\_name = (SELECT first\_name FROM drivers\_license WHERE drivers\_license\_number = NEW.drivers\_license\_number)

AND a.middle\_name = (SELECT middle\_name FROM drivers\_license WHERE drivers\_license\_number = NEW.drivers\_license\_number)

AND pe.success\_marker = TRUE

AND pe.category\_of\_transport = NEW.category\_of\_transport

AND ((NEW.type\_of\_gearbox IS NULL AND t.type\_of\_gearbox IS NULL) OR

(NEW.type\_of\_gearbox IS NOT NULL AND NEW.type\_of\_gearbox = t.type\_of\_gearbox))

) THEN RAISE EXCEPTION 'The candidate did not pass the practical exam with the specified category and gearbox type!';

ELSE RETURN NEW;

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_received\_categories\_trigger

BEFORE INSERT ON received\_categories

FOR EACH ROW

EXECUTE FUNCTION check\_received\_categories();

На рисунку 8.9 продемонстровано роботу тригера при вставці водійського посвідчення тій особі, яка не здала практичний іспит на категорію Т на механічній коробці передач.

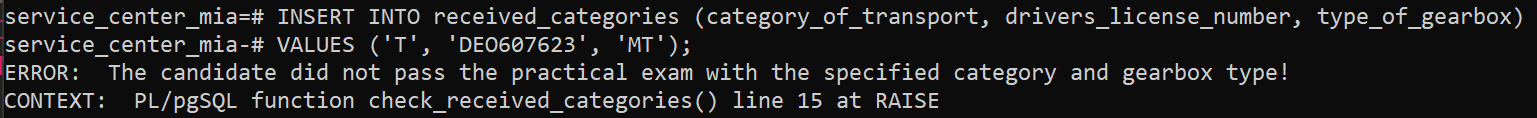


Рисунок 8.9 – Демонстрація роботи тригера при доданні посвідчення особі,

що не здала практичний іспит на категорію Т на механічній коробці передач

8.6. Тригер, що реагує на видалення водійського посвідчення.

Цей тригер реалізує механізм каскадного видалення водійського посвідчення. Видаляється усі відкриті водієм категорії. Цю логіку виконують тригер before\_delete\_drivers\_license і тригерна функція delete\_drivers\_license. Нижче наведено їхній код.

На рисунку 8.10 продемонстровано видалення водійського посвідчення.

CREATE OR REPLACE FUNCTION delete\_drivers\_license() RETURNS TRIGGER AS $$

BEGIN

DELETE FROM received\_categories

WHERE drivers\_license\_number = OLD.drivers\_license\_number;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER before\_delete\_drivers\_license

BEFORE DELETE ON drivers\_license

FOR EACH ROW

EXECUTE FUNCTION delete\_drivers\_license();

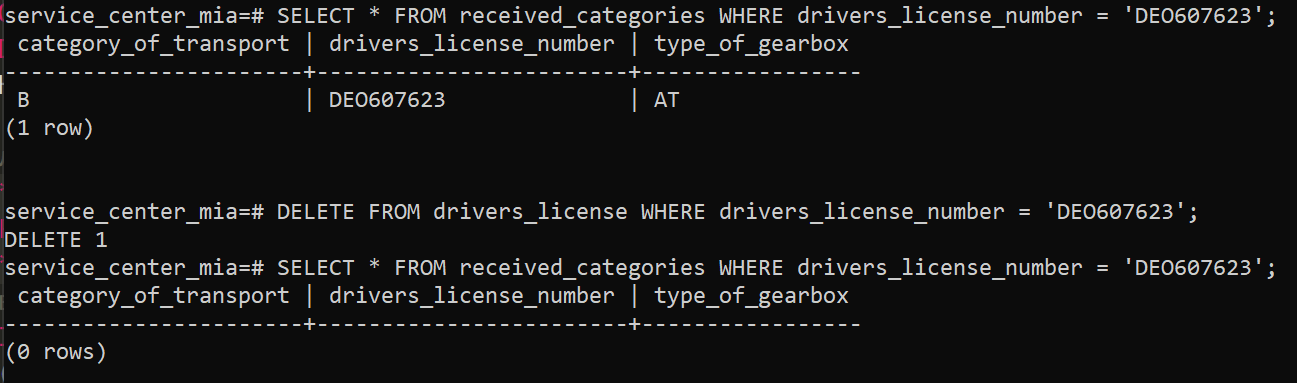


Рисунок 8.10 – Видалення водійського посвідчення

8.7. Тригер, що реагує на зміну оцінки практичного іспиту.

Цей тригер реалізує логіку перевірки правильності встановлення оцінки за практичний іспит. Механізм такий: іспит вважається складеним, якщо кількість помилок не перевищує 2 помилки. Цей тригер перевіряє правильність встановлення маркеру успіху і автоматично корегує його в залежності від кількості помилок. Важливо, що маркер успішності іспиту не можна поставити, якщо не вказана кількість помилок практичного іспиту. Цю логіку виконують тригер update\_practical\_exam\_success\_marker і тригерна функція check\_practical\_exam\_mistakes. Нижче наведено їхній код.

CREATE OR REPLACE FUNCTION check\_practical\_exam\_mistakes()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.amount\_of\_mistakes is NULL AND NEW.success\_marker IS NOT NULL THEN

RAISE NOTICE 'The number of mistakes is NULL, so the success marker is set to NULL.';

NEW.success\_marker = NULL;

ELSIF NEW.amount\_of\_mistakes < 3 AND NEW.success\_marker IS NOT TRUE THEN

IF NEW.success\_marker IS NOT TRUE THEN

NEW.success\_marker := TRUE;

RAISE NOTICE 'Success marker set to TRUE due to less than 2 mistakes';

END IF;

ELSIF NEW.amount\_of\_mistakes >= 3 AND NEW.success\_marker IS NOT FALSE THEN

IF NEW.success\_marker IS NOT FALSE THEN

NEW.success\_marker := FALSE;

RAISE NOTICE 'Success marker set to FALSE due to more than 2 mistakes';

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER update\_practical\_exam\_success\_marker

BEFORE INSERT OR UPDATE ON log\_of\_practical\_exams

FOR EACH ROW

EXECUTE FUNCTION check\_practical\_exam\_mistakes();

На рисунку 8.11 продемонстровано реакцію тригера на зміну маркера успішності практичного іспиту без вказання кількості помилок. На рисунку 8.12 показано реакцію на неправильну зміну маркеру успіху та оцінки за практичний іспит (екзамен не може бути складеним, якщо кількість помилок більша за 2).

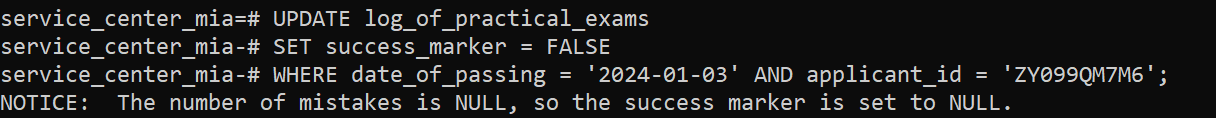


Рисунок 8.11 – Демонстрація роботи тригера при зміні маркеру успіху без вказання кількості помилок

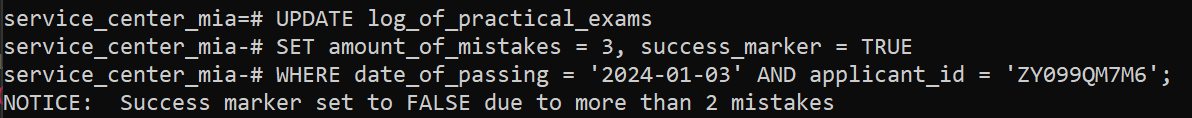


Рисунок 8.12 – Демонстрація роботи тригера при неправильній зміні маркеру успіху практичного екзамену

8.8. Тригер, що реагує на зміну оцінки теоретичного іспиту.

Цей тригер реалізує логіку перевірки правильності встановлення оцінки за теоретичний іспит. Механізм такий: іспит вважається складеним, якщо кількість правильних відповідей не менша за 18. Тригер перевіряє цю умову і автоматично змінює маркер успіху в залежності від оцінки (якщо його встановлено неправильно). Важливо, що маркер успішності іспиту не можна поставити, якщо не вказана оцінка теоретичного іспиту. Цю логіку виконують тригер update\_theoretical\_exam\_success\_marker і тригерна функція check\_theoretical\_exam\_grade. Нижче наведено їхній код.

CREATE OR REPLACE FUNCTION check\_theoretical\_exam\_grade()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.exam\_grade is NULL AND NEW.success\_marker IS NOT NULL THEN

RAISE NOTICE 'The number of mistakes is NULL, so the success marker is set to NULL.';

NEW.success\_marker = NULL;

ELSIF NEW.exam\_grade >= 18 AND NEW.success\_marker IS NOT TRUE THEN

IF NEW.success\_marker IS NOT TRUE THEN

NEW.success\_marker := TRUE;

RAISE NOTICE 'Success marker set to TRUE due to more than 17 correct answers';

END IF;

ELSIF NEW.exam\_grade < 18 AND NEW.success\_marker IS NOT FALSE THEN

IF NEW.success\_marker IS NOT FALSE THEN

NEW.success\_marker := FALSE;

RAISE NOTICE 'Success marker set to FALSE due to less than 18 correct answers';

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER update\_theoretical\_exam\_success\_marker

BEFORE INSERT OR UPDATE ON log\_of\_theoretical\_exams

FOR EACH ROW

EXECUTE FUNCTION check\_theoretical\_exam\_grade();

На рисунку 8.13 продемонстровано реакцію тригера на зміну маркера успішності теоретичного іспиту без вказання його оцінки. На рисунку 8.14 показано реакцію на неправильну зміну маркеру успіху та оцінки за теоретичний іспит (іспит не може бути складеним, якщо оцінка менша за 18).

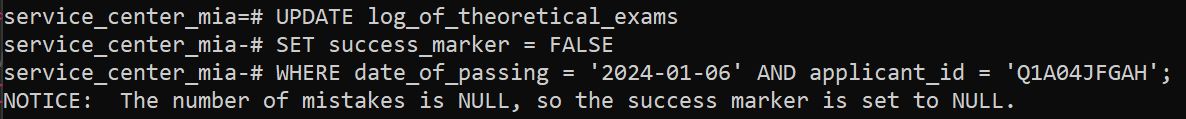


Рисунок 8.13 – Демонстрація роботи тригера при зміні маркеру успіху без вказання оцінки за екзамен

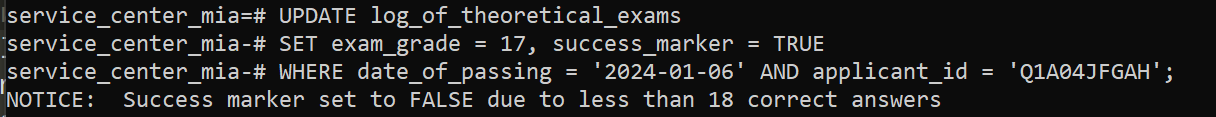


Рисунок 8.14 – Демонстрація роботи тригера при неправильній зміні маркеру успіху теоретичного екзамену

# РЕАЛІЗАЦІЯ ПРЕДСТАВЛЕНЬ

У базах даних важливим елементом є механізм створення представлень. Представлення є об'єктом бази даних, який представляє собою віртуальну таблицю на основі результатів запиту. Представлення можуть бути використані для спрощення складних запитів, об'єднання таблиць та інших операцій з даними, повторного використання коду.

Нижче наведено і описано представлення, що були реалізовані в базі даних service\_center\_mia, а також наведено результат їх використання.

9.1. Представлення, що містить у собі всі записи на практичний іспит, заплановані на поточний день.

В цьому представлення виводяться основні стовпці, необхідні для інформативного виведення запланованих практичних іспитів. Нижче наведений код і приклад використання.

CREATE OR REPLACE VIEW practical\_exams\_today AS

SELECT

pe.exam\_serial\_number,

pe.time\_of\_passing,

a.last\_name || ' ' || a.first\_name || ' ' || a.middle\_name AS full\_name,

pe.applicant\_id,

pe.transport\_id, pe.inspector\_id,

pe.success\_marker,

pe.category\_of\_transport

FROM (SELECT \* FROM log\_of\_practical\_exams WHERE date\_of\_passing = CURRENT\_DATE) pe

JOIN applicant a ON pe.applicant\_id = a.applicant\_id;

На рисунку 9.1. показано приклад виведення такого представлення. Для цього можна використати запит SELECT \* FROM practical\_exams\_today;

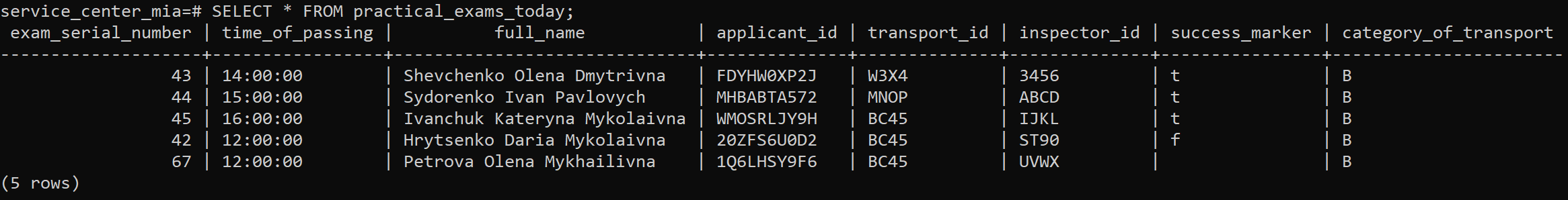


Рисунок 9.1 – Заплановані на поточний день здачі практичного іспиту

9.2. Представлення, що містить у собі всі записи на теоретичний іспит, заплановані на поточний день.

В цьому представленні виводяться не практичні, а теоретичні іспити, що були заплановані на поточний день. Дата береться СУБД із системи, і для даної дати виводяться ті стовпці, які необхідні для інформативного відображення кожного запису. Нижче наведено код цього представлення.

CREATE OR REPLACE VIEW theoretical\_exams\_today AS

SELECT te.exam\_serial\_number,

te.time\_of\_passing,

a.last\_name || ' ' || a.first\_name || ' ' || a.middle\_name AS full\_name,

te.applicant\_id,

te.inspector\_id,

te.success\_marker,

te.exam\_grade

FROM log\_of\_theoretical\_exams te

JOIN applicant a ON te.applicant\_id = a.applicant\_id

WHERE te.date\_of\_passing = CURRENT\_DATE

ORDER BY te.exam\_serial\_number;

На рисунку 9.2. показано приклад виведення створеного представлення. Для цього можна використати запит SELECT \* FROM theoretical\_exams\_today;

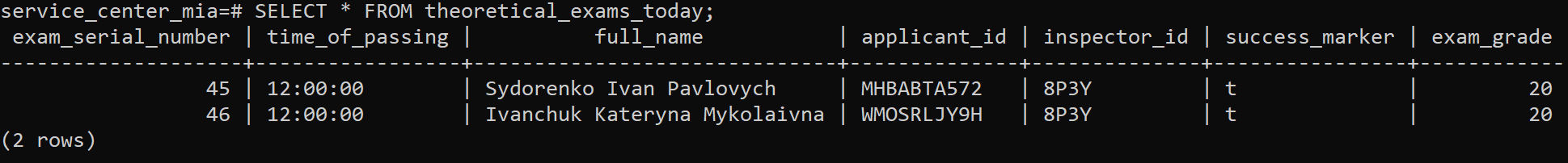


Рисунок 9.2 – Заплановані теоретичні іспити на поточний день

9.3. Представлення, що відображає отримані кожним водієм категорії.

В цьому представленні виводиться номер водійського посвідчення, ПІБ, отримані категорії та коробки передач, на яких ці категорії було отримано, а також дати отримання та закінчення строку дії посвідчення.

CREATE OR REPLACE VIEW drivers\_categories AS

SELECT dl.drivers\_license\_number,

dl.last\_name || ' ' || dl.first\_name || ' ' || dl.middle\_name AS full\_name,

rc.category\_of\_transport,

rc.type\_of\_gearbox,

dl.obtaining\_date,

dl.expiration\_date

FROM received\_categories rc LEFT JOIN drivers\_license dl ON rc.drivers\_license\_number = dl.drivers\_license\_number

ORDER BY drivers\_license\_number; JOIN applicant a ON pe.applicant\_id = a.applicant\_id;

На рисунку 9.3. показано приклад виведення всіх стовпчиків такого представлення. Для цього можна використати запит SELECT \* FROM drivers\_categories;

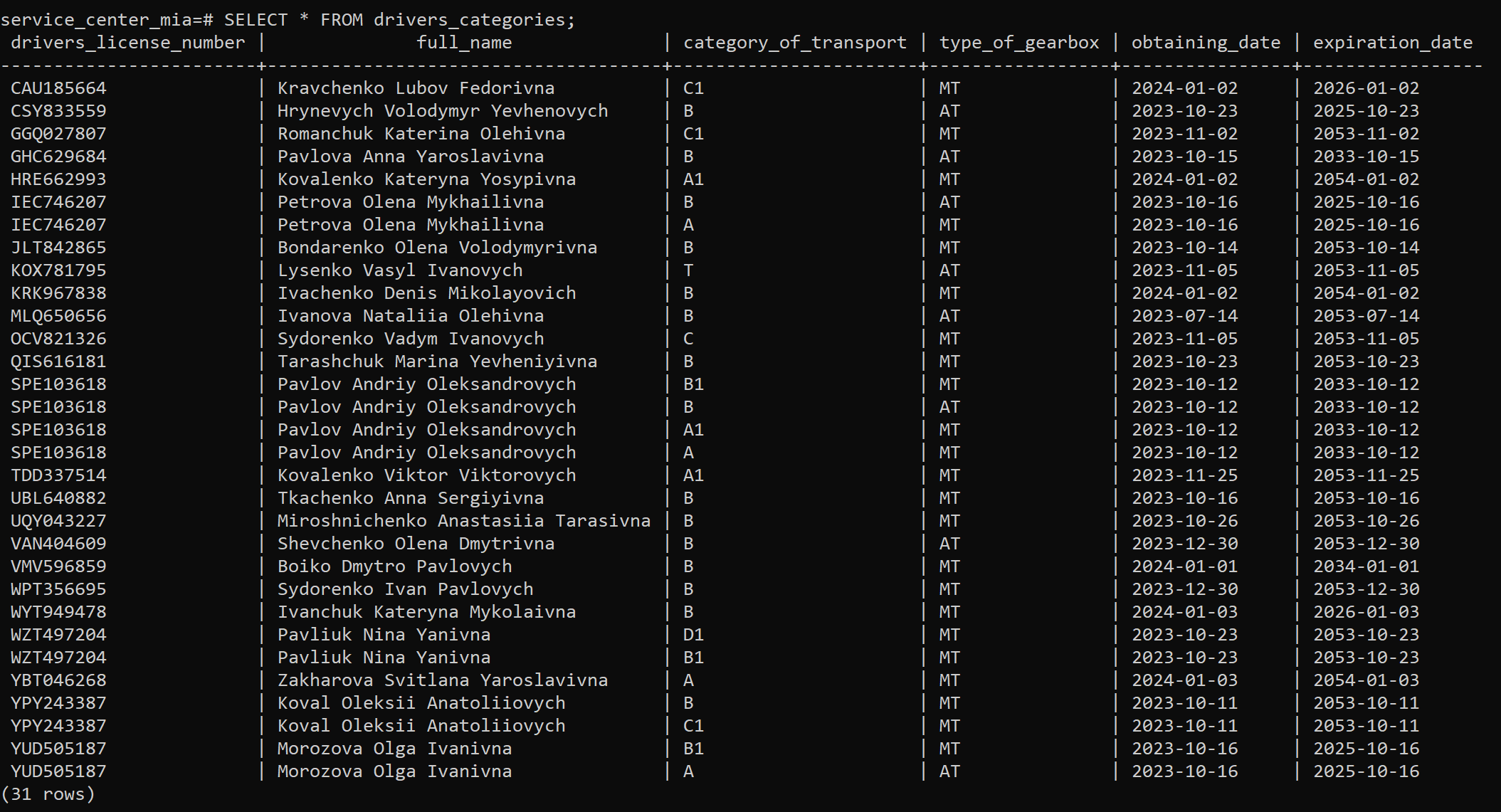


Рисунок 9.3 – Заплановані на поточний день здачі практичного іспиту

9.4. Представлення, що відображає отримані водійські посвідчення, в яких вичерпався строк ліцензії.

Водійські посвідчення діють обмежений термін і потребують перездачі іспитів через деякий час. Це представлення виводить саме ті посвідчення, які вже треба оновити. Нижче наведено його код.

CREATE OR REPLACE VIEW expired\_drivers\_licenses AS

SELECT dl.drivers\_license\_number,

a.last\_name || ' ' || a.first\_name || ' ' || a.middle\_name AS full\_name,

dl.expiration\_date

FROM drivers\_license dl

JOIN applicant a ON dl.last\_name = a.last\_name AND dl.first\_name = a.first\_name AND dl.middle\_name = a.middle\_name

WHERE dl.expiration\_date < CURRENT\_DATE;

           На рисунку 9.4. показано приклад виведення такого представлення для системної дати 1 січня 2036 року. Для виведення представлення можна використати запит SELECT \* FROM expired\_drivers\_licenses;

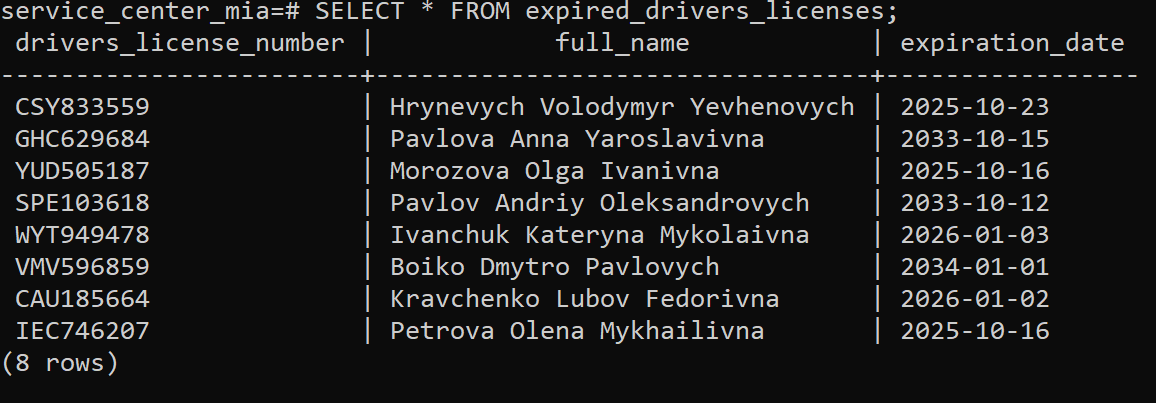


Рисунок 9.4 – Посвідчення із вичерпаним строком дії

9.5. Представлення кандидатів, що не здали практичний іспит, але готові до нього.

Це представлення виводить тих кандидатів, які зареєструвалися, мають право на керування ТЗ, успішно здали теоретичний іспит і готові здавати практичний, але поки що не здали його успішно. Код представлення і приклад виводу наведені нижче.

CREATE OR REPLACE VIEW applicants\_ready\_for\_practical\_exam AS

SELECT a.applicant\_id,

a.last\_name || ' ' || a.first\_name || ' ' || a.middle\_name AS full\_name,

a.phone\_number,

a.amount\_of\_attempts

FROM applicant a

WHERE a.applicant\_id IN (

SELECT te.applicant\_id

FROM log\_of\_theoretical\_exams te

WHERE te.success\_marker = TRUE

)

AND a.applicant\_id NOT IN (

SELECT pe.applicant\_id

FROM log\_of\_practical\_exams pe

WHERE pe.success\_marker = TRUE

);

           На рисунку 9.5. показано приклад виведення цього представлення. Слід використати запит SELECT \* FROM applicants\_ready\_for\_practical\_exam;

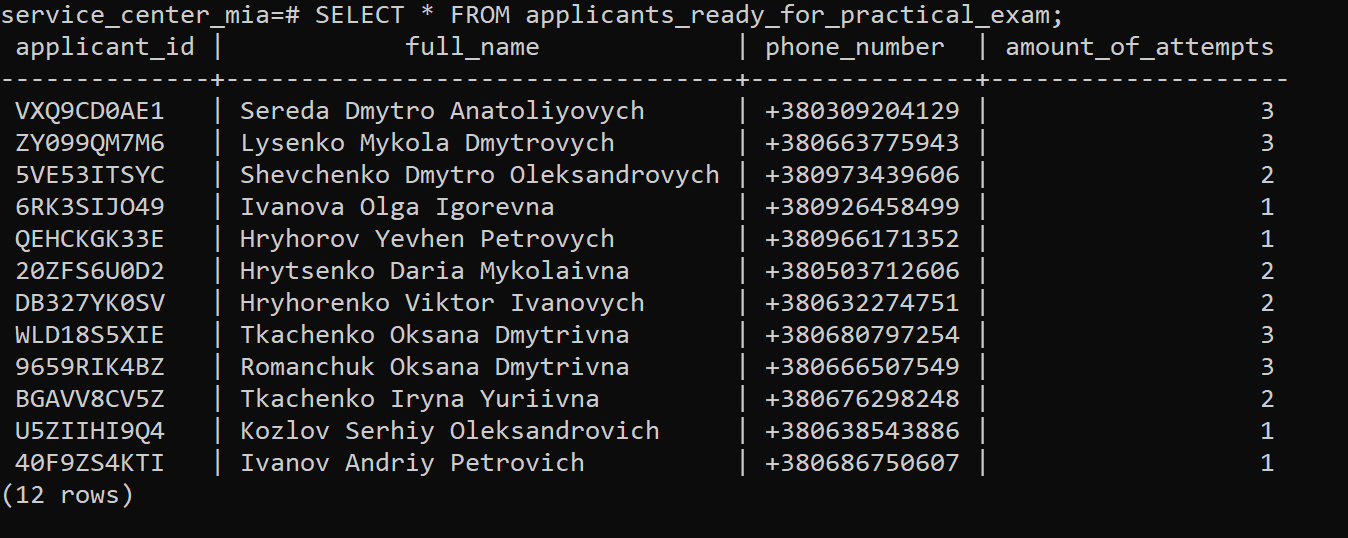


Рисунок 9.5 – Представлення кандидатів, готових до практичного іспиту

# РЕАЛІЗАЦІЯ ЗАПИТІВ

Для взаємодії з базою даних у мові програмування SQL передбачено використовування запитів. Запити в SQL використовуються для виконання різних операцій з даними, таких як отримання, оновлення, вставка або видалення інформації в базі даних. Мова маніпулювання даними DML включає в себе такі запити, як SELECT, INSERT, UPDATE, DELETE.

Нижче наведено приклади корисних запитів типу SELECT до бази даних service\_center\_mia, а також їхній словесний опис та приклад використання.

10.1. Запит, що виводить всіх інспекторів сервісного центру.

Використовується запит, що виводить всіх інспекторів практичного іспиту, а також запит, що виводить всіх інспекторів теоретичного іспиту. Обидва запити об’єднані в одну таблицю оператором UNION. На рисунку 10.1. показано виведення після запиту до бази даних.

SELECT last\_name, first\_name, middle\_name, phone\_number, inspector\_id

FROM practical\_exam\_inspector

UNION

SELECT last\_name, first\_name, middle\_name, phone\_number, inspector\_id

FROM theoretical\_exam\_inspector

ORDER BY last\_name;

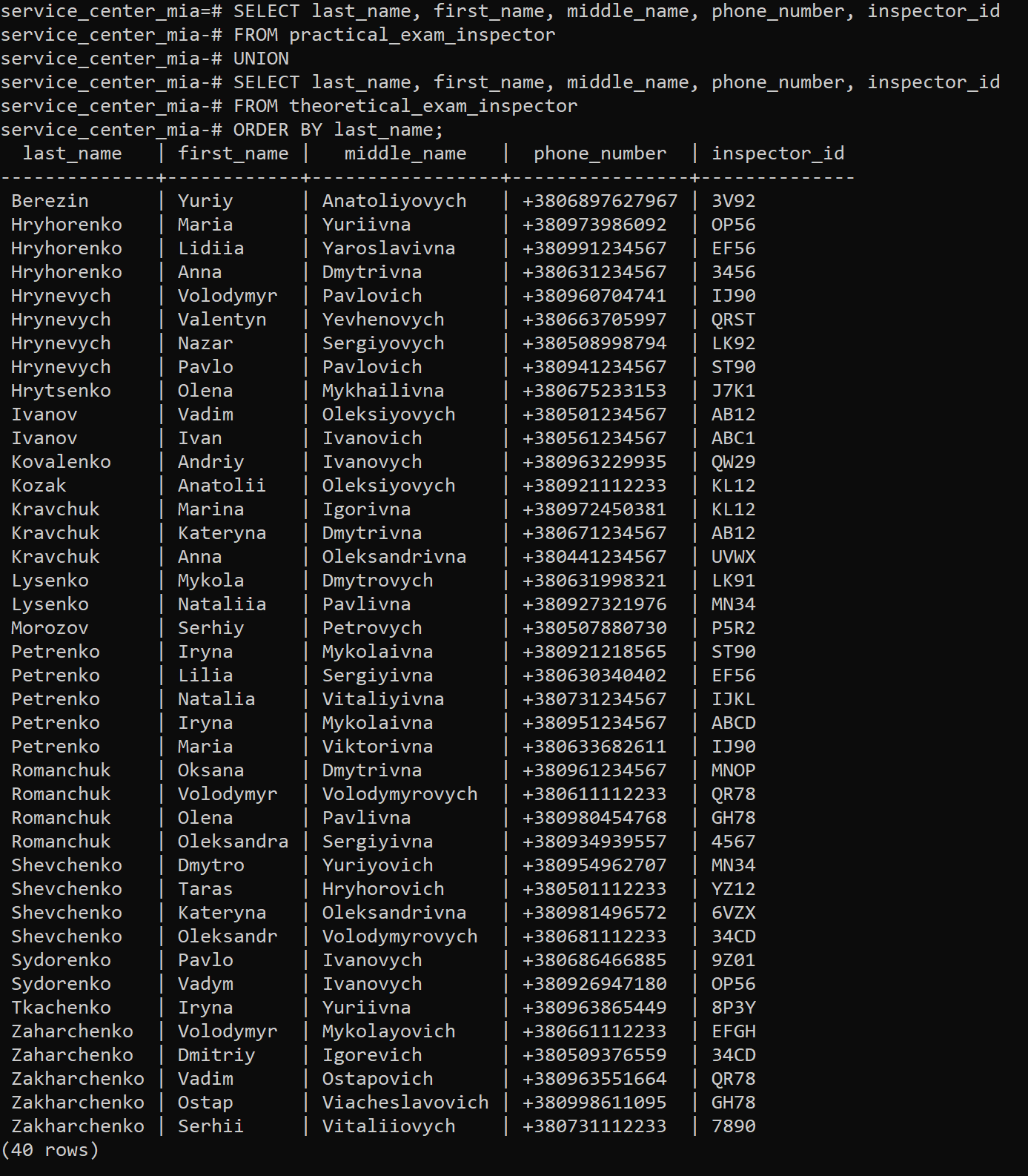


Рисунок 10.1 – Всі інспектори сервісного центру МВС

10.2. Запит, що виводить записи на теоретичний екзамен 06.01.2024.

Запит виводить ті записи на теоретичний іспит, у яких дата проведення дорівнює 6 січня 2024 року. На рисунку 10.2. показано виведення після запиту до бази даних.

SELECT lte.exam\_serial\_number AS exam\_number,

lte.time\_of\_passing,

a.last\_name || ' ' || a.first\_name || ' ' || a.middle\_name AS applicant\_name

FROM log\_of\_theoretical\_exams lte

JOIN applicant a ON lte.applicant\_id = a.applicant\_id

WHERE lte.date\_of\_passing = '2024-01-06';

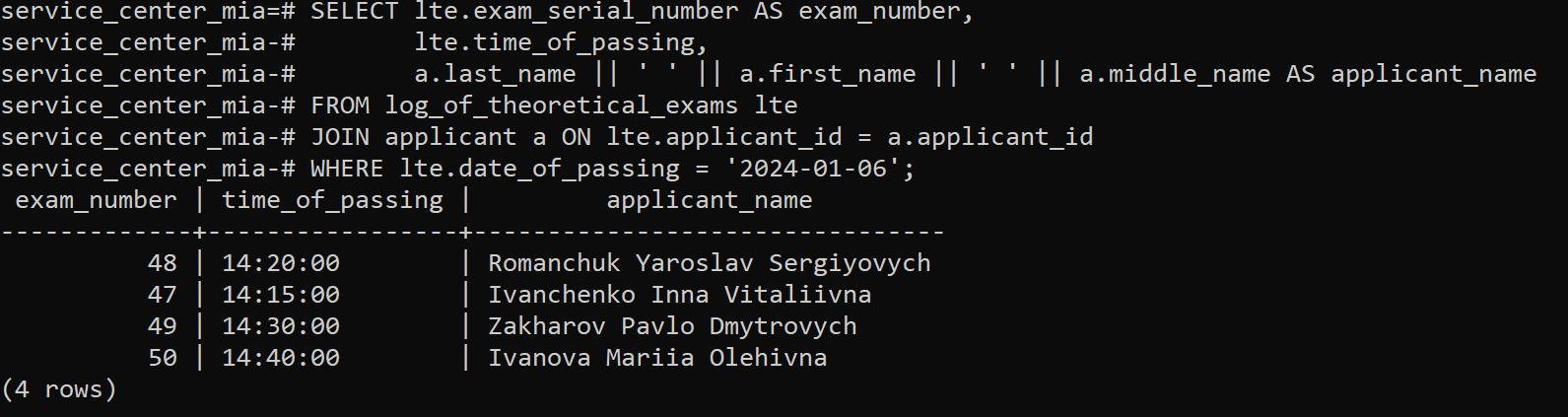


Рисунок 10.2 – Записи на теоретичний іспит на 6 січня 2024 року

10.3. Запит, що виводить інформацію про те, чи має право кандидат, що зареєстрований у базі даних СЦ, право на керування ТЗ.

Запит виводить ПІБ кандидата на отримання водійських прав, його унікальний ID та маркер можливості керувати ТЗ. Для отримання інформацію про право керування ТЗ використовується підзапит типу SELECT із таблиці medical\_certificates. На рисунку 10.3 показано результат такого запиту.

SELECT a.last\_name || ' ' || a.first\_name || ' ' || a.middle\_name AS applicant\_name,

a.applicant\_id,

a.phone\_number,

(SELECT mc.ability\_to\_drive\_marker

FROM medical\_certificates mc

WHERE mc.person\_id = a.applicant\_id LIMIT 1

) AS ability\_to\_drive

FROM applicant a

ORDER BY applicant\_name;

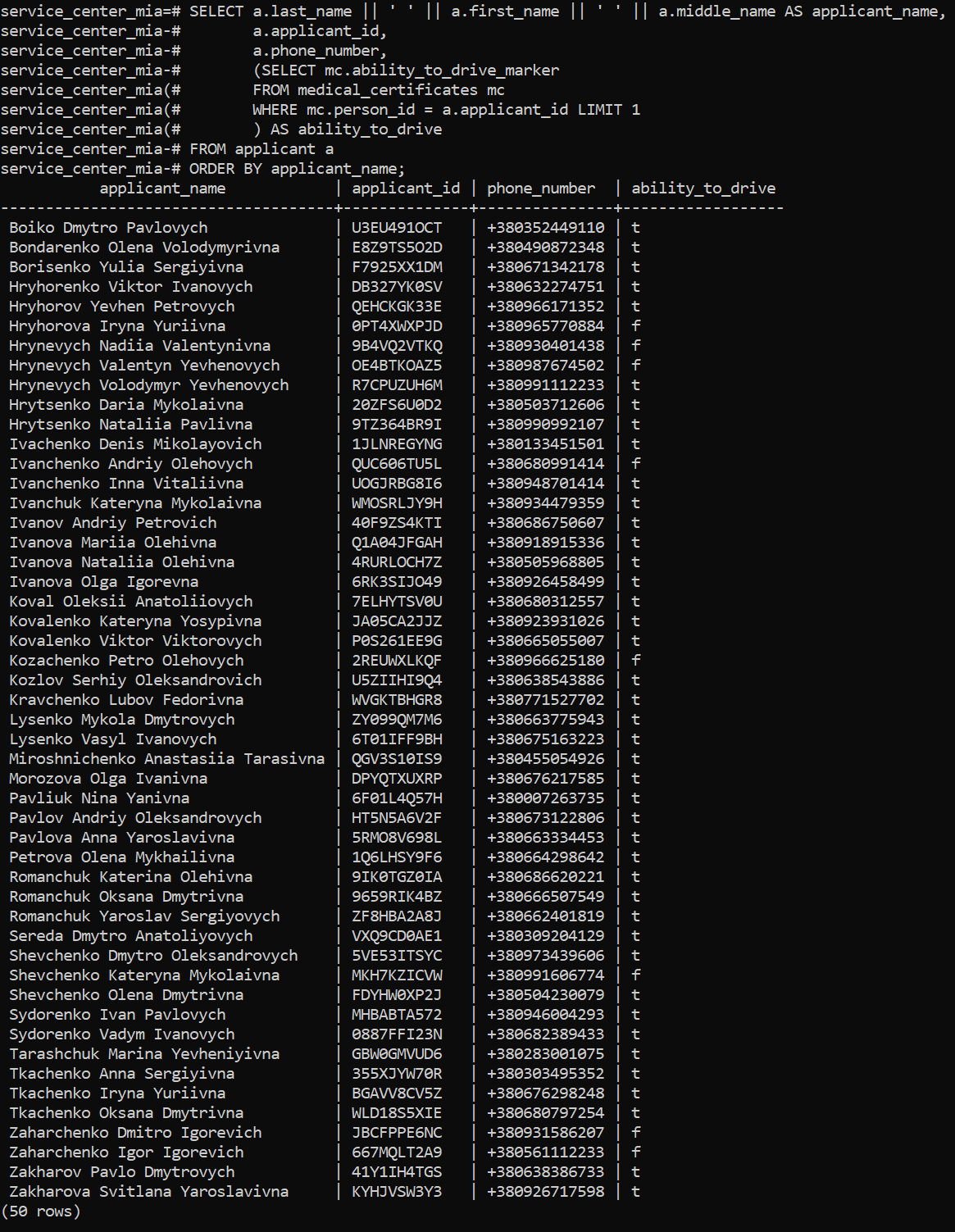


Рисунок 10.3 – Записи на практичний іспит на 6 січня 2024 року

10.4. Запит, що виводить записи на теоретичний екзамен, на які не з’явились кандидати.

Запит виводить ті записи на теоретичні іспити, які мали вже відбутись, за які кандидати не отримали оцінку, а, отже, які і проведено не було. Такі дані можна використовувати для видалення інформації про непроведені екзамени або для інших практичних цілей. Приклад використання запиту показано на рисунку 10.4.

SELECT lte.date\_of\_passing, lte.time\_of\_passing, a.last\_name, a.first\_name, a.middle\_name

FROM log\_of\_theoretical\_exams lte

JOIN applicant a ON lte.applicant\_id = a.applicant\_id

WHERE lte.date\_of\_passing < CURRENT\_DATE

AND lte.success\_marker IS NULL;

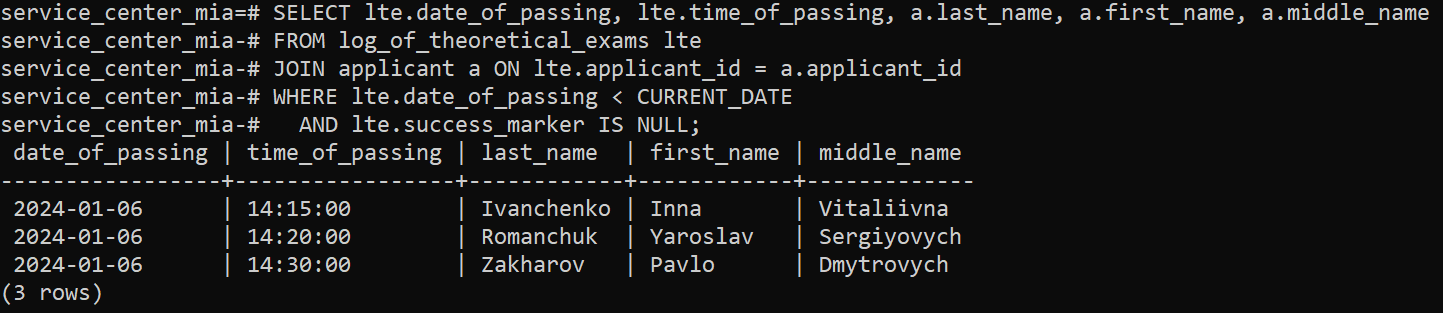


Рисунок 10.4 – Теоретичні іспити, на які не з’явилися кандидати

(дані для системної дати 07.01.2024)

10.5. Запит, що виводить таблицю отриманих кожним водієм категорій.

Запит виводить ПІБ, номер водійського посвідчення, отриману категорію та коробку передач, на якій цю категорію було отримано. Результат запиту на рисунку 10.5.

SELECT d.last\_name,

d.first\_name,

d.drivers\_license\_number AS license\_number,

rc.category\_of\_transport AS category,

rc.type\_of\_gearbox AS gearbox

FROM drivers\_license d

INNER JOIN received\_categories rc ON d.drivers\_license\_number = rc.drivers\_license\_number

ORDER BY d.last\_name, rc.category\_of\_transport;

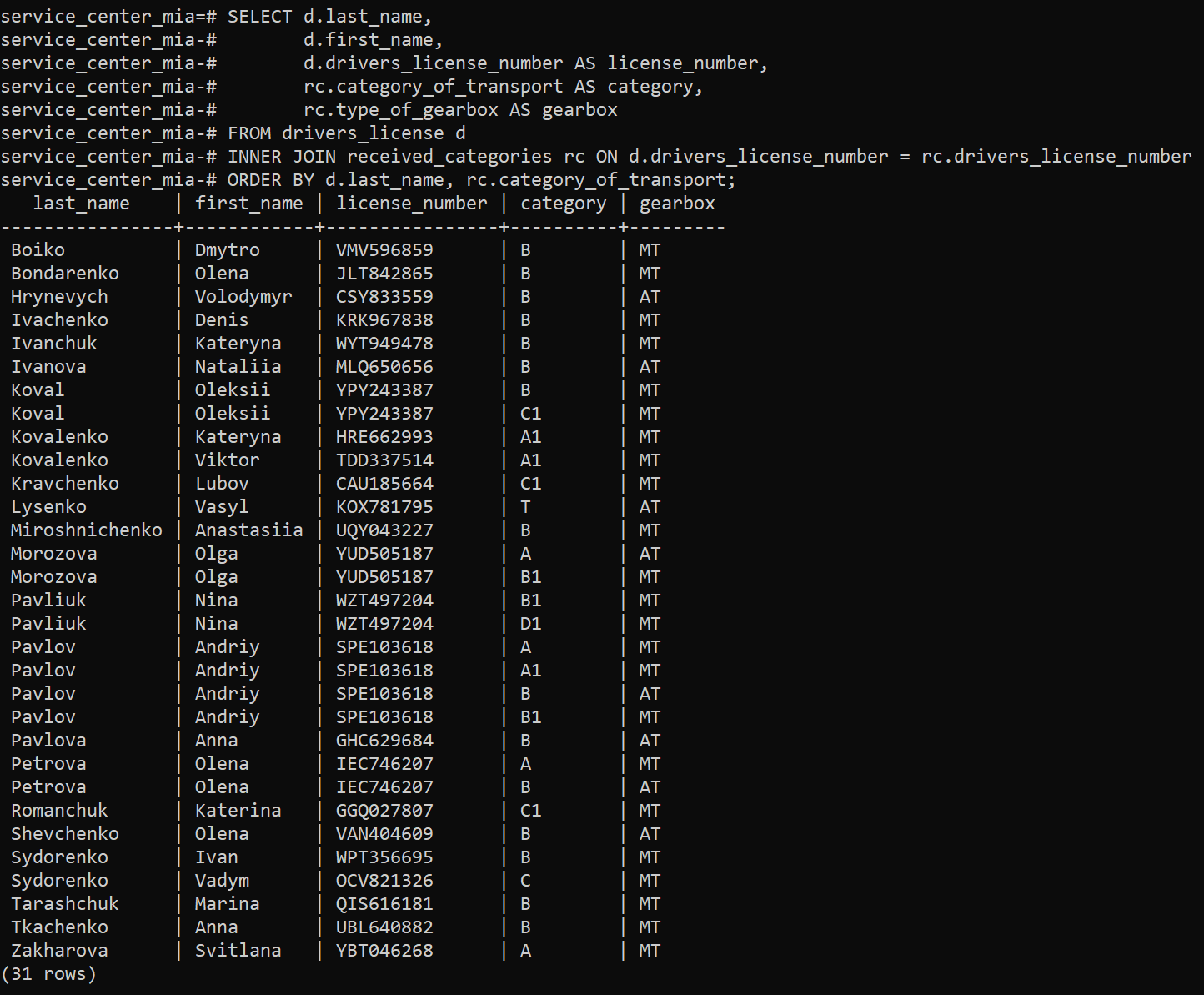


Рисунок 10.5 – Отримані кожним водієм категорії

10.6. Запит, що виводить середню кількість помилок, яку ставить інструктор практичних іспитів.

Запит виводить практичних інспекторів та середню кількість помилок, які вони ставлять. Таким чином можна дізнатися інформацію про найвимогливіших інспекторів. Результат запиту відображено на Рисунок 10.6.

SELECT pi.last\_name || ' ' || pi.first\_name AS inspector\_name,

ROUND(AVG(pe.amount\_of\_mistakes), 2) AS average\_mistakes

FROM practical\_exam\_inspector pi

INNER JOIN log\_of\_practical\_exams pe ON pi.inspector\_id = pe.inspector\_id

GROUP BY pi.inspector\_id, pi.last\_name, pi.first\_name;

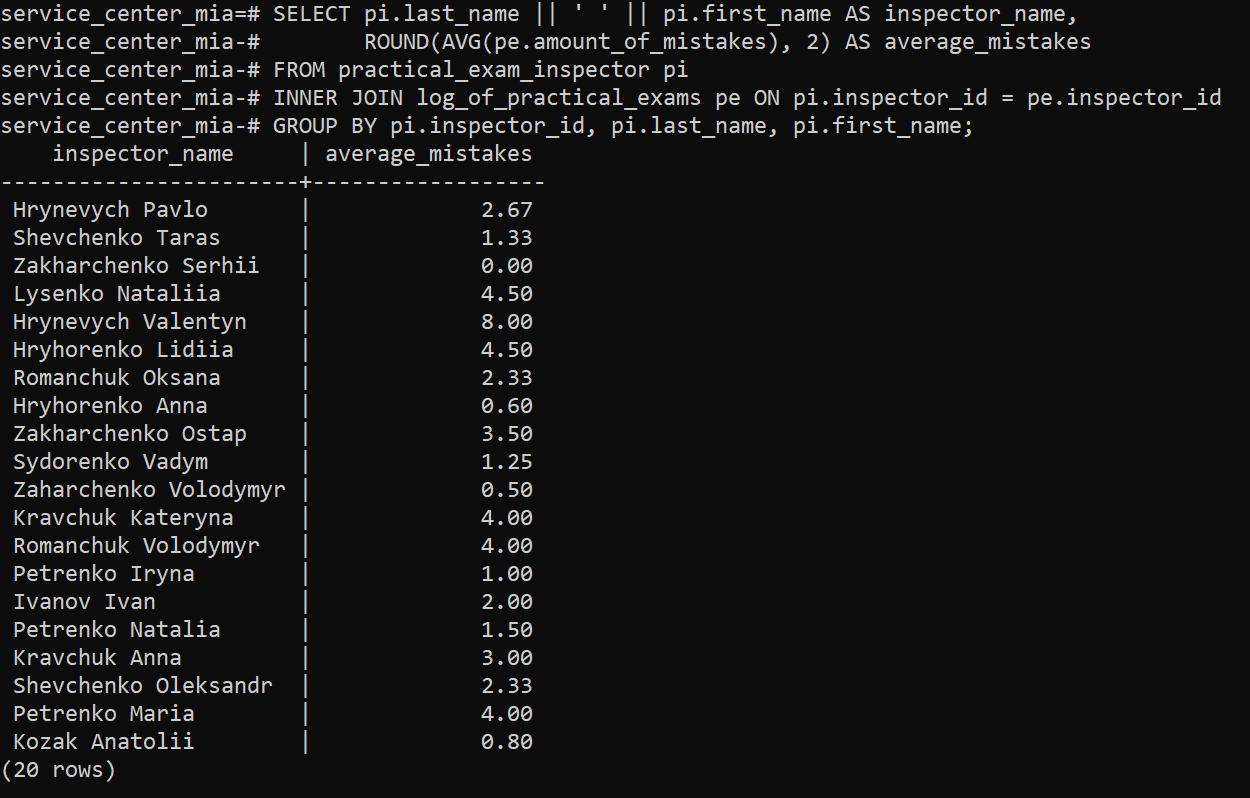


Рисунок 10.6 – Середня кількість помилок, поставлена інструктором

10.7. Запит, що виводить заплановані практичні іспити

Запит виводить ті практичні іспити, які заплановані, але ще не проведені, тобто у них не вказаний маркер успішності. Результат запиту відображено на рисунку 10.7.

SELECT

lpe.date\_of\_passing,

CONCAT(a.last\_name, ' ', a.first\_name) AS applicant\_name,

a.applicant\_id,

( SELECT t.category\_of\_transport

FROM transport t

WHERE t.transport\_id = lpe.transport\_id

) AS category\_of\_transport

FROM applicant a

JOIN log\_of\_practical\_exams lpe ON a.applicant\_id = lpe.applicant\_id

WHERE lpe.date\_of\_passing >= CURRENT\_DATE AND lpe.amount\_of\_mistakes IS NULL

ORDER BY lpe.exam\_serial\_number;

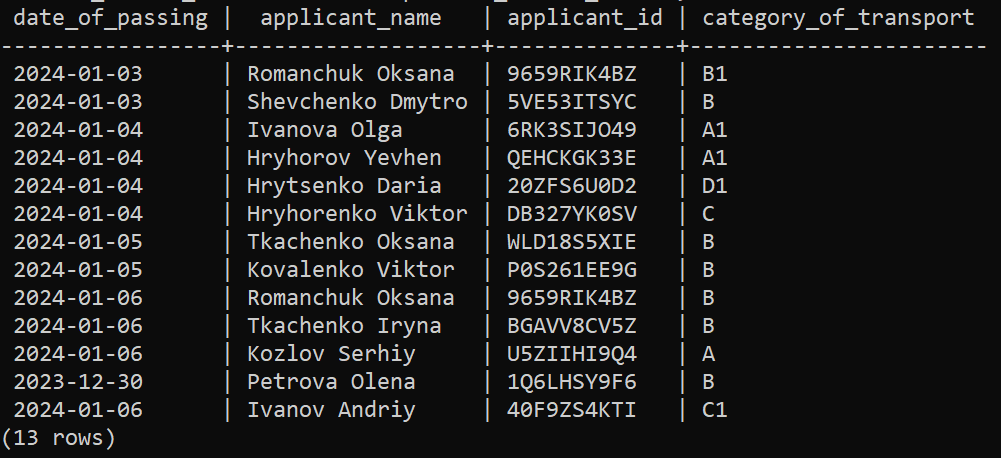


Рисунок 10.7 – Заплановані практичні іспити

10.8. Запит, що виводить % не зарахованих інструктором екзаменів.

Запит виводить загальний % не зарахованих інструктором практичних екзаменів від проведених. Результат запиту відображено на рисунку 10.8.

SELECT CONCAT(pi.last\_name, ' ', pi.first\_name),

COUNT(\*) AS total\_exams,

SUM(CASE WHEN lpe.success\_marker THEN 0 ELSE 1 END) AS unsuccessful,

ROUND(SUM(CASE WHEN lpe.success\_marker THEN 0 ELSE 1 END) \* 100.0 / COUNT(\*), 2) AS percentage\_u

FROM practical\_exam\_inspector pi

JOIN log\_of\_practical\_exams lpe ON pi.inspector\_id = lpe.inspector\_id

GROUP BY pi.inspector\_id;



Рисунок 10.8 – Виведення % не зарахованих практичним інструктором іспитів

10.9. Запит, що виводить студентів, які провалили теоретичний іспит.

Запит виводить тих студентів, які не здали теоретичний іспит хоча б один раз. Результат запиту відображено на рисунку 10.9.

SELECT CONCAT(a.last\_name, ' ', a.first\_name) AS applicant\_name,

lte.exam\_grade

FROM applicant a

JOIN log\_of\_theoretical\_exams lte ON a.applicant\_id = lte.applicant\_id

WHERE lte.exam\_grade >= 18;

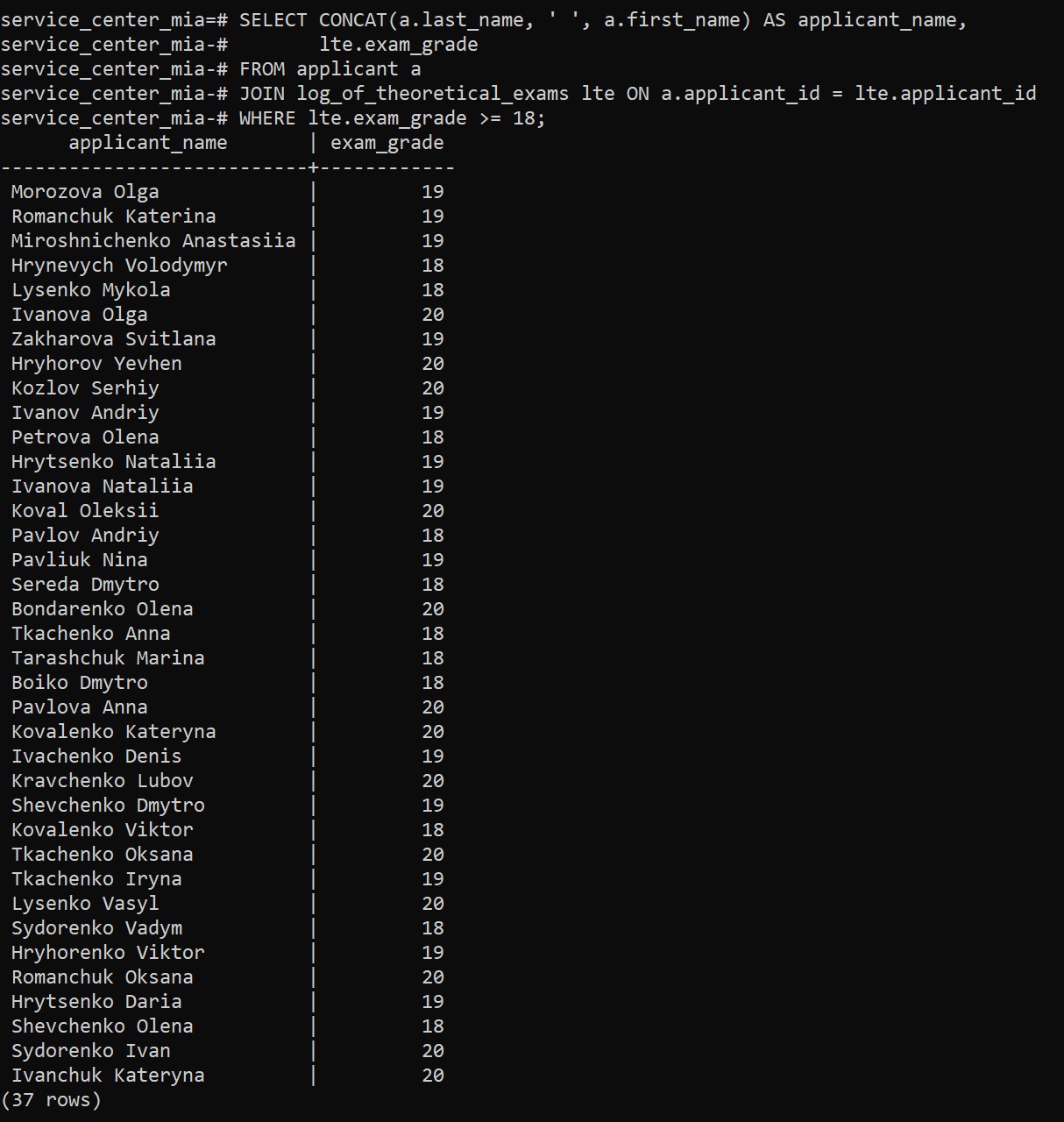


Рисунок 10.9 – Студенти, які не здали теоретичний іспит з першого разу

10.10. Однакові за суттю запити, що виводять кількість проведених кожним інспектором іспитів.

Наступний запит виводить кількість проведених екзаменів кожним інспектором теоретичних іспитів. Результат запиту на рисунку 10.10.

SELECT ti.inspector\_id,

CONCAT(ti.last\_name, ' ', ti.first\_name) as inspector\_name,

COUNT(te.applicant\_id) AS examined

FROM theoretical\_exam\_inspector ti

LEFT JOIN log\_of\_theoretical\_exams te ON ti.inspector\_id = te.inspector\_id

GROUP BY ti.inspector\_id;

Другий запит виводить ту ж інформацію, що і попередній, але вже для інспекторів практичних іспитів. Результат запиту на рисунку 10.11.

SELECT pi.inspector\_id,

CONCAT(pi.last\_name, ' ', pi.first\_name) as inspector\_name,

COUNT(pe.applicant\_id) AS examined

FROM practical\_exam\_inspector pi

LEFT JOIN log\_of\_practical\_exams pe ON pi.inspector\_id = pe.inspector\_id

GROUP BY pi.inspector\_id;

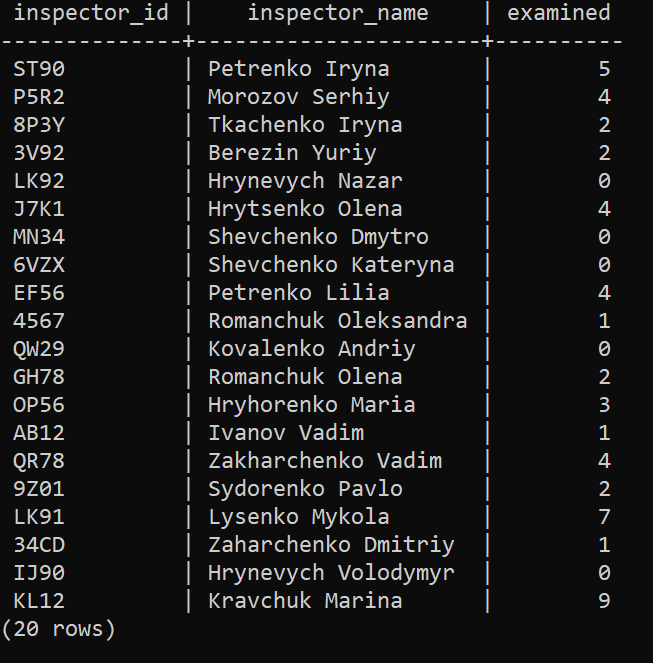


Рисунок 10.10 – Кількість проведених екзаменів інспекторів з теорії



Рисунок 10.11 – Кількість проведених екзаменів інспекторів з практики

10.11. Однакові за суттю запити, що виводять кількість транспорту кожної категорії.

Наступний запит виводить кількість транспорту кожної категорії, що належить сервісному центру. Тобто рахуються тільки ті ТЗ, що мають affiliation\_marker = TRUE. Результат запиту відображено на рисунку 10.12.

SELECT c.category\_of\_transport, COUNT(t.transport\_id) AS total\_count

FROM category\_of\_transport c

LEFT JOIN transport t ON c.category\_of\_transport = t.category\_of\_transport

WHERE t.affiliation\_marker = TRUE

GROUP BY c.category\_of\_transport;

Другий запит виводить кількість зареєстрованих у СЦ транспортних засобів, які можуть належати як сервісному центру, так і автошколам. Іншими словами, значення affiliation\_marker не важливе. Результат запиту на рисунку 10.13.

SELECT c.category\_of\_transport, COUNT(t.transport\_id) AS total\_count

FROM category\_of\_transport c

LEFT JOIN transport t ON c.category\_of\_transport = t.category\_of\_transport

GROUP BY c.category\_of\_transport;

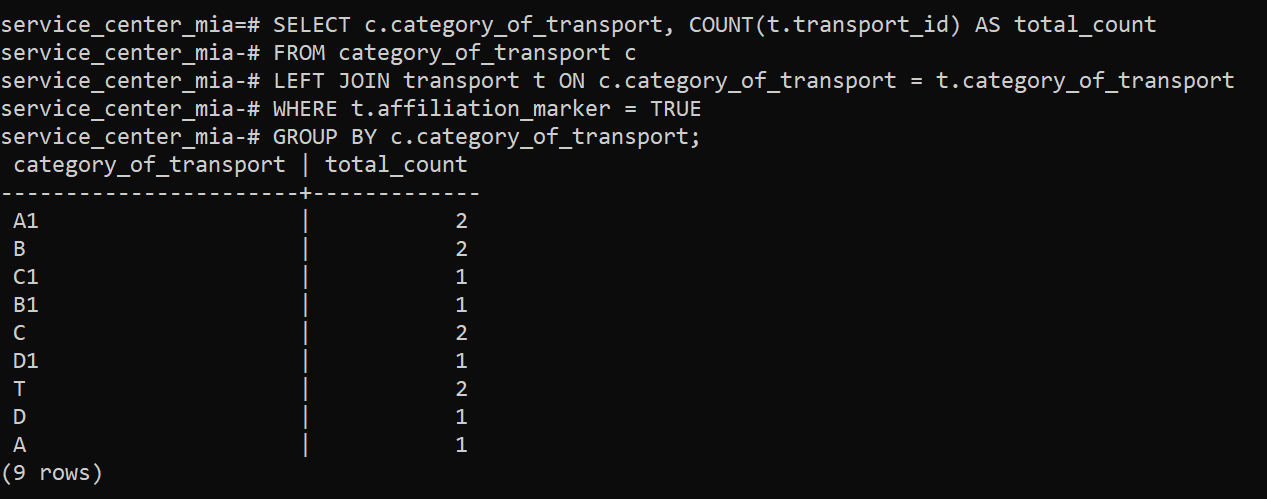


Рисунок 10.12 – Виведення кількості ТЗ кожної категорії сервісного центру

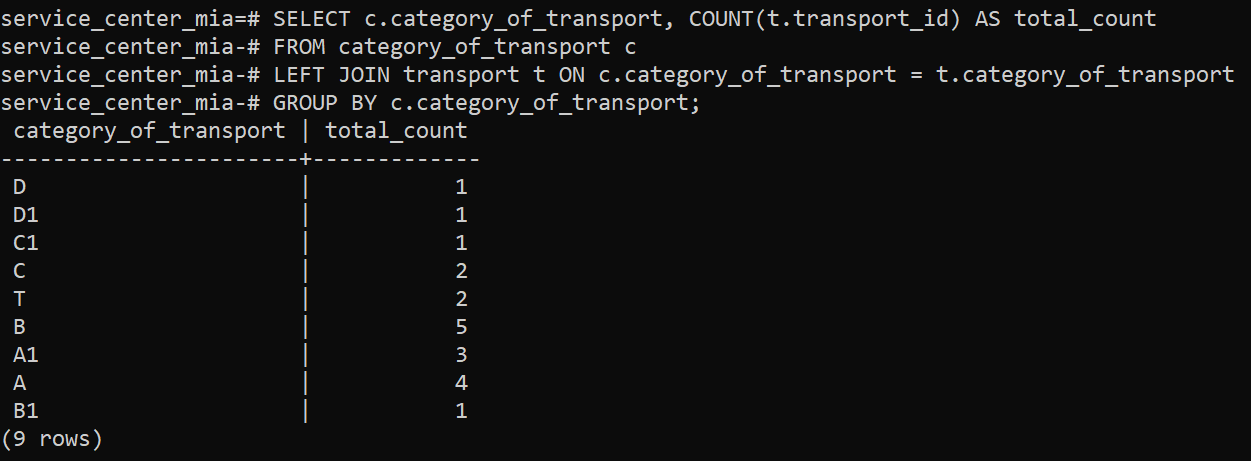


Рисунок 10.13 – Виведення кількості ТЗ кожної категорії незалежно від приналежності сервісному центру

10.12. Запит, що виводить тих, хто не здав практику із інформацією про право керування ТЗ.

Запит виводить тих студентів, які не здали практичний іспит. Окрім того, виводиться інформація про їхню можливість керування транспортним засобом, що взята з їхніх медичних карток. Результат запиту відображено на рисунку 10.14.

SELECT a.applicant\_id AS id,

CONCAT(a.last\_name, ' ', a.first\_name, ' ', a.middle\_name) AS name,

CASE WHEN mc.ability\_to\_drive\_marker = 't' THEN 'Керувати може'

ELSE ''

END AS ability\_to\_drive

FROM drivers\_license d

RIGHT JOIN applicant a ON a.last\_name = d.last\_name AND a.first\_name = d.first\_name AND a.middle\_name = d.middle\_name

INNER JOIN medical\_certificates mc ON a.applicant\_id = mc.person\_id

WHERE d.drivers\_license\_number IS NULL;

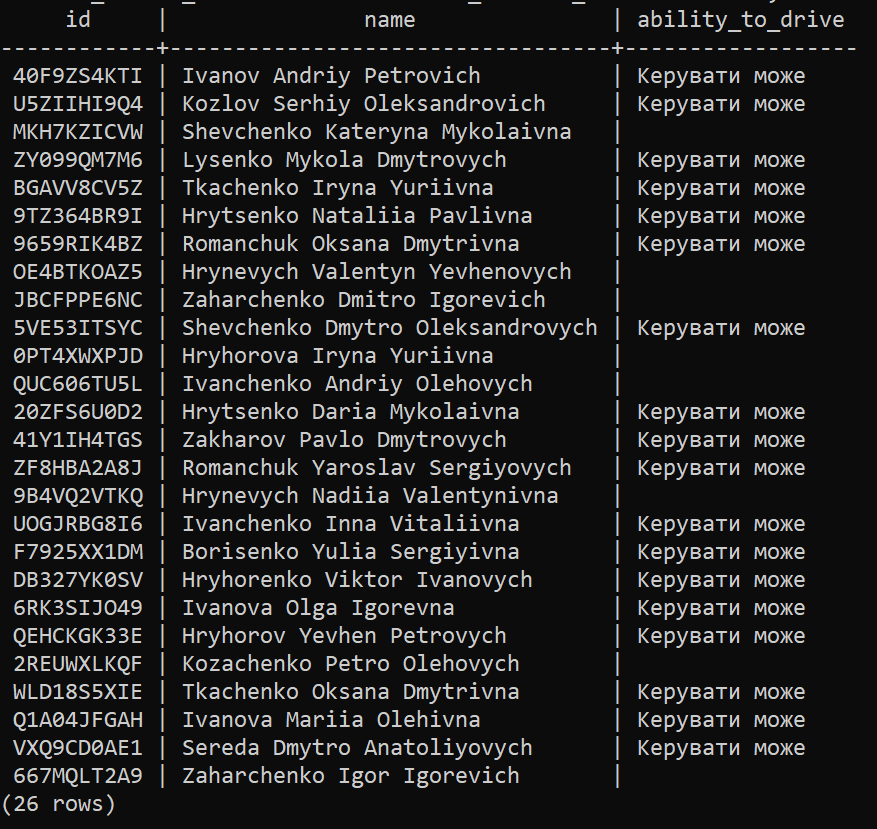


Рисунок 10.14 – Студенти, які не здали практичний іспит

10.13. Запит, що виводить інформацію про водійське посвідчення за шаблоном із 3 перших букв.

Запит виводить інформацію про водіїв, номер водійського посвідчення яких починається на OCV. Для такої ідентифікації використовується шаблон LIKE. Результат запиту відображено на рисунку 10.15.

SELECT a.applicant\_id,

CONCAT(a.last\_name, ' ', a.first\_name, ' ', a.middle\_name) AS full\_name,

d.drivers\_license\_number

FROM applicant a

INNER JOIN drivers\_license d ON a.last\_name = d.last\_name AND a.first\_name = d.first\_name AND a.middle\_name = d.middle\_name

WHERE d.drivers\_license\_number LIKE 'OCV%';

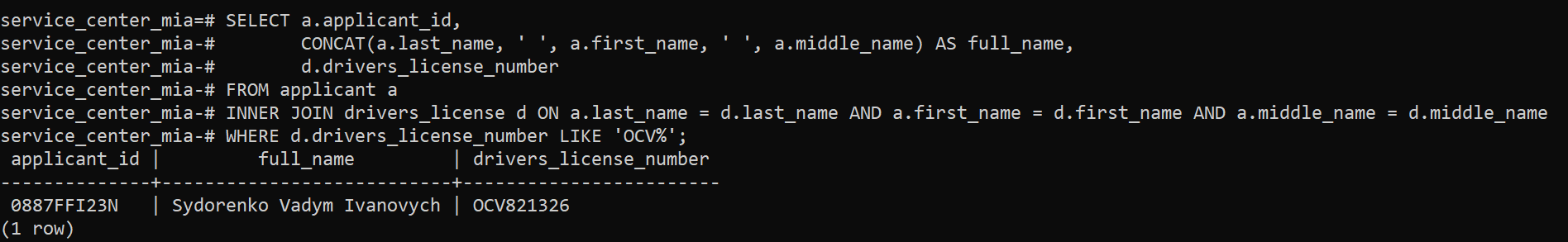


Рисунок 10.15 – Водії, водійські посвідчення яких починаються на ‘OCV’

10.14. Запит, що виводить екзаменаторів, які частіше зараховують екзамени за інших інспекторів.

Запит виводить тих інспекторів практичного екзамену, які зараховують практичні іспити частіше за колег, тобто кількість зарахованих іспитів у яких вища за середню. Результат запиту відображено на рисунку 10.16.

WITH SuccessfulExams AS (

SELECT inspector\_id, COUNT(\*) AS successful\_exams\_count

FROM log\_of\_practical\_exams

WHERE success\_marker = TRUE

GROUP BY inspector\_id

), AverageSuccessfulExams AS (

SELECT AVG(successful\_exams\_count) AS avg\_successful\_exams

FROM SuccessfulExams)

SELECT pi.last\_name, pi.first\_name, pi.middle\_name, sec.successful\_exams\_count AS successful\_exams, ROUND(ase.avg\_successful\_exams, 2) AS average

FROM practical\_exam\_inspector pi

JOIN SuccessfulExams sec ON pi.inspector\_id = sec.inspector\_id

CROSS JOIN AverageSuccessfulExams ase

WHERE sec.successful\_exams\_count > ase.avg\_successful\_exams;

|  |  |
| --- | --- |
|  |  |

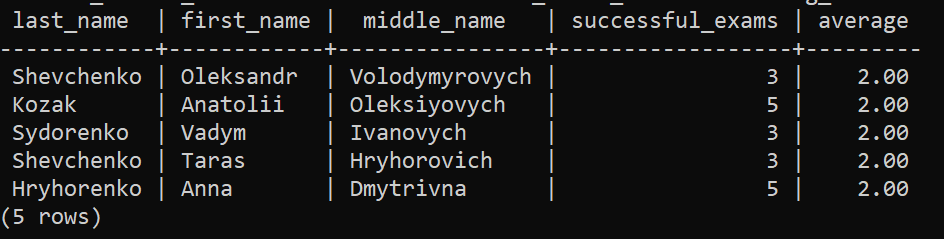


Рисунок 10.16 – Інспектори, які частіше зараховують практичний екзамен

10.15. Запит, що виводить ті водійські посвідчення, які треба оновити в поточному календарному році.

Запит виводить тих водіїв, яким в поточному році треба оновити своє водійське посвідчення. Для порівняння року береться дата системи. Результат запиту для системної дати 1 січня 2026 року відображено на рисунку 10.17.

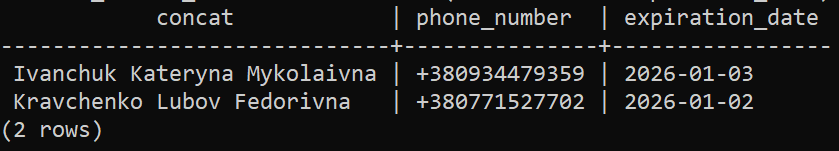
SELECT CONCAT(a.last\_name, ' ', a.first\_name, ' ', a.middle\_name), a.phone\_number, dl.expiration\_date

FROM drivers\_license dl

JOIN applicant a ON a.first\_name = dl.first\_name AND a.last\_name = dl.last\_name AND a.middle\_name = dl.middle\_name

WHERE EXTRACT(YEAR FROM dl.expiration\_date) = EXTRACT(YEAR FROM CURRENT\_DATE);

|  |  |
| --- | --- |
|  |  |

 Рисунок 10.17 – Водії, яким в цьому році треба оновити посвідчення

10.16. Запит, що виводить тих інспекторів теоретичного іспиту, які мають значний досвід.

Запит виводить інспекторів теоретичного іспиту, які мають значний досвід роботи (більше 50 проведених екзаменів). Через невелику тестову вибірку даних, запит не виведе жодного інспектора (Рисунок 10.18). Але, якщо замість 50 проведених екзаменів поставити 5 (для перевірки правильності роботи запиту), запит виведе тих інспекторів, які провели більше 5 теоретичних іспитів (Рисунок 10.19).

SELECT tei.last\_name, tei.first\_name, tei.middle\_name, tei.inspector\_id, exam\_count.examined

FROM theoretical\_exam\_inspector tei

JOIN (

SELECT inspector\_id, COUNT(\*) AS examined

FROM log\_of\_theoretical\_exams

GROUP BY inspector\_id

HAVING COUNT(\*) > 50

) AS exam\_count ON tei.inspector\_id = exam\_count.inspector\_id;

|  |  |
| --- | --- |
|  |  |

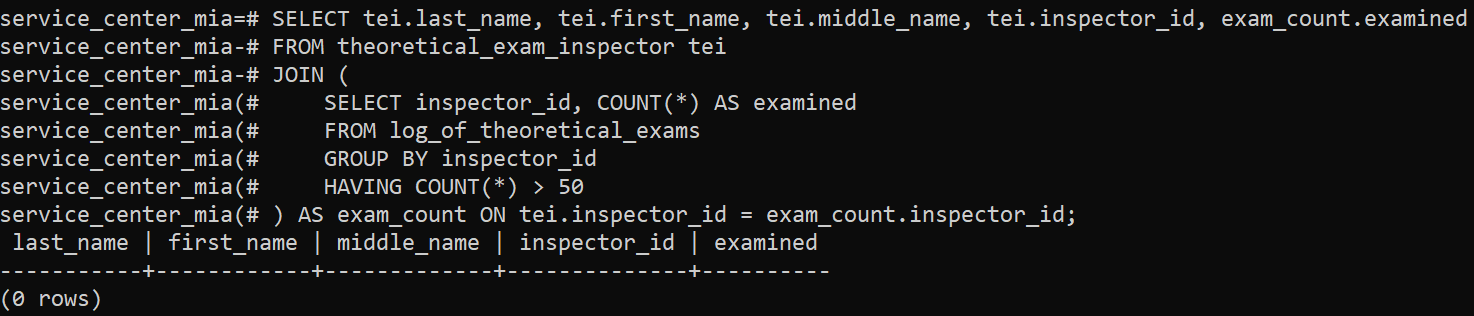


 Рисунок 10.18 – Інспектори теоретичного іспиту, що мають значний досвід роботи

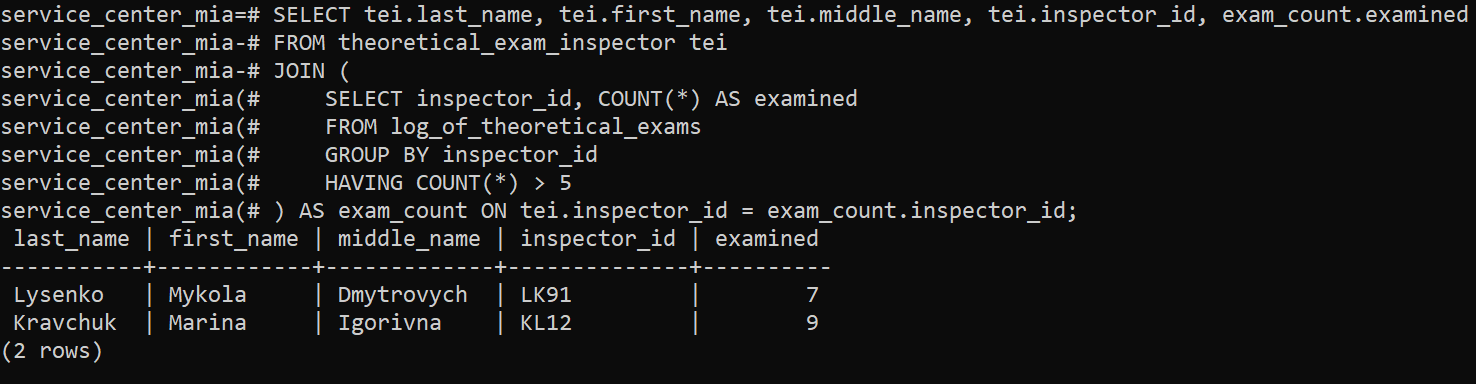


 Рисунок 10.19 – Перевірка правильності роботи запиту

10.17. Запит, що виводить тих водіїв, що здавали практичний іспит на вантажівці.

Запит виводить тих водіїв, які здавали практичний іспит на вантажівці (окрім вантажівки можна підставити будь-який інший тип транспорту із таблиці type\_of\_transport, який записується у тимчасову таблицю temp\_variables). Результат виводу на рисунку 10.20.

CREATE TEMP TABLE temp\_variables AS

SELECT 'Truck' AS type\_of\_transport;

SELECT a.last\_name, a.first\_name, a.middle\_name, subquery.category\_of\_transport, tv.type\_of\_transport

FROM applicant a

JOIN (

SELECT le.applicant\_id, t.category\_of\_transport, t.purpose\_of\_transport

FROM log\_of\_practical\_exams le

JOIN transport t ON le.transport\_id = t.transport\_id

) AS subquery ON a.applicant\_id = subquery.applicant\_id

JOIN temp\_variables tv ON subquery.purpose\_of\_transport = tv.type\_of\_transport;

DROP TABLE temp\_variables;

|  |  |
| --- | --- |
|  |  |

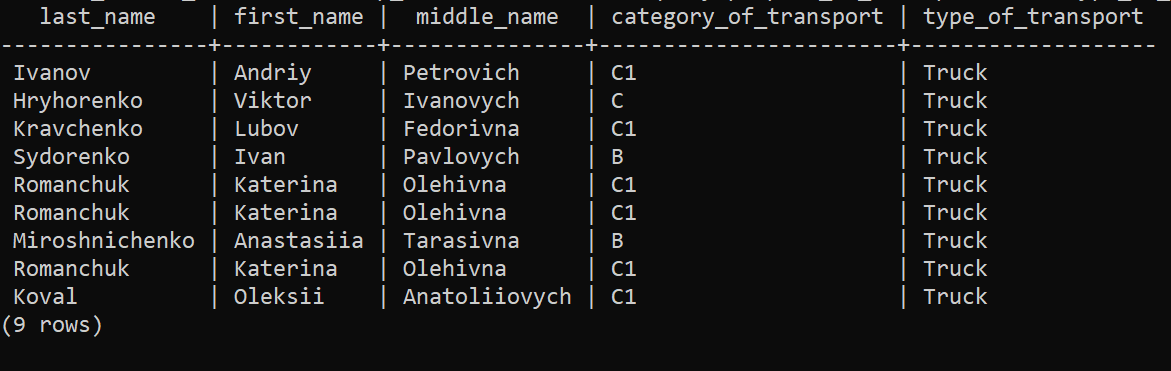


 Рисунок 10.20 – Водії, які здавали практику на вантажівці

10.18. Запит, що виводить тих водіїв, які мають задану групу крові.

Запит виводить тих водіїв, які у яких в медичній картці прописана шукана група крові. Важливо, що шукати можна декілька груп крові одночасно. Результат виводу шуканої групи крові 2+ та 4- на рисунку 10.21.

CREATE TEMP TABLE temp\_blood\_group (group\_of\_blood VARCHAR(2));

INSERT INTO temp\_blood\_group VALUES ('2+');

INSERT INTO temp\_blood\_group VALUES ('4-');

SELECT a.last\_name,

a.first\_name,

a.middle\_name,

tbg.group\_of\_blood AS blood

FROM applicant a, temp\_blood\_group tbg

WHERE EXISTS ( SELECT 1

FROM medical\_certificates mc

WHERE mc.person\_id = a.applicant\_id

AND tbg.group\_of\_blood = CONCAT(mc.group\_of\_blood, mc.rhesus\_factor));

DROP TABLE temp\_blood\_group;

|  |  |
| --- | --- |
|  |  |



 Рисунок 10.21 – Водії, що мають кров 2+ або 4-

10.19. Запит, що виводить той транспортний засіб, на якому проведено найбільшу кількість практичних іспитів.

Запит виводить машину-рекордсмена по кількості прийнятих екзаменів. Результат запиту на рисунку 10.22.

SELECT t.\*, COUNT(lpe.transport\_id) AS total\_exams

FROM transport t

LEFT JOIN log\_of\_practical\_exams lpe ON t.transport\_id = lpe.transport\_id

GROUP BY t.transport\_id

ORDER BY total\_exams DESC

LIMIT 1;

|  |  |
| --- | --- |
|  |  |

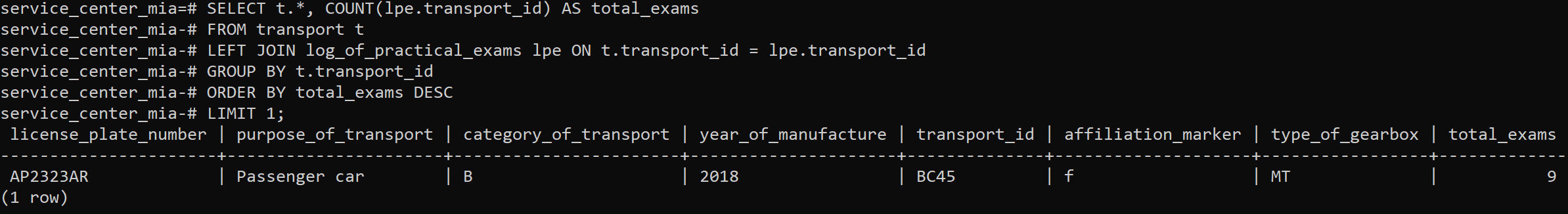


 Рисунок 10.22 – Машина-рекордсмен за кількістю проведених іспитів

10.20. Запит, що виводить записи на практичний екзамен, на які не з’явились кандидати і які не було проведено.

Запит виводить ті записи на практичні іспит, які вже мали відбутись, але не були проведені та за які не були виставлені маркери успішності. Приклад використання запиту показано на рисунку 10.23.

SELECT lpe.date\_of\_passing, lpe.time\_of\_passing, a.last\_name, a.first\_name, a.middle\_name

FROM log\_of\_practical\_exams lpe

JOIN applicant a ON lpe.applicant\_id = a.applicant\_id

WHERE lpe.date\_of\_passing < CURRENT\_DATE

AND lpe.success\_marker IS NULL;

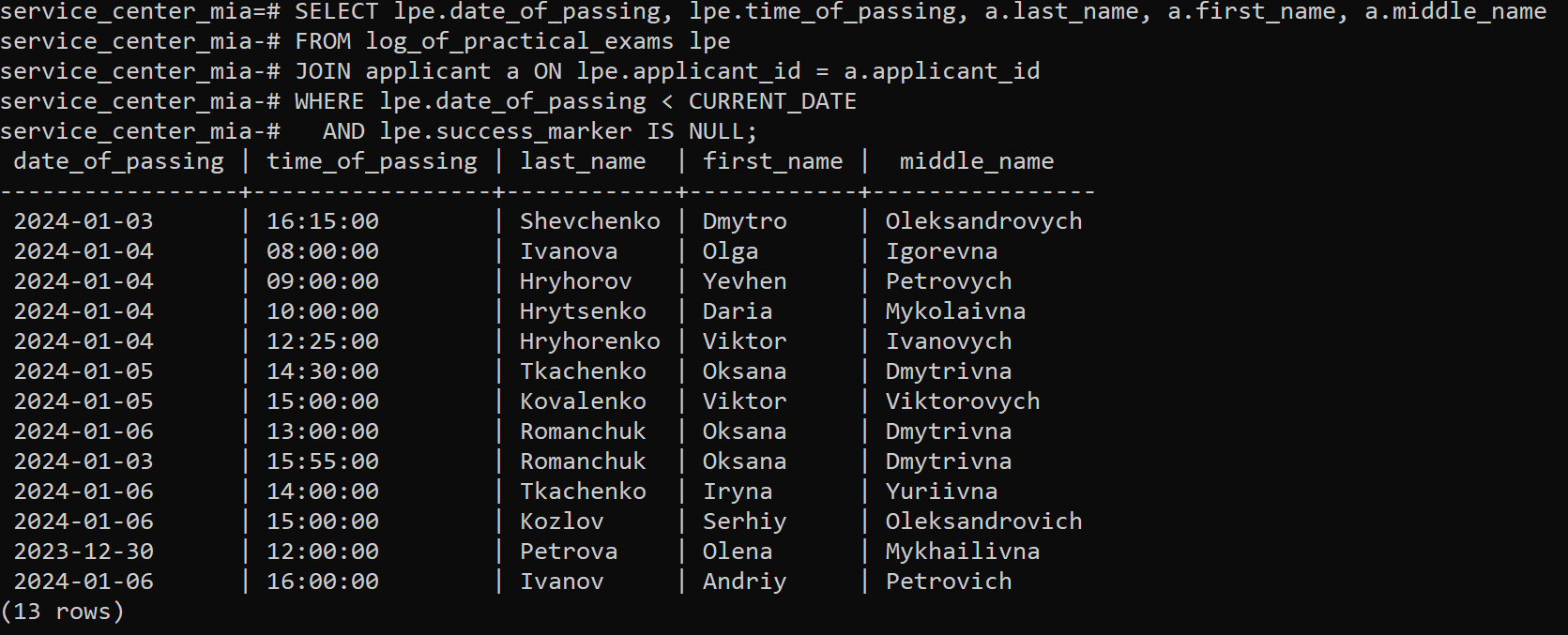


Рисунок 10.23 – Практичні іспити, на які не з’явилися кандидати

(дані для системної дати 10.01.2024)

# РЕАЛІЗАЦІЯ ФУНКЦІЙ ТА ПРОЦЕДУР

Збережені функції та процедури - це програмні об'єкти в базі даних, які дозволяють зберігати та викликати певний набір SQL-операцій. Ці об’єкти можуть приймати параметри, їх оброблювати, або виконувати будь-які інші дії, необхідні розробнику. Для роботи із функціями та процедурами можна визначати та використовувати локальні змінні. Окрім того, функції можуть повертати значення, рядки, таблиці, якщо це необхідно.

Нижче наведено реалізовані функції, їхній словесний опис та суть, а також приклади використання.

11.1. Функція, що виводить отримані кожним водієм категорії.

Ця функція виводить отримані кожним водієм категорії у зручному вигляді, рядком. Для цього використовується агрегатна функція STRING\_AGG. Нижче наведено код функції, а на рисунку 11.1 – приклад роботи. Для виклику функції слід використати запит SELECT \* FROM get\_license\_categories();

CREATE OR REPLACE FUNCTION get\_license\_categories()

RETURNS TABLE (full\_name TEXT, categories TEXT)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT dl.last\_name || ' ' || dl.first\_name || ' ' || dl.middle\_name AS full\_name,

STRING\_AGG(rc.category\_of\_transport, ', ') AS categories

FROM drivers\_license dl

LEFT JOIN received\_categories rc ON dl.drivers\_license\_number = rc.drivers\_license\_number

GROUP BY dl.last\_name, dl.first\_name, dl.middle\_name

ORDER BY dl.last\_name, dl.first\_name, dl.middle\_name;

END;

$$;

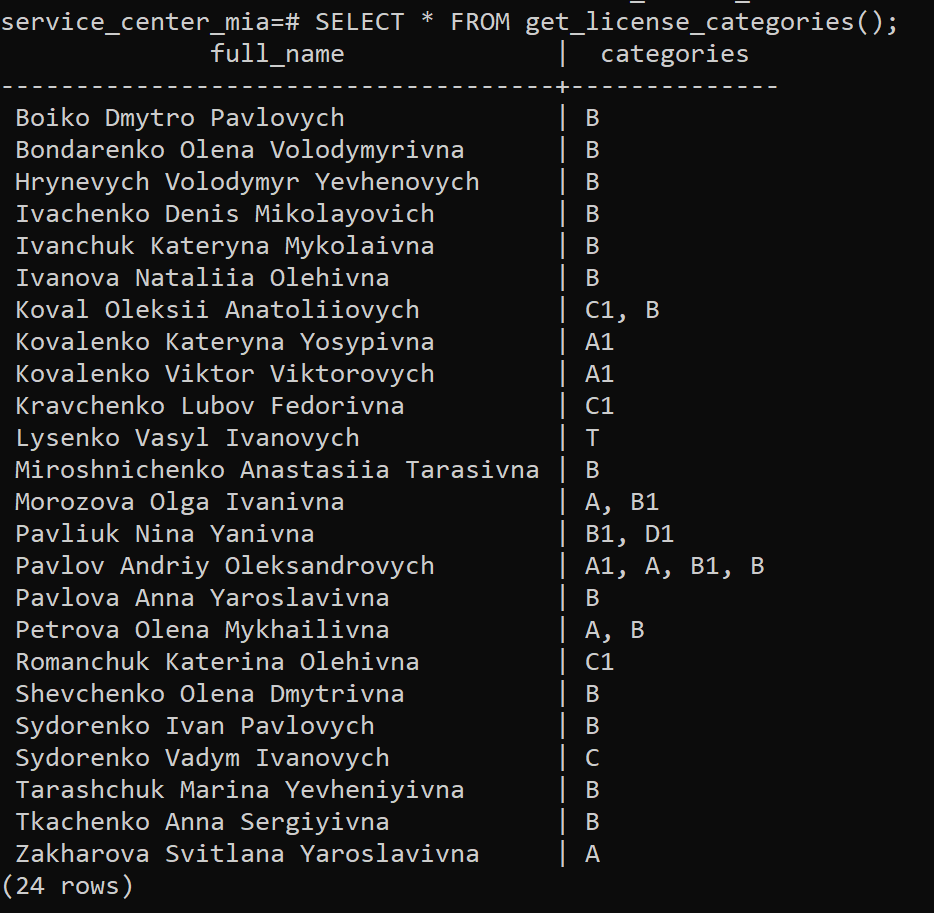


Рисунок 11.1 – Отримані кожним водієм категорії

11.2. Процедура для додання водійського посвідчення та категорії.

Ця процедура спрощує додання водійського посвідчення та інформації про відкриту категорію та коробку передач. Вона замінює 2 DML запити 1 викликом процедури із необхідними для вставки параметрами. Окрім того, ця процедура допомагає уникнути деяких помилок. Нижче наведено код процедури, а на рисунку 11.2 – результат виклику цієї функції.

CREATE OR REPLACE PROCEDURE add\_license\_and\_category(

in\_drivers\_license\_number VARCHAR(9),

in\_applicant\_id VARCHAR(10),

in\_obtaining\_date DATE,

in\_validity\_years INTEGER,

in\_category\_of\_transport VARCHAR(2),

in\_type\_of\_gearbox varchar(2))

AS $$ DECLARE v\_expiration\_date DATE;

BEGIN

v\_expiration\_date := in\_obtaining\_date + INTERVAL '1 YEAR' \* in\_validity\_years;

INSERT INTO drivers\_license (drivers\_license\_number, last\_name, first\_name, middle\_name, obtaining\_date, expiration\_date)

SELECT in\_drivers\_license\_number,

a.last\_name,

a.first\_name,

a.middle\_name,

in\_obtaining\_date,

v\_expiration\_date

FROM applicant a

WHERE a.applicant\_id = in\_applicant\_id

ON CONFLICT (drivers\_license\_number) DO NOTHING;

INSERT INTO received\_categories (category\_of\_transport, drivers\_license\_number, type\_of\_gearbox)

VALUES (in\_category\_of\_transport, in\_drivers\_license\_number, in\_type\_of\_gearbox)

ON CONFLICT (category\_of\_transport, drivers\_license\_number) DO NOTHING;

END;

$$ LANGUAGE plpgsql;

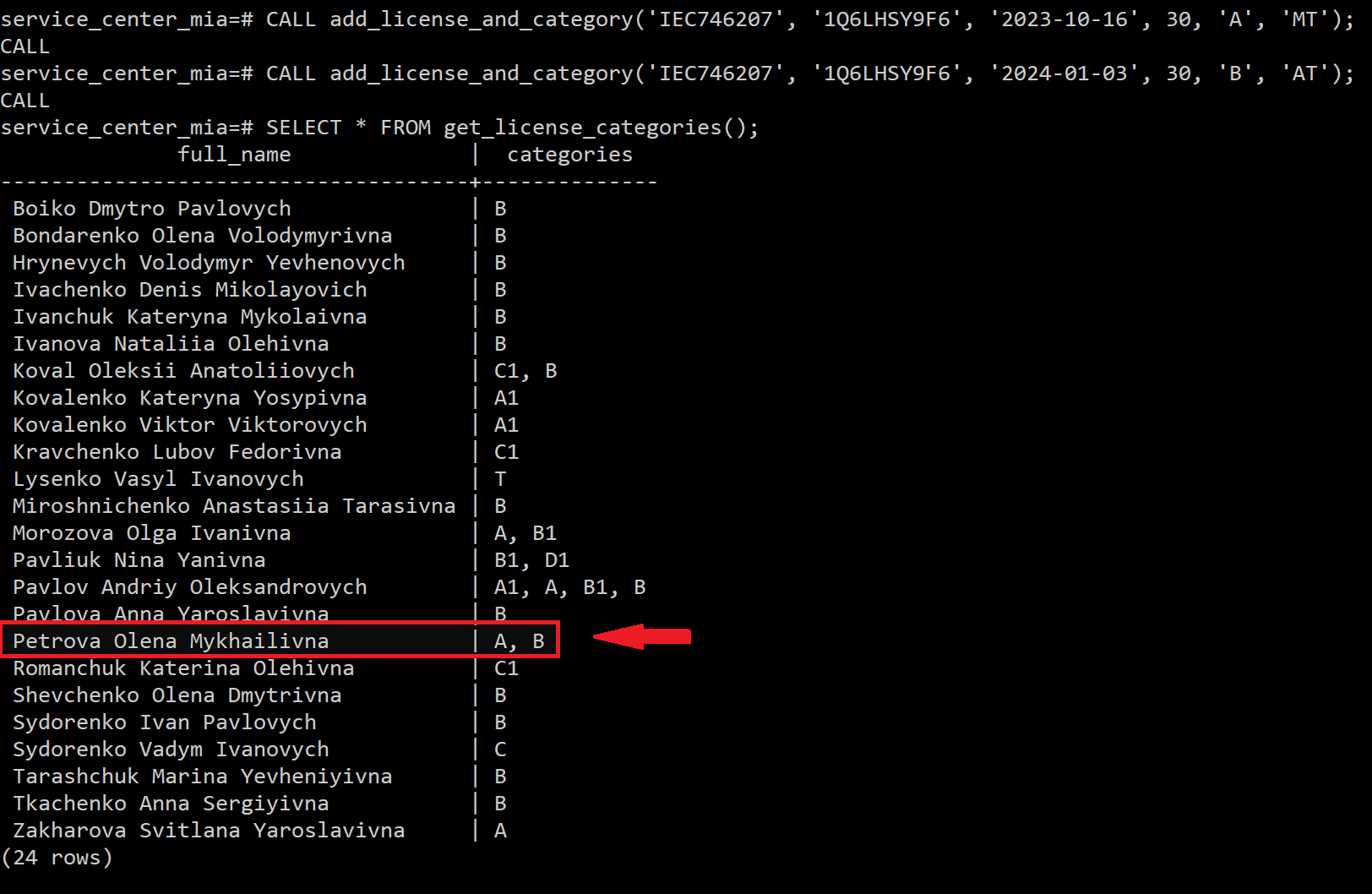


Рисунок 11.2 – Результат додання відкритої водієм категорії

11.3. Функція, що виводить водіїв, які здали хоча б n іспитів.

Ця функція виводить інформацію про тих водіїв, які здали практичний іспит не менше n разів. Для цього в функцію передається єдиний параметр – число n. Код функції наведено нижче, приклад використання – на рисунку 11.3.

CREATE OR REPLACE FUNCTION get\_applicants\_with\_n\_successful\_exams(n INTEGER)

RETURNS TABLE (

ID VARCHAR(10),

full\_name TEXT,

successful\_exams INTEGER)

AS $$

BEGIN

RETURN QUERY

SELECT a.applicant\_id AS ID,

CONCAT(a.last\_name, ' ', a.first\_name, ' ', a.middle\_name) AS full\_name,

CAST(ij.successful\_exam\_count AS INTEGER) AS successful\_exams

FROM applicant a

INNER JOIN (

SELECT pe.applicant\_id, COUNT(pe.exam\_serial\_number) AS successful\_exam\_count

FROM log\_of\_practical\_exams pe

WHERE pe.success\_marker = TRUE

GROUP BY pe.applicant\_id

HAVING COUNT(pe.exam\_serial\_number) >= n

) AS ij ON a.applicant\_id = ij.applicant\_id;

END;

$$ LANGUAGE plpgsql;

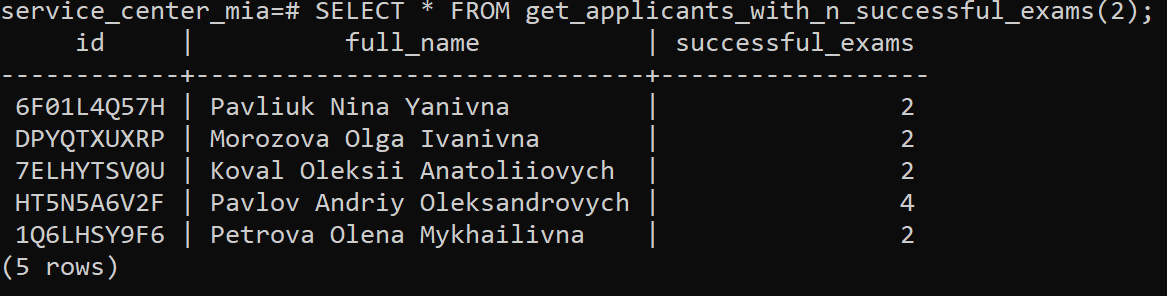


Рисунок 11.3 – Водії, що здали не менше 2 практичних іспитів

11.4. Функція, що перевіряє, чи вільний інспектор практичного іспиту в заданий час.

Ця функція повертає значення типу BOOLEAN, яке визначає, чи вільний інспектор практичного екзамену у вказаний час. Інспектор вважається вільним, якщо в межах 20 хвилин від заданого часу у нього немає активних записів на іспит. Якщо такий запис є – інспектор вважається зайнятим. Код функції наведено нижче, результати виклику – на рисунку 11.4 та рисунку 11.5.

CREATE OR REPLACE FUNCTION is\_practical\_inspector\_available(p\_inspector\_id VARCHAR(4), p\_exam\_date DATE, p\_exam\_time TIME)

RETURNS BOOLEAN AS $$

DECLARE temp\_start\_time TIME;

temp\_end\_time TIME;

marker BOOLEAN;

BEGIN

temp\_start\_time := p\_exam\_time - INTERVAL '20 minutes';

temp\_end\_time := p\_exam\_time + INTERVAL '20 minutes';

SELECT EXISTS (

SELECT 1

FROM log\_of\_practical\_exams lpe

WHERE lpe.date\_of\_passing = p\_exam\_date

AND lpe.time\_of\_passing BETWEEN temp\_start\_time AND temp\_end\_time

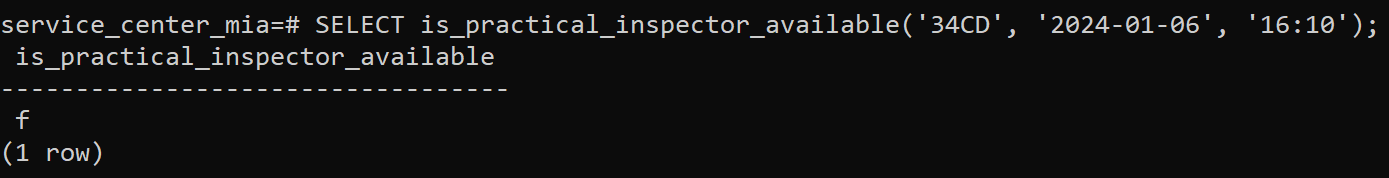
AND lpe.inspector\_id = p\_inspector\_id

) INTO marker;

RETURN NOT marker;

END;

$$ LANGUAGE plpgsql;



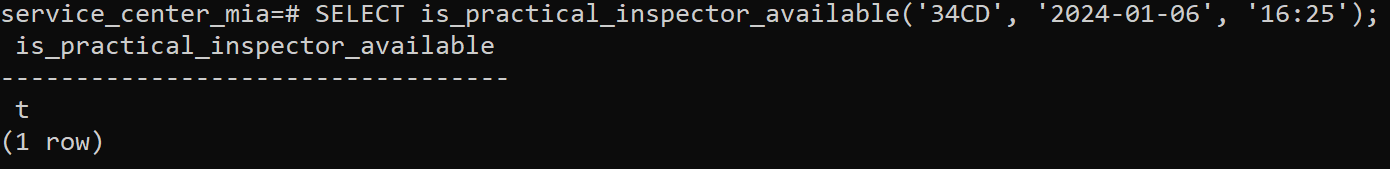
Рисунок 11.4 – Перевірка, чи зайнятий інспектор (зайнятий)

Рисунок 11.5 – Перевірка, чи зайнятий інспектор (не зайнятий)

11.5. Функція, що перевіряє, чи є можливість здати теоретичний іспит в заданий час.

Ця функція перевіряє, чи є місця в кабінеті для здачі теоретичного іспиту. Вільні місця є, якщо в межах 20 хвилин від заданого часу кількість записів на теоретичний іспит не перевищує 20 (тобто кількості комп’ютерів в кабінеті). Якщо ж перевищує, то функція повертає FALSE. Нижче наведено код функції, а на рисунку 11.6 показано результат її виклику.

CREATE OR REPLACE FUNCTION is\_theoretical\_inspector\_available(p\_exam\_date DATE, p\_exam\_time TIME)

RETURNS BOOLEAN AS $$

DECLARE temp\_start\_time TIME;

temp\_end\_time TIME;

amount INTEGER;

BEGIN

temp\_start\_time := p\_exam\_time - INTERVAL '20 minutes';

temp\_end\_time := p\_exam\_time + INTERVAL '20 minutes';

SELECT COUNT(\*)

INTO amount

FROM log\_of\_theoretical\_exams lte

WHERE lte.date\_of\_passing = p\_exam\_date

AND lte.time\_of\_passing BETWEEN temp\_start\_time AND temp\_end\_time;

RETURN amount < 21;

END;

$$ LANGUAGE plpgsql;

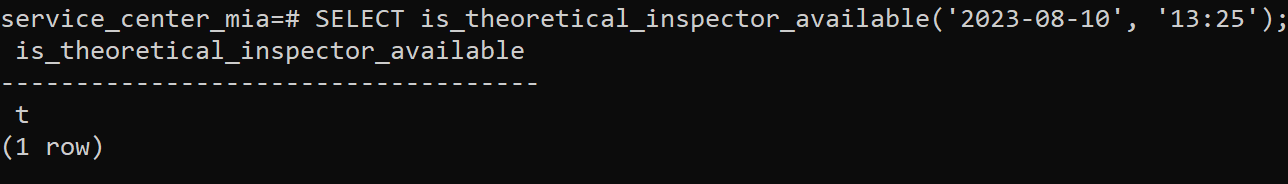


Рисунок 11.6 – Перевірка, чи є вільні місця на теоретичний іспит

11.6. Функція, що перевіряє, чи вільний транспорт в зазначений час.

Ця функція повертає значення типу BOOLEAN, яке визначає, чи вільний транспорт в заданий час. Транспорт вважається вільним, якщо в межах 20 хвилин на ньому не проводиться практичний іспит. Якщо ж іспит проводиться – функція повертає значення FASLE. Нижче наведено код функції, а на рисунку 11.7 та 11.8 показано результат виклику цієї функції.

CREATE OR REPLACE FUNCTION is\_transport\_available(p\_transport\_id VARCHAR(4), p\_exam\_date DATE, p\_exam\_time TIME)

RETURNS BOOLEAN AS $$

DECLARE temp\_start\_time TIME;

temp\_end\_time TIME;

marker BOOLEAN;

BEGIN

temp\_start\_time := p\_exam\_time - INTERVAL '20 minutes';

temp\_end\_time := p\_exam\_time + INTERVAL '20 minutes';

SELECT EXISTS (

SELECT 1

FROM log\_of\_practical\_exams lpe

WHERE lpe.date\_of\_passing = p\_exam\_date

AND lpe.time\_of\_passing BETWEEN temp\_start\_time AND temp\_end\_time

AND lpe.transport\_id = p\_transport\_id

) INTO marker;

RETURN NOT marker;

END;

$$ LANGUAGE plpgsql;

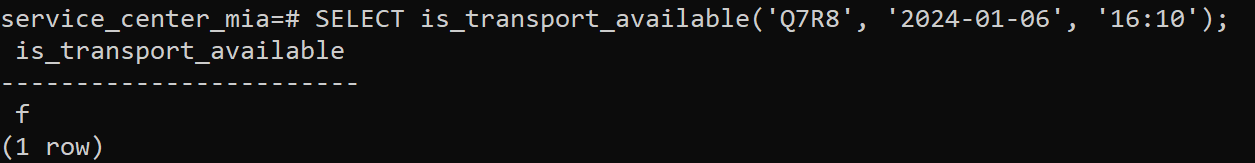


Рисунок 11.7 – Перевірка, чи зайнятий транспорт (зайнятий)

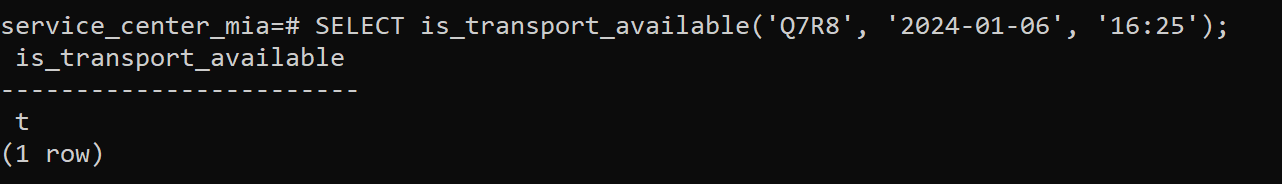


Рисунок 11.8 – Перевірка, чи зайнятий транспорт (не зайнятий)

|  |  |
| --- | --- |
|  |  |

11.7. Процедура для вставки практичного іспиту.

Ця процедура дозволяє полегшити вставку практичного іспиту. Вона виконує всі необхідні перевірки (доступність інспектору та транспорту, а також перевірку на допуск до проведення практичного іспиту). Код процедури наведено нижче, на рисунках 11.9, 11.10, 11.11 та 11.12 показано результати виклику цієї процедури.

CREATE OR REPLACE PROCEDURE insert\_practical\_exam(

p\_date DATE,

p\_time TIME,

p\_transport\_id VARCHAR(4),

p\_inspector\_id VARCHAR(4),

p\_applicant\_id VARCHAR(10))

AS $$ DECLARE v\_category\_of\_transport VARCHAR(2);

BEGIN

IF NOT is\_practical\_inspector\_available(p\_inspector\_id, p\_date, p\_time) THEN

RAISE EXCEPTION 'This inspector is not available!';

ELSIF NOT is\_transport\_available(p\_transport\_id, p\_date, p\_time) THEN

RAISE EXCEPTION 'This trasport is not available!';

ELSE SELECT category\_of\_transport INTO v\_category\_of\_transport FROM transport WHERE transport\_id = p\_transport\_id;

INSERT INTO log\_of\_practical\_exams (date\_of\_passing, time\_of\_passing, transport\_id, inspector\_id, applicant\_id, success\_marker, amount\_of\_mistakes, category\_of\_transport)

VALUES (p\_date, p\_time, p\_transport\_id, p\_inspector\_id, p\_applicant\_id, NULL, NULL, v\_category\_of\_transport);

RAISE NOTICE 'Inserted!';

END IF;

END;

$$ LANGUAGE plpgsql;

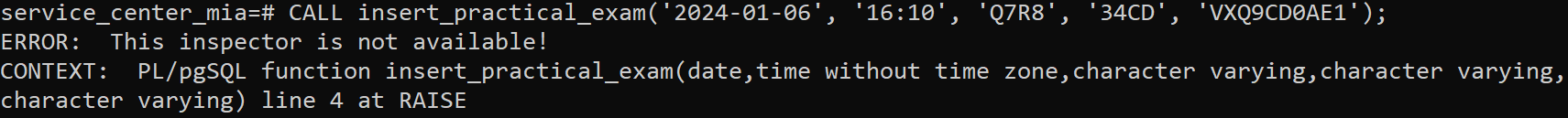


Рисунок 11.9 – Вставка практичного екзамену тоді, коли зайнятий інспектор

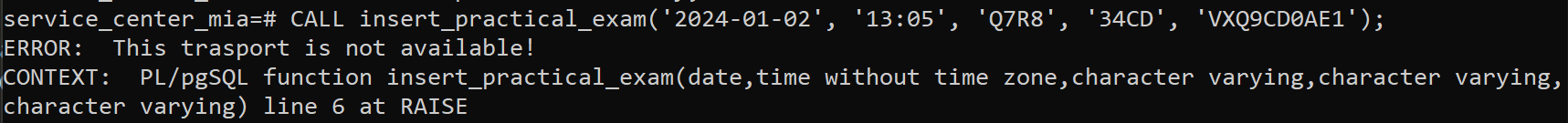


Рисунок 11.10 – Вставка практичного екзамену тоді, коли зайнятий транспорт

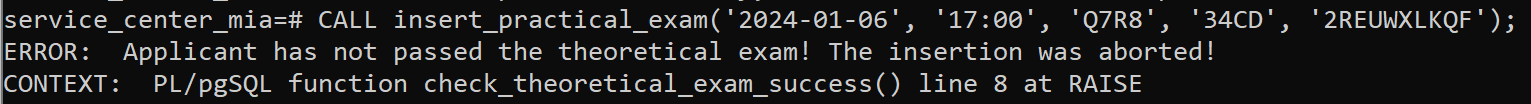


Рисунок 11.11 – Вставка практичного іспиту для тієї особи, яка не пройшла теоретичний іспит (спрацював тригер 8.1.)

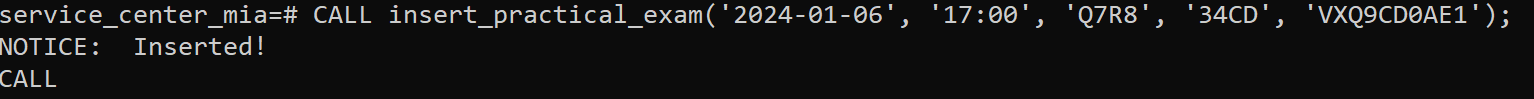


Рисунок 11.12 – Вставка практичного іспиту, коли дотримано всіх необхідних умов

11.8. Процедура для вставки теоретичного іспиту.

Ця процедура дозволяє полегшити вставку теоретичного іспиту. Вона виконує всі необхідні перевірки (перевірка, чи достатньо місць в кабінеті, чи має кандидат право на керування ТЗ). Код процедури наведено нижче, на рисунках 11.13 та 11.14 показано результати виклику цієї процедури.

CREATE OR REPLACE PROCEDURE insert\_theoretical\_exam(

p\_date DATE,

p\_time TIME,

p\_inspector\_id VARCHAR(4),

p\_applicant\_id VARCHAR(10))

AS $$ DECLARE v\_category\_of\_transport VARCHAR(2);

BEGIN

IF NOT is\_theoretical\_inspector\_available(p\_date, p\_time) THEN

RAISE EXCEPTION 'Too many appointments at this time! Choose another.';

ELSE INSERT INTO log\_of\_theoretical\_exams (date\_of\_passing, time\_of\_passing, applicant\_id, inspector\_id, success\_marker, exam\_grade)

VALUES (p\_date, p\_time, p\_inspector\_id, p\_applicant\_id, NULL, NULL);

RAISE NOTICE 'Inserted!';

END IF;

END;

$$ LANGUAGE plpgsql;

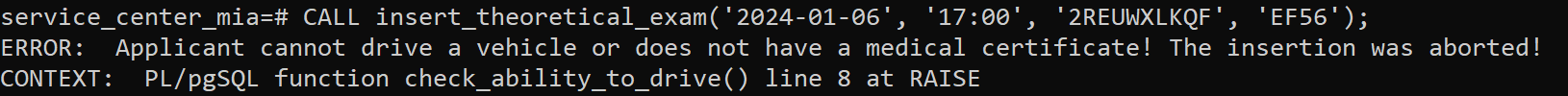


Рисунок 11.13 – Вставка практичного іспиту для тієї особи, яка не має права на керування ТЗ (спрацював тригер 8.2.)

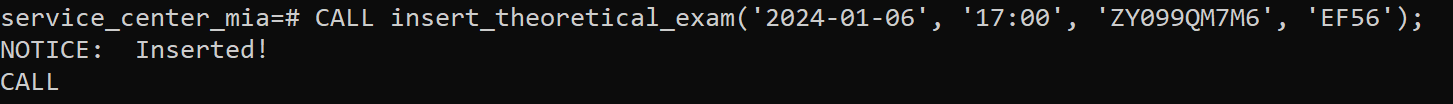


Рисунок 11.14 – Вставка теоретичного іспиту тій особі,

яка має право на керування ТЗ

11.9. Процедура зміни маркеру здатності керування ТЗ.

Ця процедура дозволяє полегшити процес зміни маркеру здатності керування транспортним засобом. Якщо людина більше не має права на керування – процедура видаляє посвідчення, отримані категорії, а також всі записи на теоретичний та практичний іспит, що дозволяє почистити базу даних від непотрібних записів. Код процедури наведено нижче, результат виклику – на рисунках 11.15 та 11.16.

CREATE OR REPLACE PROCEDURE update\_ability\_to\_drive(p\_person\_id VARCHAR(10), p\_new\_marker BOOLEAN)

AS $$ DECLARE

old\_marker BOOLEAN;

v\_last\_name VARCHAR(50);

v\_first\_name VARCHAR(50);

v\_middle\_name VARCHAR(50);

v\_drivers\_license\_number VARCHAR(9);

BEGIN

SELECT ability\_to\_drive\_marker FROM medical\_certificates WHERE person\_id = p\_person\_id INTO old\_marker;

IF old\_marker = p\_new\_marker THEN RAISE NOTICE 'Ability to drive was not changed';

ELSE UPDATE medical\_certificates

SET ability\_to\_drive\_marker = p\_new\_marker

WHERE person\_id = p\_person\_id;

IF p\_new\_marker = false THEN

-- Отримуємо ПІБ та номер посвідчення

SELECT last\_name, first\_name, middle\_name

INTO v\_last\_name, v\_first\_name, v\_middle\_name

FROM applicant

WHERE applicant\_id = p\_person\_id;

SELECT drivers\_license\_number

INTO v\_drivers\_license\_number

FROM drivers\_license

WHERE last\_name = v\_last\_name

AND first\_name = v\_first\_name

AND middle\_name = v\_middle\_name;

-- Видаляємо водійське посвічення

IF v\_drivers\_license\_number IS NOT NULL THEN

DELETE FROM drivers\_license WHERE drivers\_license\_number = v\_drivers\_license\_number;

DELETE FROM log\_of\_practical\_exams WHERE applicant\_id = p\_person\_id;

DELETE FROM log\_of\_theoretical\_exams WHERE applicant\_id = p\_person\_id;

END IF;

UPDATE applicant SET amount\_of\_attempts = 0 WHERE applicant\_id = p\_person\_id;

END IF;

END IF;

END;

$$ LANGUAGE plpgsql;

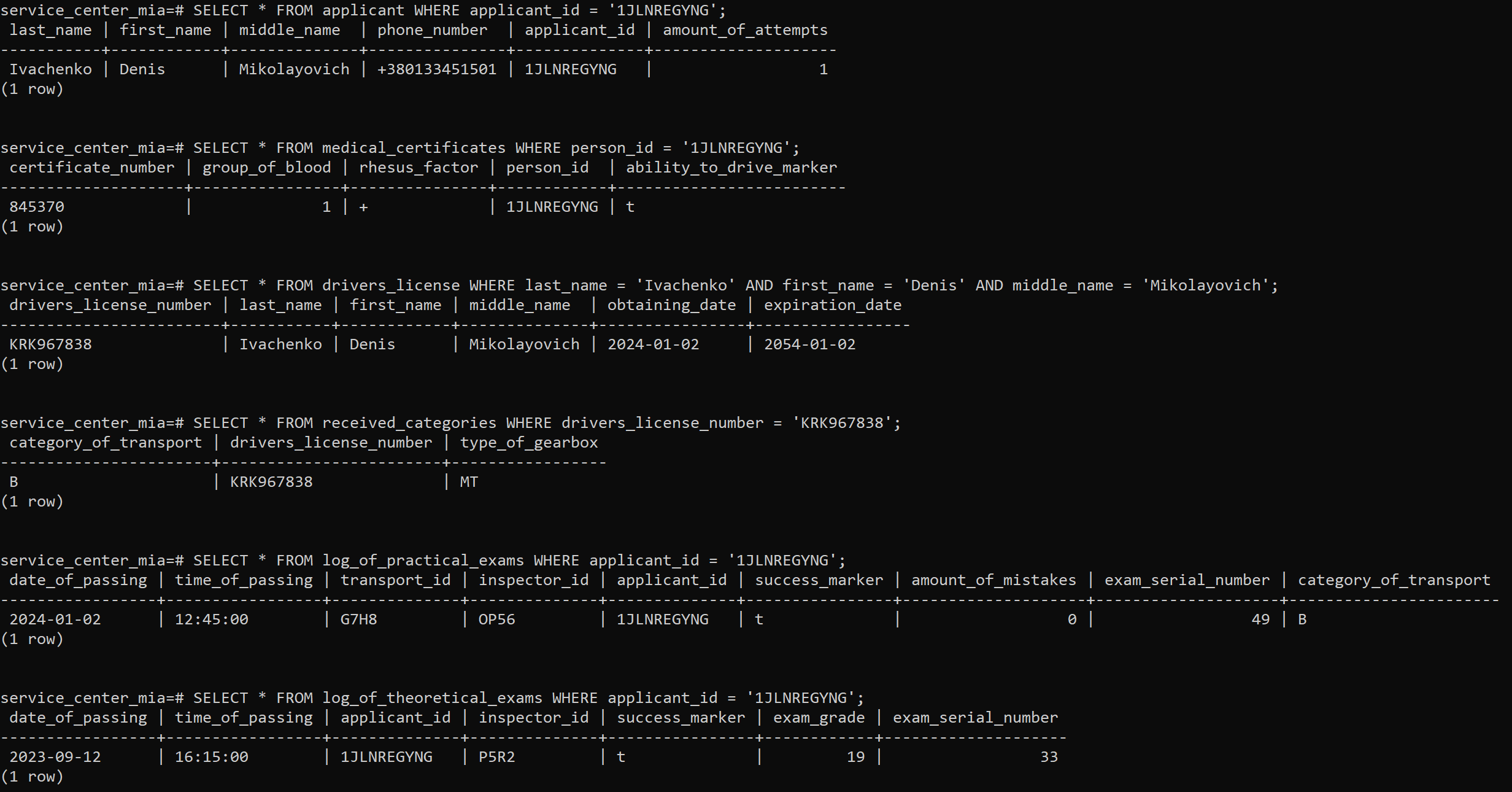


Рисунок 11.15 – Записи до визнання людини не здатною керувати ТЗ

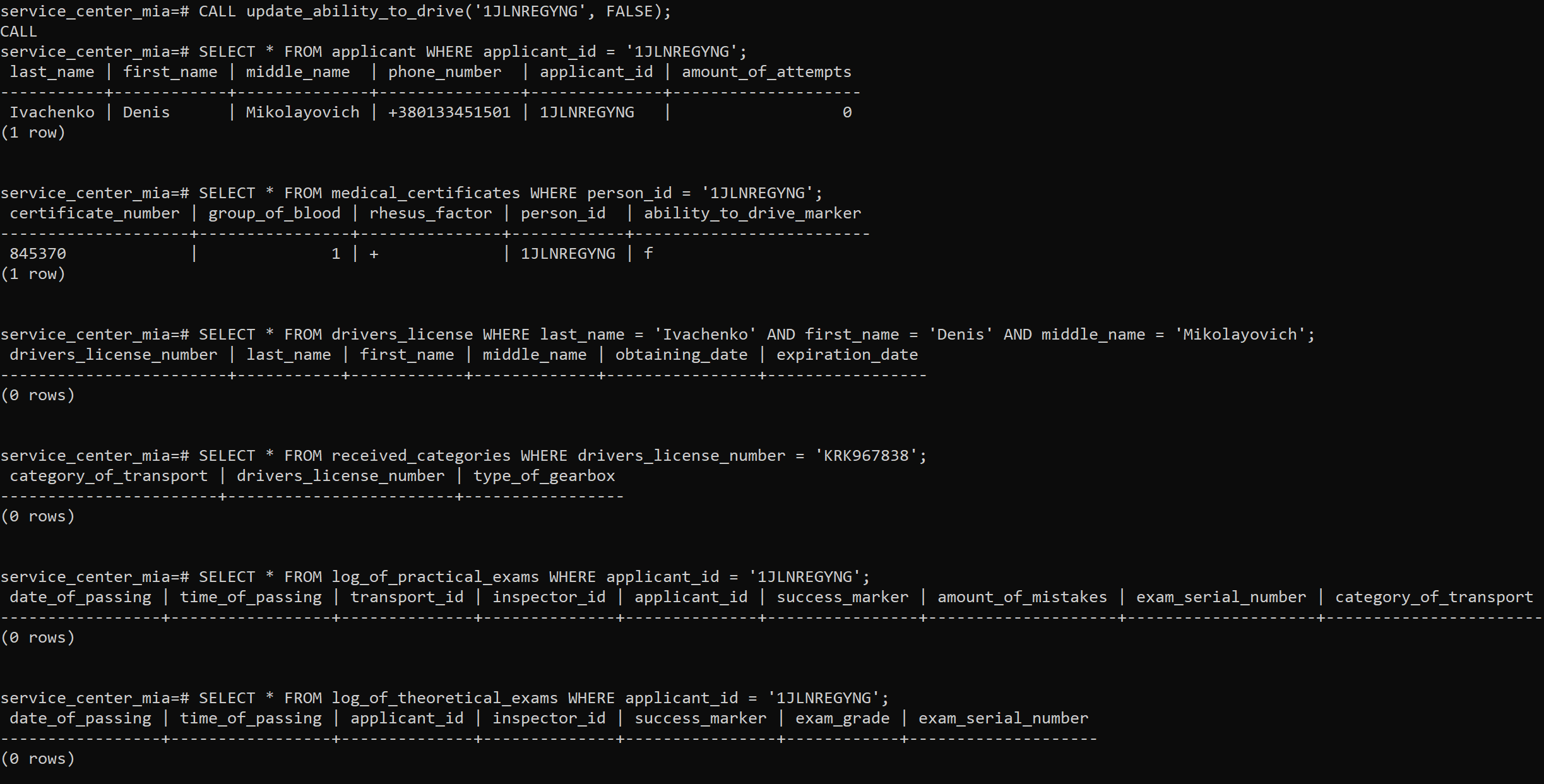


Рисунок 11.16 – Записи після визнання людини не здатною керувати ТЗ

11.10. Процедура підрахунку кількості запланованих екзаменів.

Ця процедура виводить точну кількість практичних та теоретичних екзаменів на поточний день. Код процедури наведено нижче, результат роботи – на рисунку 11.17.

CREATE OR REPLACE PROCEDURE report\_exams\_today()

AS $$ DECLARE

practical\_exam\_count INTEGER;

theoretical\_exam\_count INTEGER;

BEGIN

SELECT COUNT(\*)

INTO practical\_exam\_count

FROM log\_of\_practical\_exams

WHERE date\_of\_passing = CURRENT\_DATE;

SELECT COUNT(\*)

INTO theoretical\_exam\_count

FROM log\_of\_theoretical\_exams

WHERE date\_of\_passing = CURRENT\_DATE;

RAISE NOTICE 'Practical exams today: %', practical\_exam\_count;

RAISE NOTICE 'Theoretical exams today: %', theoretical\_exam\_count;

END;

$$ LANGUAGE plpgsql;

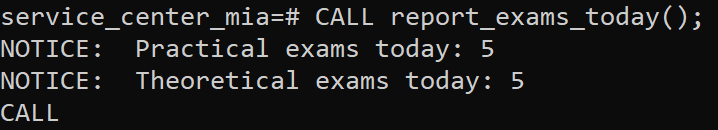


Рисунок 11.17 – Кількість екзаменів для системної дати 06.01.2024

11.11. Процедура видалення непроведених практичних іспитів.

Ця процедура видаляє всі практичні іспити, які так і не було проведено. Дата таких іспитів має бути меншою за поточну, а маркер успішності – NULL. Код реалізації такої логіки наведено нижче, результат – на рисунках 11.18, 11.19 та 11.20.

CREATE OR REPLACE PROCEDURE delete\_unconducted\_practical\_exams()

AS $$

BEGIN

DELETE FROM log\_of\_practical\_exams

WHERE success\_marker IS NULL AND date\_of\_passing < CURRENT\_DATE;

RAISE NOTICE 'Unconducted practical exams have been deleted';

END;

$$ LANGUAGE plpgsql;

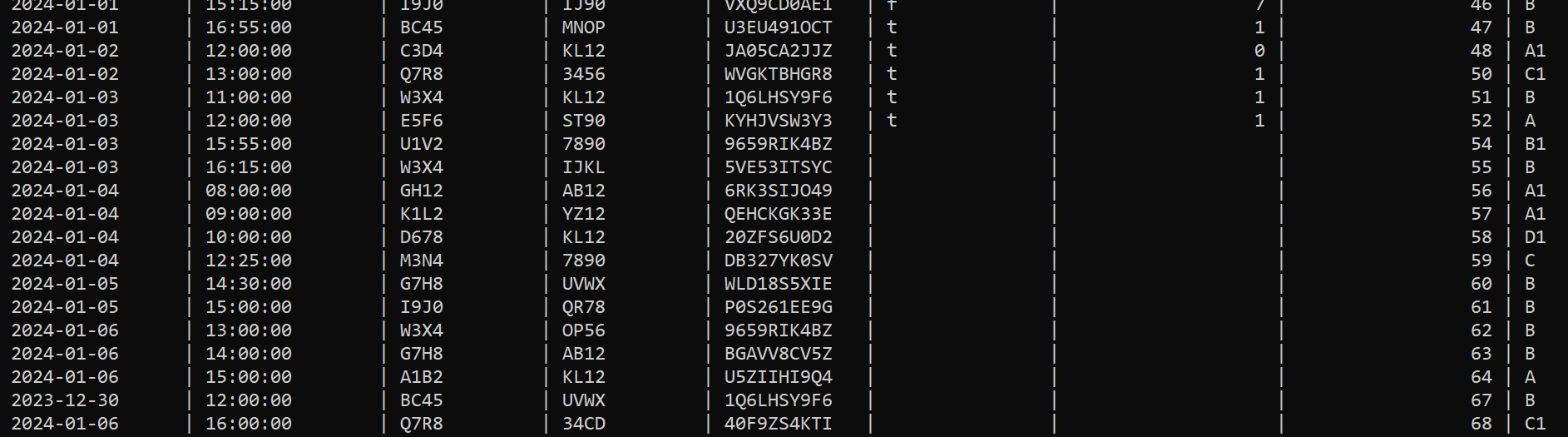


Рисунок 11.18 – Не проведені практичні іспити, що зберігаються у базі даних

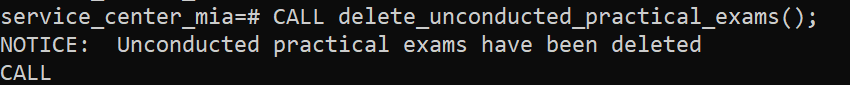


Рисунок 11.19 – Виклик процедури

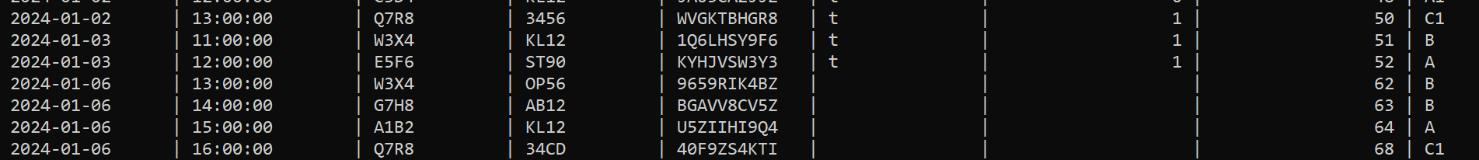


Рисунок 11.20 – Записи на практичний іспит після видалення непроведених

(системна дата – 06.01.2024)

11.12. Процедура видалення непроведених теоретичних іспитів.

Принцип роботи цієї процедури дуже схожий на принцип роботи процедури 11.11. Ця ж процедура видаляє всі теоретичні іспити, які так і не було проведено. Дата таких іспитів має бути меншою за поточну, а маркер успішності – NULL. Код реалізації такої логіки наведено нижче, результат – на рисунках 11.21, 11.22 та 11.23.

CREATE OR REPLACE PROCEDURE delete\_unconducted\_theoretical\_exams()

AS $$

BEGIN

DELETE FROM log\_of\_theoretical\_exams

WHERE success\_marker IS NULL AND date\_of\_passing < CURRENT\_DATE;

RAISE NOTICE 'Unconducted theoretical exams have been deleted';

END;

$$ LANGUAGE plpgsql;

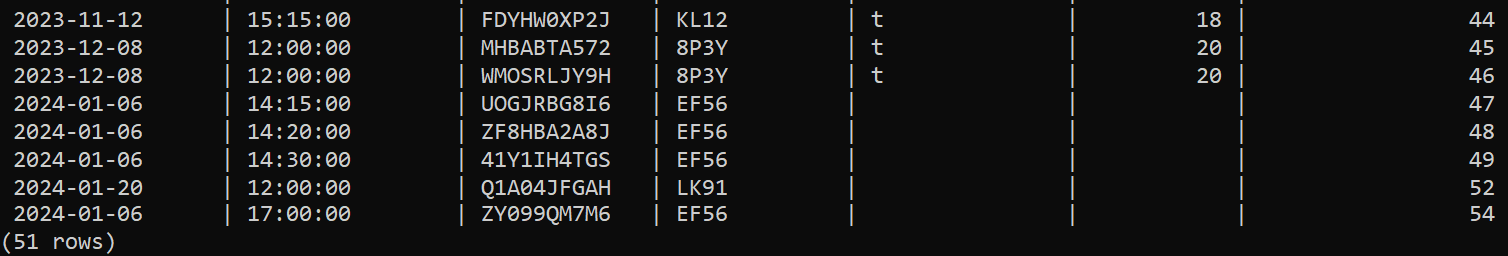


Рисунок 11.21 – Не проведені теоретичні іспити, що зберігаються у базі даних

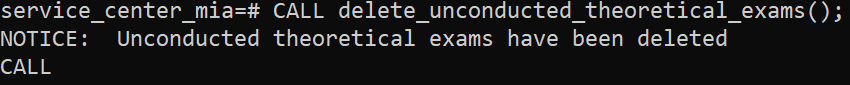


Рисунок 11.22 – Виклик процедури

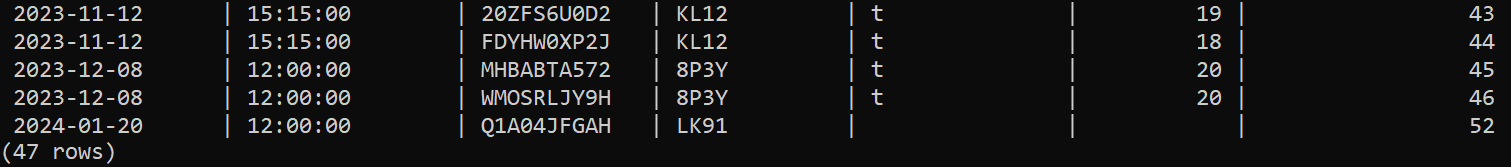


Рисунок 11.23 – Записи на теоретичний іспит після видалення непроведених

(системна дата – 07.01.2024)

# ОПТИМІЗАЦІЯ

Процес покращення продуктивності та ефективності роботи бази даних є важливим елементом її реалізації. Це особливо важливо в сучасних реаліях, де великі обсяги даних і велика кількість одночасних запитів можуть призводити до затримок у відповіді системи. Оптимізація спрямована на зменшення часу виконання запитів, використання ресурсів і підвищення продуктивності.

Одним із шляхів оптимізації роботи бази даних – це використання індексів. Нижче наведений код створення індексів в базі даних service\_center\_mia.

CREATE INDEX idx\_practical\_exam\_inspector\_id ON practical\_exam\_inspector(inspector\_id);

CREATE INDEX idx\_theoretical\_exam\_inspector\_id ON theoretical\_exam\_inspector(inspector\_id);

CREATE INDEX idx\_applicant\_id ON applicant(applicant\_id);

CREATE INDEX idx\_transport\_id ON transport(transport\_id);

CREATE INDEX idx\_log\_practical\_exam\_serial\_number ON log\_of\_practical\_exams(exam\_serial\_number);

CREATE INDEX idx\_log\_theoretical\_exam\_serial\_number ON log\_of\_theoretical\_exams(exam\_serial\_number);

CREATE INDEX idx\_medical\_certificate\_number ON medical\_certificates(certificate\_number);

CREATE INDEX idx\_drivers\_license\_number ON drivers\_license(drivers\_license\_number);

Важливо додати, що не завжди індекси покращують роботу, а іноді навіть сповільнюють її. Таке може статися, коли індексацію поставлено на поля, що мають багато повторюваних значень, або на поля, які дуже часто оновлюються. Завелика кількість індексів теж може негативно впливати на оптимізацію.

Вплив створених індексів слід протестувати. Нижче будуть наведені 3 різнотипні запити. Завдяки ним можна протестувати середню зміну у швидкості роботи бази даних. Для підрахунку часу виконання використаємо «EXPLAIN ANALYZE» перед запитом.

12.1. Аналіз впливу індексів на першому запиті.

Код першого запиту наведено нижче. В таблиці 12.1 наведено час виконання запиту для 5 випробувань до і після додання індексів. На рисунку 12.1 та 12.2 показано схеми запитів та час перших випробувань.

EXPLAIN ANALYZE

SELECT pi.last\_name || ' ' || pi.first\_name AS inspector\_name,

ROUND(AVG(pe.amount\_of\_mistakes), 2) AS average\_mistakes

FROM practical\_exam\_inspector pi

INNER JOIN log\_of\_practical\_exams pe ON pi.inspector\_id = pe.inspector\_id

GROUP BY pi.inspector\_id, pi.last\_name, pi.first\_name;

Таблиця 12.1 – Аналіз впливу індексів на першому запиті

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер спроби | 1 | 2 | 3 | 4 | 5 |
| Час виконання до додання індексу, мс | 0.147 | 0.136 | 0.135 | 0.125 | 0.126 |
| Час виконання після додання індексу, мс | 0.117 | 0.121 | 0.120 | 0.114 | 0.110 |

Середнє значення перед доданням індексів = 0.134 мс.

Середнє значення після додання індексів = 0.116 мс.

Отже, з індексами час, витрачений на запит, зменшився на 14%.

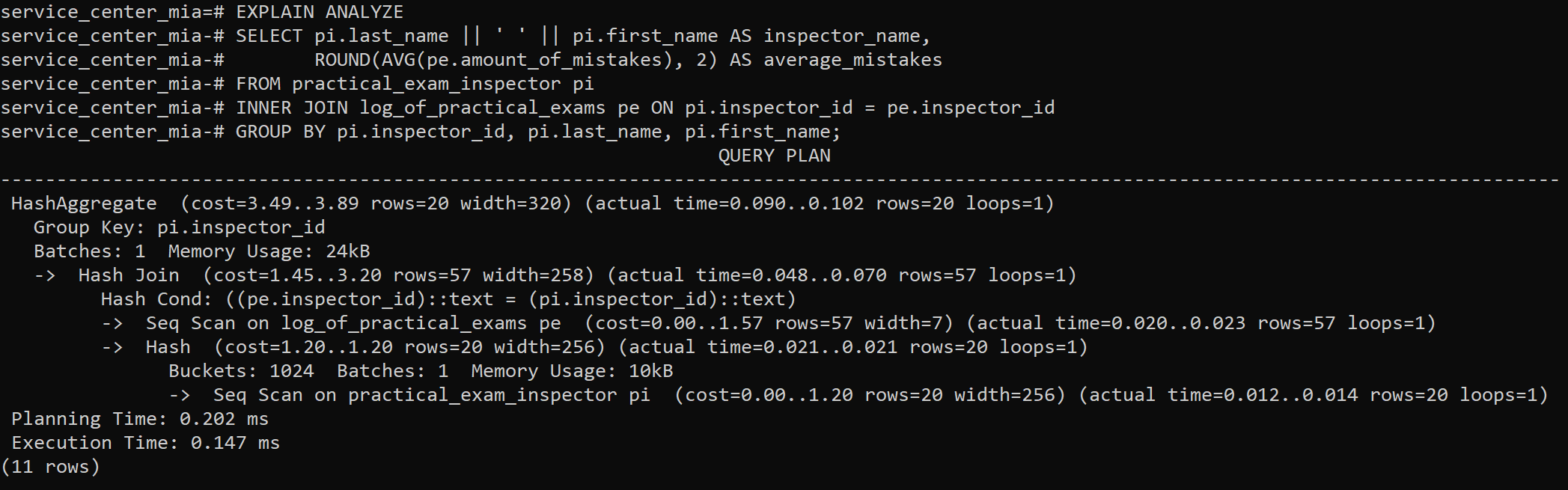


Рисунок 12.1 – Схема запиту 1 до додання індексів



Рисунок 12.2 – Схема запиту 1 після додання індексів

12.2. Аналіз впливу на другому запиті.

Код другого запиту наведено нижче. В таблиці 12.2 наведено час виконання запиту для 5 випробувань до і після додання індексів. На рисунку 12.3 та 12.4 показано схеми запитів та час перших випробувань.

EXPLAIN ANALYZE

WITH SuccessfulExams AS (

SELECT inspector\_id, COUNT(\*) AS successful\_exams\_count

FROM log\_of\_practical\_exams

WHERE success\_marker = TRUE

GROUP BY inspector\_id

), AverageSuccessfulExams AS (

SELECT AVG(successful\_exams\_count) AS avg\_successful\_exams

FROM SuccessfulExams

)

SELECT pi.last\_name, pi.first\_name, pi.middle\_name, sec.successful\_exams\_count AS successful\_exams, ROUND(ase.avg\_successful\_exams, 2) AS average

FROM practical\_exam\_inspector pi

JOIN SuccessfulExams sec ON pi.inspector\_id = sec.inspector\_id

CROSS JOIN AverageSuccessfulExams ase

WHERE sec.successful\_exams\_count > ase.avg\_successful\_exams;

Таблиця 12.2 – Аналіз впливу на другому запиті

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер спроби | 1 | 2 | 3 | 4 | 5 |
| Час виконання до додання індексу, мс | 0.197 | 0.174 | 0.113 | 0.122 | 0.124 |
| Час виконання після додання індексу, мс | 0.109 | 0.129 | 0.121 | 0.124 | 0.090 |

Середнє значення перед доданням індексів = 0.146 мс.

Середнє значення після додання індексів = 0.115 мс.

Отже, з індексами час, витрачений на запит, зменшився на 20%.

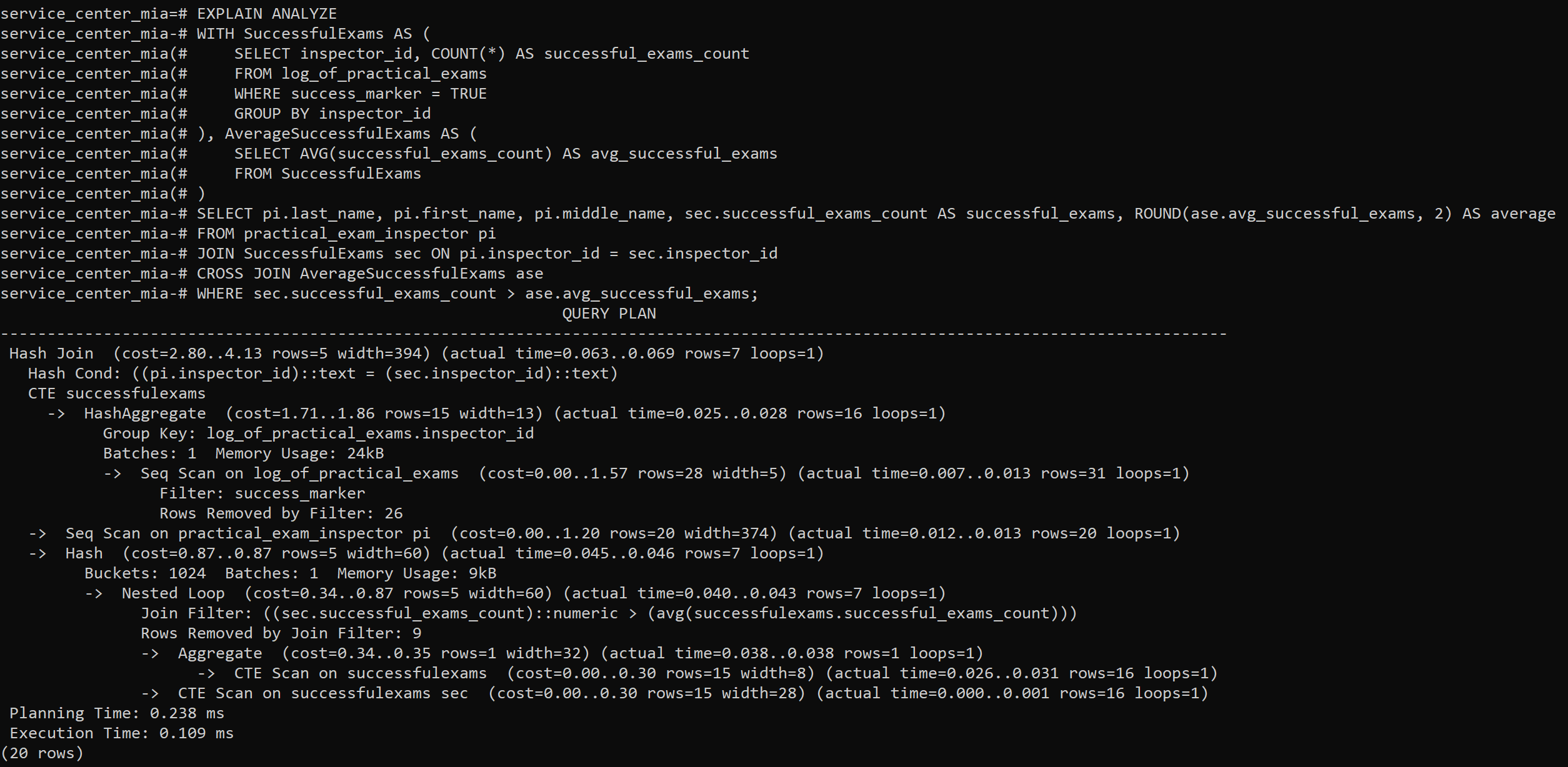


Рисунок 12.3 – Схема запиту 2 до додання індексів

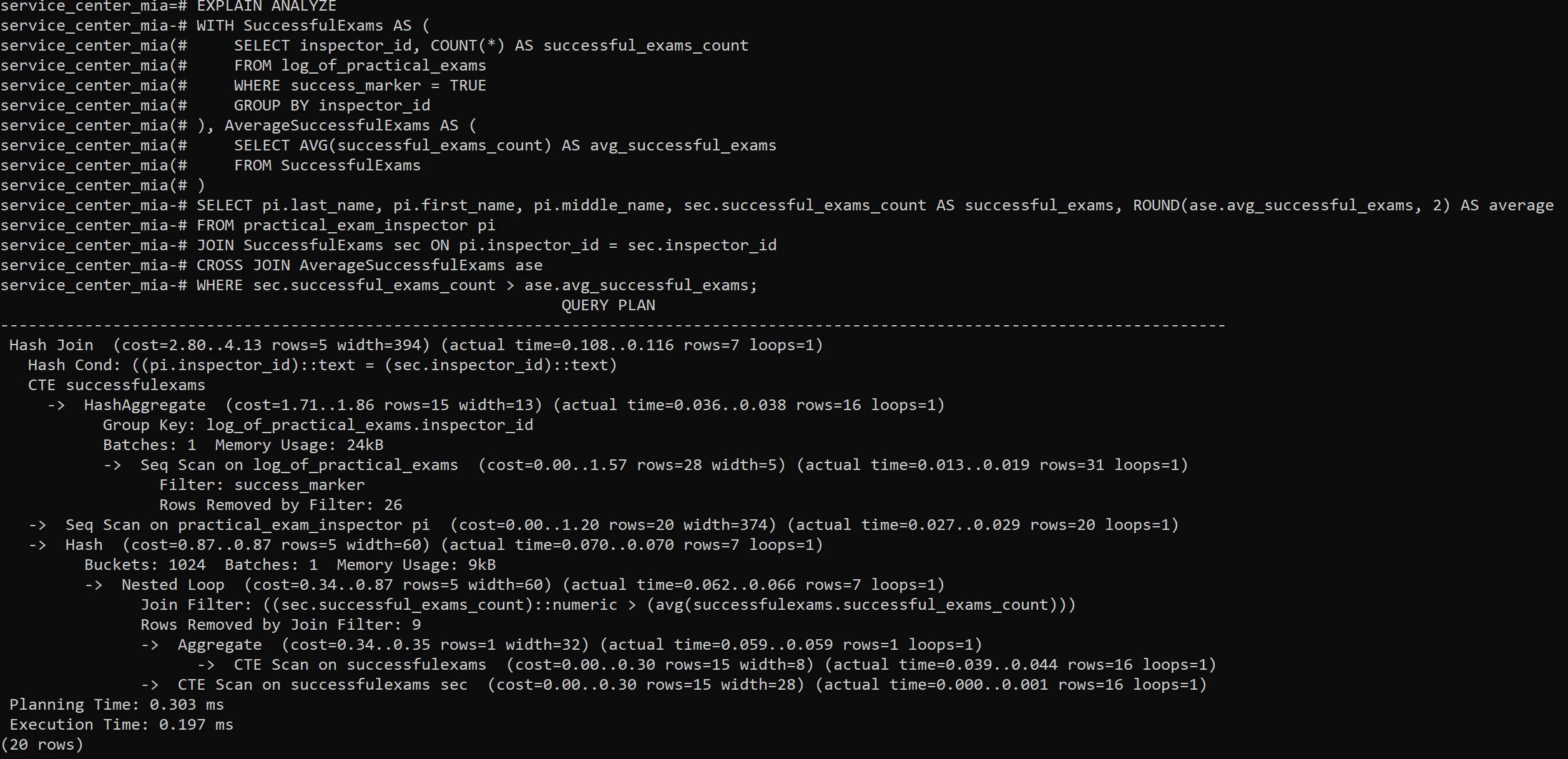


Рисунок 12.4 – Схема запиту 2 після додання індексів

12.3. Аналіз впливу на третьому запиті.

Код третього запиту наведено нижче. В таблиці 12.3 наведено час виконання запиту для 5 випробувань до і після додання індексів. На рисунку 12.5 та 12.6 показано схеми запитів та час перших випробувань.

CREATE TEMP TABLE temp\_variables AS

SELECT 'Truck' AS type\_of\_transport;

EXPLAIN ANALYZE

SELECT a.last\_name, a.first\_name, a.middle\_name, subquery.category\_of\_transport, tv.type\_of\_transport

FROM applicant a

JOIN (

SELECT le.applicant\_id, t.category\_of\_transport, t.purpose\_of\_transport

FROM log\_of\_practical\_exams le

JOIN transport t ON le.transport\_id = t.transport\_id

) AS subquery ON a.applicant\_id = subquery.applicant\_id

JOIN temp\_variables tv ON subquery.purpose\_of\_transport = tv.type\_of\_transport;

DROP TABLE temp\_variables;

Таблиця 12.3 – Аналіз впливу на третьому запиті

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер спроби | 1 | 2 | 3 | 4 | 5 |
| Час виконання до додання індексу, мс | 0.177 | 0.128 | 0.143 | 0.114 | 0.172 |
| Час виконання після додання індексу, мс | 0.111 | 0.130 | 0.113 | 0.116 | 0.121 |

Середнє значення перед доданням індексів = 0.147 мс.

Середнє значення після додання індексів = 0.118 мс.

Отже, з індексами час, витрачений на запит, зменшився на 20%.

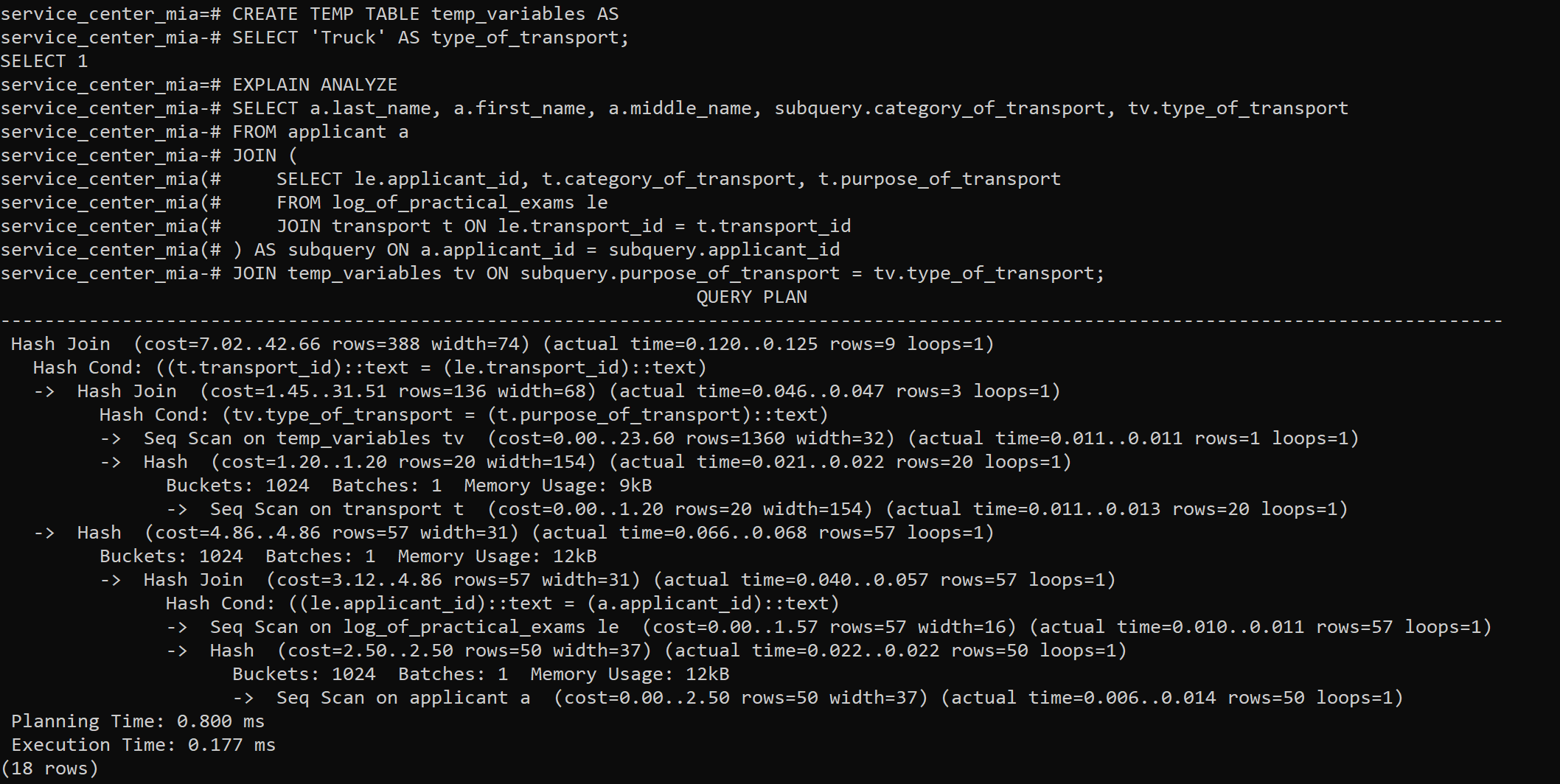


Рисунок 12.5 – Схема запиту 3 до додання індексів

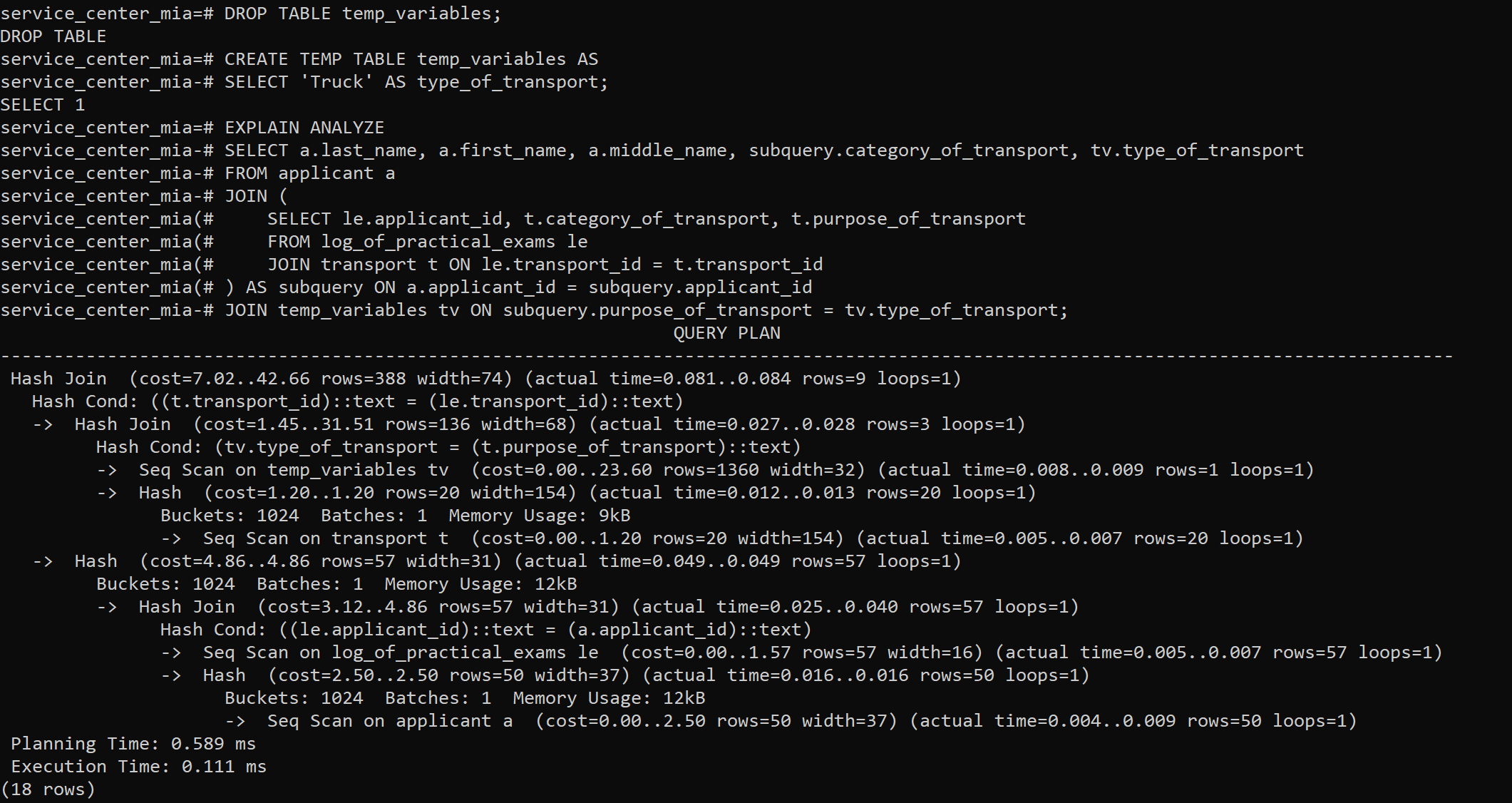


Рисунок 12.6 – Схема запиту 3 після додання індексів

Отже, введення індексації дещо покращило роботу з базою даних, зменшення часу виконання запитів становить до 20%.

# ВИСНОВКИ

У ході виконання курсової роботи було проведено аналіз предметної області, аналіз бізнес-процесів, аналіз вже існуючих програмних продуктів, розроблено ER модель бази даних. На основі цього аналізу розроблено базу даних засобами діалекту PostgreSQL, проведено роботу із базою даних.

Аналіз предметної області дозволив змоделювати базу даних під конкретні умови та потреби, був застосований для розуміння бізнес-процесів, визначення вимог користувачів, формування концептуальної моделі. Аналіз предметної області потрібен для забезпечення відповідності нормативам та стандартам тієї предметної області, для якої розробляється програмне забезпечення.

Розуміння бізнес-процесів дозволило визначити які саме завдання має вирішувати база даних у конкретному бізнес-середовищі. Визначення ключових бізнес-процесів допомагає точно визначити потрібні дані та їх структуру.

Вимоги користувачів продиктовані реальними процесами тієї області, для якої розроблено дане ПЗ. Обмеження доступу гарантує цілісність даних, захист конфіденційності, запобігає витоку даних.

Розроблена ER діаграма спрощує розробку бази даних та дозволяє розробити її правильно, згідно всім потребам. Діаграма містить ключові сутності та їхню взаємодію.

Розроблення бази даних та робота з нею включають програмну реалізацію бази даних, створення тригерів, збережуваних функцій та процедур, представлень, запитів. Запити забезпечують ефективну взаємодію користувача із базою даних, тригери дозволяють забезпечити правильну роботу бази даних, функції та процедури – спростити її.

Розроблена база даних відповідає сформованим вимогам, забезпечує роботу сервісного центру МВС в частині прийому екзаменів, водійських посвідчень. База даних здатна не тільки забезпечити ефективну роботу, а й спростити її для всіх учасників процесу прийому екзаменів та отримання водійських посвідчень.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. PostgreSQL 16.1 Documentation. PostgreSQL. URL: <https://www.postgresql.org/docs/16/index.html> (date of access: 29.12.2023).
2. PostgreSQL Tutorial. Basic PostgreSQL Tutorial. URL: <https://www.postgresqltutorial.com/> (date of access: 29.12.2023).
3. 1. PostgreSQL: how to write a trigger. Cybertec. URL: <https://www.cybertec-postgresql.com/en/postgresql-how-to-write-a-trigger/?gclid=CjwKCAiA-bmsBhAGEiwAoaQNmhaZCtF1KjIFP6hnvQLb17NYcr42-p3LqrYRVh6Gefs9uAFv_2UmNBoC2JcQAvD_BwE> (date of access: 29.12.2023).
4. Everything you need to know about Postgres stored procedures and functions. EnterpriseDB. URL: <https://www.enterprisedb.com/postgres-tutorials/everything-you-need-know-about-postgres-stored-procedures-and-functions> (date of access: 29.12.2023).
5. PostgreSQL - CONSTRAINTS. TutorialsPoint. URL: <https://www.tutorialspoint.com/postgresql/postgresql_constraints.htm> (date of access: 29.12.2023).
6. Postgres views. Introduction. Hasura. URL: <https://hasura.io/docs/latest/schema/postgres/postgres-guides/views/#:~:text=A%20Postgres%20view%20is%20a,directly%20queried%20via%20the%20same> (date of access: 29.12.2023).
7. Crow's foot notation – relationship symbols and how to read diagrams. FreeCodeCamp. URL: <https://www.freecodecamp.org/news/crows-foot-notation-relationship-symbols-and-how-to-read-diagrams/> (date of access: 29.12.2023).

# ДОДАТОК А КОД ВСТАВКИ ДАНИХ ЗА ДОПОМОГОЮ ЗАПИТІВ ТИПУ INSERT

Заповнення бази даних можна організувати за допомогою DML запитів типу INSERT. Нижче наведені запити для заповнення тестовими даними бази даних service\_center\_mia.

-- Вставка категорій транспорту --

INSERT INTO category\_of\_transport (category\_of\_transport)

VALUES ('A'),

('A1'),

('B'),

('B1'),

('C'),

('C1'),

('D'),

('D1'),

('T');

-- Вставка типів транспорту --

INSERT INTO type\_of\_transport (type\_of\_transport)

VALUES ('Motorcycle'),

('Moped'),

('Scooter'),

('Chopper'),

('Passenger car'),

('Truck'),

('Lorry'),

('ATV'),

('Bus'),

('Minibus'),

('Tram'),

('Trolleybus');

-- Вставка транспорту --

INSERT INTO transport (license\_plate\_number, purpose\_of\_transport, category\_of\_transport, year\_of\_manufacture, transport\_id, affiliation\_marker, type\_of\_gearbox)

VALUES ('AA1234BB', 'Motorcycle', 'A', '2010', 'A1B2', '1', 'AT'),

('BC5678DE', 'Moped', 'A1', '2015', 'C3D4', '1', 'MT'),

('KM9101OM', 'Chopper', 'A', '2016', 'E5F6', '0', 'MT'),

('IT1122IT', 'Passenger car', 'B', '2008', 'G7H8', '1', 'MT'),

('VE1313AK', 'Passenger car', 'B', '2021', 'I9J0', '1', 'AT'),

('NN1414NN', 'Scooter', 'A1', '2022', 'K1L2', '0', 'MT'),

('LO1515LO', 'Truck', 'C', '2004', 'M3N4', '1', 'MT'),

('AI1616IA', 'Motorcycle', 'A', '2010', 'O5P6', '0', 'MT'),

('UP1717UR', 'Truck', 'C1', '2009', 'Q7R8', '1', 'MT'),

('ST1818TS', 'Lorry', 'C', '2020', 'S9T0', '1', 'MT'),

('LA1929AK', 'ATV', 'B1', '2018', 'U1V2', '1', 'MT'),

('RI2020RI', 'Passenger car', 'B', '2003', 'W3X4', '0', 'AT'),

('KO2121KO', 'Motorcycle', 'A', '2016', 'Y5Z6', '0', 'MT'),

('MM2222MM', 'Bus', 'D', '2011', '123A', '1', 'MT'),

('AP2323AR', 'Passenger car', 'B', '2018', 'BC45', '0', 'MT'),

('II2424II', 'Minibus', 'D1', '2019', 'D678', '1', 'MT'),

('VO2525VO', 'Tram', 'T', '1985', 'EF90', '1', NULL),

('TT2626TT', 'Moped', 'A1', '2021', 'GH12', '1', 'MT'),

('NN2727NN', 'Trolleybus', 'T', '2004', 'IJKL', '1', 'AT'),

('SS2828SS', 'Truck', 'B', '2006', 'MNOP', '0' , 'MT');

-- Вставка інспекторів з теорії --

INSERT INTO theoretical\_exam\_inspector (last\_name, first\_name, middle\_name, phone\_number, inspector\_id)

VALUES ('Kovalenko', 'Andriy', 'Ivanovych', '+380963229935', 'QW29'),

('Hrytsenko', 'Olena', 'Mykhailivna', '+380675233153', 'J7K1'),

('Morozov', 'Serhiy', 'Petrovych', '+380507880730', 'P5R2'),

('Shevchenko', 'Kateryna', 'Oleksandrivna', '+380981496572', '6VZX'),

('Lysenko', 'Mykola', 'Dmytrovych', '+380631998321', 'LK91'),

('Tkachenko', 'Iryna', 'Yuriivna', '+380963865449', '8P3Y'),

('Ivanov', 'Vadim', 'Oleksiyovych', '+380501234567', 'AB12'),

('Zaharchenko', 'Dmitriy', 'Igorevich', '+380509376559', '34CD'),

('Petrenko', 'Lilia', 'Sergiyivna', '+380630340402', 'EF56'),

('Romanchuk', 'Olena', 'Pavlivna', '+380980454768', 'GH78'),

('Hrynevych', 'Volodymyr', 'Pavlovich', '+380960704741', 'IJ90'),

('Kravchuk', 'Marina', 'Igorivna', '+380972450381', 'KL12'),

('Shevchenko', 'Dmytro', 'Yuriyovich', '+380954962707', 'MN34'),

('Hryhorenko', 'Maria', 'Yuriivna', '+380973986092', 'OP56'),

('Zakharchenko', 'Vadim', 'Ostapovich', '+380963551664', 'QR78'),

('Petrenko', 'Iryna', 'Mykolaivna', '+380921218565', 'ST90'),

('Sydorenko', 'Pavlo', 'Ivanovych', '+380686466885', '9Z01'),

('Romanchuk', 'Oleksandra', 'Sergiyivna', '+380934939557', '4567'),

('Berezin', 'Yuriy', 'Anatoliyovych', '+3806897627967', '3V92'),

('Hrynevych', 'Nazar', 'Sergiyovych', '+380508998794', 'LK92');

-- Вставка інспекторів з практики --

INSERT INTO practical\_exam\_inspector (last\_name, first\_name, middle\_name, phone\_number, inspector\_id)

VALUES ('Kravchuk', 'Kateryna', 'Dmytrivna', '+380671234567', 'AB12'),

('Shevchenko', 'Oleksandr', 'Volodymyrovych', '+380681112233', '34CD'),

('Hryhorenko', 'Lidiia', 'Yaroslavivna', '+380991234567', 'EF56'),

('Zakharchenko', 'Ostap', 'Viacheslavovich', '+380998611095', 'GH78'),

('Petrenko', 'Maria', 'Viktorivna', '+380633682611', 'IJ90'),

('Kozak', 'Anatolii', 'Oleksiyovych', '+380921112233', 'KL12'),

('Lysenko', 'Nataliia', 'Pavlivna', '+380927321976', 'MN34'),

('Sydorenko', 'Vadym', 'Ivanovych', '+380926947180', 'OP56'),

('Romanchuk', 'Volodymyr', 'Volodymyrovych', '+380611112233', 'QR78'),

('Hrynevych', 'Pavlo', 'Pavlovich', '+380941234567', 'ST90'),

('Ivanov', 'Ivan', 'Ivanovich', '+380561234567', 'ABC1'),

('Zaharchenko', 'Volodymyr', 'Mykolayovich', '+380661112233', 'EFGH'),

('Petrenko', 'Natalia', 'Vitaliyivna', '+380731234567', 'IJKL'),

('Romanchuk', 'Oksana', 'Dmytrivna', '+380961234567', 'MNOP'),

('Hrynevych', 'Valentyn', 'Yevhenovych', '+380663705997', 'QRST'),

('Kravchuk', 'Anna', 'Oleksandrivna', '+380441234567', 'UVWX'),

('Shevchenko', 'Taras', 'Hryhorovich', '+380501112233', 'YZ12'),

('Hryhorenko', 'Anna', 'Dmytrivna', '+380631234567', '3456'),

('Zakharchenko', 'Serhii', 'Vitaliiovych', '+380731112233', '7890'),

('Petrenko', 'Iryna', 'Mykolaivna', '+380951234567', 'ABCD');

-- Вставка осіб, що складають іспит --

INSERT INTO applicant (last\_name, first\_name, middle\_name, phone\_number, applicant\_id, amount\_of\_attempts)

VALUES ('Ivanov', 'Andriy', 'Petrovich', '+380686750607', '40F9ZS4KTI', 0),

('Petrova', 'Olena', 'Mykhailivna', '+380664298642', '1Q6LHSY9F6', 0),

('Kozlov', 'Serhiy', 'Oleksandrovich', '+380638543886', 'U5ZIIHI9Q4', 0),

('Shevchenko', 'Kateryna', 'Mykolaivna', '+380991606774', 'MKH7KZICVW', 0),

('Lysenko', 'Mykola', 'Dmytrovych', '+380663775943', 'ZY099QM7M6', 0),

('Tkachenko', 'Iryna', 'Yuriivna', '+380676298248', 'BGAVV8CV5Z', 0),

('Hrytsenko', 'Nataliia', 'Pavlivna', '+380990992107', '9TZ364BR9I', 0),

('Sydorenko', 'Vadym', 'Ivanovych', '+380682389433', '0887FFI23N', 0),

('Romanchuk', 'Oksana', 'Dmytrivna', '+380666507549', '9659RIK4BZ', 0),

('Hrynevych', 'Valentyn', 'Yevhenovych', '+380987674502', 'OE4BTKOAZ5', 0),

('Ivanova', 'Nataliia', 'Olehivna', '+380505968805', '4RURLOCH7Z', 0),

('Zaharchenko', 'Dmitro', 'Igorevich', '+380931586207', 'JBCFPPE6NC', 0),

('Pavlova', 'Anna', 'Yaroslavivna', '+380663334453', '5RMO8V698L', 0),

('Shevchenko', 'Dmytro', 'Oleksandrovych', '+380973439606', '5VE53ITSYC', 0),

('Hryhorova', 'Iryna', 'Yuriivna', '+380965770884', '0PT4XWXPJD', 0),

('Ivanchenko', 'Andriy', 'Olehovych', '+380680991414', 'QUC606TU5L', 0),

('Kovalenko', 'Viktor', 'Viktorovych', '+380665055007', 'P0S261EE9G', 0),

('Hrytsenko', 'Daria', 'Mykolaivna', '+380503712606', '20ZFS6U0D2', 0),

('Morozova', 'Olga', 'Ivanivna', '+380676217585', 'DPYQTXUXRP', 0),

('Sydorenko', 'Ivan', 'Pavlovych', '+380946004293', 'MHBABTA572', 0),

('Romanchuk', 'Yaroslav', 'Sergiyovych', '+380662401819', 'ZF8HBA2A8J', 0),

('Hrynevych', 'Nadiia', 'Valentynivna', '+380930401438', '9B4VQ2VTKQ', 0),

('Koval', 'Oleksii', 'Anatoliiovych', '+380680312557', '7ELHYTSV0U', 0),

('Ivanchenko', 'Inna', 'Vitaliivna', '+380948701414', 'UOGJRBG8I6', 0),

('Zakharov', 'Pavlo', 'Dmytrovych', '+380638386733', '41Y1IH4TGS', 0),

('Romanchuk', 'Katerina', 'Olehivna', '+380686620221', '9IK0TGZ0IA', 0),

('Hryhorenko', 'Viktor', 'Ivanovych', '+380632274751', 'DB327YK0SV', 0),

('Ivanova', 'Olga', 'Igorevna', '+380926458499', '6RK3SIJO49', 0),

('Zakharova', 'Svitlana', 'Yaroslavivna', '+380926717598', 'KYHJVSW3Y3', 0),

('Pavlov', 'Andriy', 'Oleksandrovych', '+380673122806', 'HT5N5A6V2F', 0),

('Hryhorov', 'Yevhen', 'Petrovych', '+380966171352', 'QEHCKGK33E', 0),

('Ivanchuk', 'Kateryna', 'Mykolaivna', '+380934479359', 'WMOSRLJY9H', 0),

('Kozachenko', 'Petro', 'Olehovych', '+380966625180', '2REUWXLKQF', 0),

('Shevchenko', 'Olena', 'Dmytrivna', '+380504230079', 'FDYHW0XP2J', 0),

('Lysenko', 'Vasyl', 'Ivanovych', '+380675163223', '6T01IFF9BH', 0),

('Tkachenko', 'Oksana', 'Dmytrivna', '+380680797254', 'WLD18S5XIE', 0),

('Hrynevych', 'Volodymyr', 'Yevhenovych', '+380991112233', 'R7CPUZUH6M', 0),

('Ivanova', 'Mariia', 'Olehivna', '+380918915336', 'Q1A04JFGAH', 0),

('Borisenko', 'Yulia', 'Sergiyivna', '+380671342178', 'F7925XX1DM', 0),

('Zaharchenko', 'Igor', 'Igorevich', '+380561112233', '667MQLT2A9', 0),

('Pavliuk', 'Nina', 'Yanivna', '+380007263735', '6F01L4Q57H', 0),

('Miroshnichenko', 'Anastasiia', 'Tarasivna', '+380455054926', 'QGV3S10IS9', 0),

('Sereda', 'Dmytro', 'Anatoliyovych', '+380309204129', 'VXQ9CD0AE1', 0),

('Bondarenko', 'Olena', 'Volodymyrivna', '+380490872348', 'E8Z9TS5O2D', 0),

('Tkachenko', 'Anna', 'Sergiyivna', '+380303495352', '355XJYW70R', 0),

('Tarashchuk', 'Marina', 'Yevheniyivna', '+380283001075', 'GBW0GMVUD6', 0),

('Boiko', 'Dmytro', 'Pavlovych', '+380352449110', 'U3EU491OCT', 0),

('Kovalenko', 'Kateryna', 'Yosypivna', '+380923931026', 'JA05CA2JJZ', 0),

('Ivachenko', 'Denis', 'Mikolayovich', '+380133451501', '1JLNREGYNG', 0),

('Kravchenko', 'Lubov', 'Fedorivna', '+380771527702', 'WVGKTBHGR8', 0);

-- Вставка медичних сертифікатів --

INSERT INTO medical\_certificates (certificate\_number, group\_of\_blood, rhesus\_factor, person\_id, ability\_to\_drive\_marker)

VALUES ('570880', 1, '+', '40F9ZS4KTI', true),

('771723', 2, '-', '1Q6LHSY9F6', true),

('210505', 3, '-', 'U5ZIIHI9Q4', true),

('892669', 4, '-', 'MKH7KZICVW', false),

('504467', 1, '+', 'ZY099QM7M6', true),

('744499', 2, '-', 'BGAVV8CV5Z', true),

('219834', 3, '+', '9TZ364BR9I', true),

('748130', 4, '-', '0887FFI23N', true),

('115071', 1, '+', '9659RIK4BZ', true),

('744799', 2, '-', 'OE4BTKOAZ5', false),

('636463', 3, '-', '4RURLOCH7Z', true),

('243896', 4, '-', 'JBCFPPE6NC', false),

('880491', 1, '+', '5RMO8V698L', true),

('175158', 2, '-', '5VE53ITSYC', true),

('891544', 3, '+', '0PT4XWXPJD', false),

('504991', 4, '-', 'QUC606TU5L', false),

('566679', 1, '+', 'P0S261EE9G', true),

('758083', 2, '+', '20ZFS6U0D2', true),

('771874', 3, '+', '41Y1IH4TGS', true),

('352717', 4, '-', 'DPYQTXUXRP', true),

('819665', 1, '+', 'MHBABTA572', true),

('771873', 2, '-', 'ZF8HBA2A8J', true),

('722046', 3, '+', '9B4VQ2VTKQ', false),

('115076', 4, '-', '7ELHYTSV0U', true),

('570871', 1, '+', 'UOGJRBG8I6', true),

('892658', 2, '+', 'F7925XX1DM', true),

('839245', 3, '+', '9IK0TGZ0IA', true),

('135157', 4, '-', 'DB327YK0SV', true),

('744794', 1, '+', '6RK3SIJO49', true),

('596700', 2, '-', 'KYHJVSW3Y3', true),

('522122', 3, '+', 'HT5N5A6V2F', true),

('219837', 4, '-', 'QEHCKGK33E', true),

('749499', 1, '+', 'WMOSRLJY9H', true),

('539733', 2, '-', '2REUWXLKQF', false),

('723740', 3, '-', 'FDYHW0XP2J', true),

('570869', 4, '-', '6T01IFF9BH', true),

('748137', 1, '+', 'WLD18S5XIE', true),

('892657', 2, '-', 'R7CPUZUH6M', true),

('175157', 3, '+', 'Q1A04JFGAH', true),

('993569', 3, '+', '6F01L4Q57H', true),

('293570', 2, '+', 'QGV3S10IS9', true),

('977285', 1, '+', 'VXQ9CD0AE1', true),

('246920', 1, '+', 'E8Z9TS5O2D', true),

('699213', 3, '-', '355XJYW70R', true),

('384401', 3, '+', 'GBW0GMVUD6', true),

('121750', 2, '-', 'U3EU491OCT', true),

('866647', 2, '-', 'JA05CA2JJZ', true),

('845370', 1, '+', '1JLNREGYNG', true),

('711323', 4, '-', 'WVGKTBHGR8', true),

('504997', 4, '-', '667MQLT2A9', false);

-- Вставка іспитів з теорії --

INSERT INTO log\_of\_theoretical\_exams (date\_of\_passing, time\_of\_passing, applicant\_id, inspector\_id, success\_marker, exam\_grade)

VALUES ('2023-07-14', '11:45', '4RURLOCH7Z', 'KL12', FALSE, 16),

('2023-07-14', '12:25', '0887FFI23N', 'KL12', FALSE, 13),

('2023-07-14', '14:00', 'DPYQTXUXRP', 'KL12', TRUE, 19),

('2023-07-14', '15:25', '7ELHYTSV0U', 'KL12', FALSE, 13),

('2023-07-21', '14:00', '9IK0TGZ0IA', 'J7K1', TRUE, 19),

('2023-07-21', '15:15', '6F01L4Q57H', 'J7K1', FALSE, 17),

('2023-07-21', '16:00', 'QGV3S10IS9', 'J7K1', TRUE, 19),

('2023-07-21', '17:00', 'R7CPUZUH6M', 'J7K1', TRUE, 18),

('2023-08-01', '12:00', 'ZY099QM7M6', 'OP56', TRUE, 18),

('2023-08-01', '12:15', 'MHBABTA572', 'OP56', FALSE, 17),

('2023-08-01', '12:20', '6RK3SIJO49', 'OP56', TRUE, 20),

('2023-08-08', '12:00', 'VXQ9CD0AE1', 'LK91', FALSE, 10),

('2023-08-08', '12:25', 'Q1A04JFGAH', 'LK91', FALSE, 14),

('2023-08-09', '12:45', 'KYHJVSW3Y3', '3V92', TRUE, 19),

('2023-08-09', '13:05', 'QEHCKGK33E', '3V92', TRUE, 20),

('2023-08-09', '13:15', 'U5ZIIHI9Q4', 'LK91', TRUE, 20),

('2023-08-10', '13:20', 'BGAVV8CV5Z', 'LK91', FALSE, 12),

('2023-08-10', '13:20', '40F9ZS4KTI', 'LK91', TRUE, 19),

('2023-08-11', '13:20', 'FDYHW0XP2J', 'LK91', FALSE, 15),

('2023-08-11', '14:00', '1Q6LHSY9F6', '9Z01', TRUE, 18),

('2023-08-15', '13:00', '9TZ364BR9I', '4567', TRUE, 19),

('2023-08-15', '11:15', '4RURLOCH7Z', 'ST90', TRUE, 19),

('2023-08-15', '12:00', '7ELHYTSV0U', 'ST90', TRUE, 20),

('2023-08-15', '12:35', 'HT5N5A6V2F', 'ST90', TRUE, 18),

('2023-08-15', '15:00', '6F01L4Q57H', 'ST90', TRUE, 19),

('2023-08-15', '16:00', 'VXQ9CD0AE1', 'ST90', TRUE, 18),

('2023-08-16', '10:00', 'E8Z9TS5O2D', 'QR78', TRUE, 20),

('2023-08-16', '11:00', '355XJYW70R', 'QR78', TRUE, 18),

('2023-08-16', '11:45', 'GBW0GMVUD6', 'QR78', TRUE, 18),

('2023-08-16', '12:00', 'U3EU491OCT', 'QR78', TRUE, 18),

('2023-09-12', '15:15', '5RMO8V698L', 'P5R2', TRUE, 20),

('2023-09-12', '16:00', 'JA05CA2JJZ', 'P5R2', TRUE, 20),

('2023-09-12', '16:15', '1JLNREGYNG', 'P5R2', TRUE, 19),

('2023-09-12', '16:45', 'WVGKTBHGR8', 'P5R2', TRUE, 20),

('2023-10-15', '12:20', '5VE53ITSYC', 'AB12', TRUE, 19),

('2023-10-19', '09:00', 'P0S261EE9G', '34CD', TRUE, 18),

('2023-10-18', '14:20', 'WLD18S5XIE', 'GH78', TRUE, 20),

('2023-10-18', '14:35', 'BGAVV8CV5Z', 'GH78', TRUE, 19),

('2023-10-27', '12:20', '6T01IFF9BH', '9Z01', TRUE, 20),

('2023-11-12', '15:00', '0887FFI23N', 'KL12', TRUE, 18),

('2023-11-12', '15:00', 'DB327YK0SV', 'KL12', TRUE, 19),

('2023-11-12', '15:10', '9659RIK4BZ', 'KL12', TRUE, 20),

('2023-11-12', '15:15', '20ZFS6U0D2', 'KL12', TRUE, 19),

('2023-11-12', '15:15', 'FDYHW0XP2J', 'KL12', TRUE, 18),

('2023-12-08', '12:00', 'MHBABTA572', '8P3Y', TRUE, 20),

('2023-12-08', '12:00', 'WMOSRLJY9H', '8P3Y', TRUE, 20),

('2024-01-06', '14:15', 'UOGJRBG8I6', 'EF56', NULL, NULL),

('2024-01-06', '14:20', 'ZF8HBA2A8J', 'EF56', NULL, NULL),

('2024-01-06', '14:30', '41Y1IH4TGS', 'EF56', NULL, NULL),

('2024-01-06', '14:40', 'Q1A04JFGAH', 'EF56', NULL, NULL);

-- Вставка іспитів з практики --

INSERT INTO log\_of\_practical\_exams (date\_of\_passing, time\_of\_passing, transport\_id, inspector\_id, applicant\_id, success\_marker, amount\_of\_mistakes, category\_of\_transport)

VALUES ('2023-10-11', '09:00', 'I9J0', 'KL12', '4RURLOCH7Z', TRUE, 2, 'B'),

('2023-10-11', '11:00', 'Q7R8', '34CD', '7ELHYTSV0U', TRUE, 0, 'C1'),

('2023-10-12', '12:00', 'Q7R8', 'QR78', '9IK0TGZ0IA', FALSE, 4, 'C1'),

('2023-10-12', '13:00', 'E5F6', 'OP56', 'HT5N5A6V2F', TRUE, 1, 'A'),

('2023-10-12', '13:15', 'I9J0', 'IJ90', 'R7CPUZUH6M', FALSE, 4, 'B'),

('2023-10-14', '14:00', 'U1V2', 'YZ12', '6F01L4Q57H', TRUE, 1, 'B1'),

('2023-10-14', '15:00', 'MNOP', '34CD', 'QGV3S10IS9', FALSE, 4, 'B'),

('2023-10-14', '15:30', 'I9J0', 'QR78', 'VXQ9CD0AE1', FALSE, 8, 'B'),

('2023-10-14', '15:45', 'BC45', 'EFGH', 'E8Z9TS5O2D', TRUE, 0, 'B'),

('2023-10-16', '11:05', 'BC45', 'GH78', '355XJYW70R', TRUE, 0, 'B'),

('2023-10-16', '12:15', 'A1B2', 'AB12', 'ZY099QM7M6', FALSE, 4, 'A'),

('2023-10-16', '13:00', 'W3X4', '3456', 'DPYQTXUXRP', TRUE, 2, 'A'),

('2023-10-23', '12:00', 'GH12', '7890', 'HT5N5A6V2F', TRUE, 0, 'A1'),

('2023-10-23', '12:45', 'D678', '3456', '6F01L4Q57H', TRUE, 0, 'D1'),

('2023-10-23', '14:50', 'G7H8', 'OP56', 'QGV3S10IS9', FALSE, 3, 'B'),

('2023-10-23', '16:00', 'BC45', 'YZ12', 'GBW0GMVUD6', TRUE, 2, 'B'),

('2023-10-23', '17:00', 'I9J0', 'KL12', 'R7CPUZUH6M', TRUE, 0, 'B'),

('2023-10-25', '09:00', 'G7H8', 'EF56', '1Q6LHSY9F6', FALSE, 5, 'B'),

('2023-10-25', '11:00', 'W3X4', '34CD', '9TZ364BR9I', TRUE, 1, 'B'),

('2023-10-25', '11:34', 'I9J0', 'MN34', '5RMO8V698L', TRUE, 2, 'B'),

('2023-10-25', '12:00', 'U1V2', 'GH78', '5VE53ITSYC', FALSE, 7, 'B1'),

('2023-10-26', '09:00', 'U1V2', 'QR78', 'DPYQTXUXRP', TRUE, 1, 'B1'),

('2023-10-26', '10:00', 'BC45', 'MNOP', '7ELHYTSV0U', TRUE, 1, 'B'),

('2023-10-26', '11:00', 'Q7R8', '34CD', '9IK0TGZ0IA', FALSE, 5, 'C1'),

('2023-10-26', '12:00', 'I9J0', 'IJ90', 'HT5N5A6V2F', TRUE, 2, 'B'),

('2023-10-26', '13:00', 'G7H8', 'OP56', 'QGV3S10IS9', TRUE, 1, 'B'),

('2023-10-29', '08:00', 'S9T0', 'ST90', 'P0S261EE9G', FALSE, 4, 'C'),

('2023-10-29', '08:45', 'O5P6', 'EFGH', '1Q6LHSY9F6', TRUE, 1, 'A'),

('2023-10-29', '09:00', 'Y5Z6', 'MNOP', 'WLD18S5XIE', FALSE, 5, 'A'),

('2023-10-29', '11:00', '123A', 'QRST', 'BGAVV8CV5Z', FALSE, 8, 'D'),

('2023-11-02', '15:00', 'BC45', '34CD', 'VXQ9CD0AE1', FALSE, 4, 'B'),

('2023-11-02', '15:55', 'G7H8', 'QR78', 'U3EU491OCT', FALSE, 3, 'B'),

('2023-11-02', '16:00', 'U1V2', '3456', 'HT5N5A6V2F', TRUE, 0, 'B1'),

('2023-11-02', '16:30', 'Q7R8', 'KL12', '9IK0TGZ0IA', TRUE, 1, 'C1'),

('2023-11-05', '12:45', 'EF90', 'MN34', 'ZY099QM7M6', FALSE, 7, 'T'),

('2023-11-05', '13:00', 'IJKL', '34CD', '6T01IFF9BH', TRUE, 0, 'T'),

('2023-11-05', '14:00', 'S9T0', 'YZ12', '0887FFI23N', TRUE, 1, 'C'),

('2023-11-25', '12:00', 'IJKL', 'UVWX', 'WLD18S5XIE', FALSE, 3, 'T'),

('2023-11-25', '12:00', 'I9J0', 'EF56', 'DB327YK0SV', FALSE, 4, 'B'),

('2023-11-25', '12:00', 'C3D4', 'ABC1', 'P0S261EE9G', TRUE, 2, 'A1'),

('2023-11-25', '15:50', 'O5P6', 'IJKL', '9659RIK4BZ', FALSE, 3, 'A'),

('2023-12-30', '12:00', 'BC45', 'ST90', '20ZFS6U0D2', FALSE, 3, 'B'),

('2023-12-30', '14:00', 'W3X4', '3456', 'FDYHW0XP2J', TRUE, 0, 'B'),

('2023-12-30', '15:00', 'MNOP', 'ABCD', 'MHBABTA572', TRUE, 1, 'B'),

('2023-12-30', '16:00', 'BC45', 'IJKL', 'WMOSRLJY9H', TRUE, 0, 'B'),

('2024-01-01', '15:15', 'I9J0', 'IJ90', 'VXQ9CD0AE1', FALSE, 7, 'B'),

('2024-01-01', '16:55', 'BC45', 'MNOP', 'U3EU491OCT', TRUE, 1, 'B'),

('2024-01-02', '12:00', 'C3D4', 'KL12', 'JA05CA2JJZ', TRUE, 0, 'A1'),

('2024-01-02', '12:45', 'G7H8', 'OP56', '1JLNREGYNG', TRUE, 0, 'B'),

('2024-01-02', '13:00', 'Q7R8', '3456', 'WVGKTBHGR8', TRUE, 1, 'C1'),

('2024-01-03', '11:00', 'W3X4', 'KL12', '1Q6LHSY9F6', TRUE, 1, 'B'),

('2024-01-03', '12:00', 'E5F6', 'ST90', 'KYHJVSW3Y3', TRUE, 1, 'A'),

('2024-01-03', '15:55', 'Y5Z6', 'IJ90', 'ZY099QM7M6', NULL, NULL, 'A'),

('2024-01-03', '15:55', 'U1V2', '7890', '9659RIK4BZ', NULL, NULL, 'B1'),

('2024-01-03', '16:15', 'W3X4', 'IJKL', '5VE53ITSYC', NULL, NULL, 'B'),

('2024-01-04', '08:00', 'GH12', 'AB12', '6RK3SIJO49', NULL, NULL, 'A1'),

('2024-01-04', '09:00', 'K1L2', 'YZ12', 'QEHCKGK33E', NULL, NULL, 'A1'),

('2024-01-04', '10:00', 'D678', 'KL12', '20ZFS6U0D2', NULL, NULL, 'D1'),

('2024-01-04', '12:25', 'M3N4', '7890', 'DB327YK0SV', NULL, NULL, 'C'),

('2024-01-05', '14:30', 'G7H8', 'UVWX', 'WLD18S5XIE', NULL, NULL, 'B'),

('2024-01-05', '15:00', 'I9J0', 'QR78', 'P0S261EE9G', NULL, NULL, 'B'),

('2024-01-06', '13:00', 'W3X4', 'OP56', '9659RIK4BZ', NULL, NULL, 'B'),

('2024-01-06', '14:00', 'G7H8', 'AB12', 'BGAVV8CV5Z', NULL, NULL, 'B'),

('2024-01-06', '15:00', 'A1B2', 'KL12', 'U5ZIIHI9Q4', NULL, NULL, 'A'),

('2024-01-06', '16:00', 'Q7R8', '34CD', '40F9ZS4KTI', NULL, NULL, 'C1');

-- Вставка водійських прав --

INSERT INTO drivers\_license (drivers\_license\_number, last\_name, first\_name, middle\_name, obtaining\_date, expiration\_date)

VALUES ('DEO607623', 'Hrytsenko', 'Nataliia', 'Pavlivna', '2023-10-12', '2053-10-12'),

('GHC629684', 'Pavlova', 'Anna', 'Yaroslavivna', '2023-10-15', '2033-10-15'),

('IEC746207', 'Petrova', 'Olena', 'Mykhailivna', '2023-10-16', '2025-10-16'),

('KOX781795', 'Lysenko', 'Vasyl', 'Ivanovych', '2023-11-05', '2053-11-05'),

('OCV821326', 'Sydorenko', 'Vadym', 'Ivanovych', '2023-11-05', '2053-11-05'),

('TDD337514', 'Kovalenko', 'Viktor', 'Viktorovych', '2023-11-25', '2053-11-25'),

('VAN404609', 'Shevchenko', 'Olena', 'Dmytrivna', '2023-12-30', '2053-12-30'),

('WPT356695', 'Sydorenko', 'Ivan', 'Pavlovych', '2023-12-30', '2053-12-30'),

('WYT949478', 'Ivanchuk', 'Kateryna', 'Mykolaivna', '2024-01-03', '2026-01-03'),

('YBT046268', 'Zakharova', 'Svitlana', 'Yaroslavivna', '2024-01-03', '2054-01-03'),

('MLQ650656', 'Ivanova', 'Nataliia', 'Olehivna', '2023-07-14', '2053-07-14'),

('YUD505187', 'Morozova', 'Olga', 'Ivanivna', '2023-10-16', '2025-10-16'),

('YPY243387', 'Koval', 'Oleksii', 'Anatoliiovych', '2023-10-11', '2053-10-11'),

('GGQ027807', 'Romanchuk', 'Katerina', 'Olehivna', '2023-11-02', '2053-11-02'),

('SPE103618', 'Pavlov', 'Andriy', 'Oleksandrovych', '2023-10-12', '2033-10-12'),

('CSY833559', 'Hrynevych', 'Volodymyr', 'Yevhenovych', '2023-10-23', '2025-10-23'),

('WZT497204', 'Pavliuk', 'Nina', 'Yanivna', '2023-10-23', '2053-10-23'),

('UQY043227', 'Miroshnichenko', 'Anastasiia', 'Tarasivna', '2023-10-26', '2053-10-26'),

('JLT842865', 'Bondarenko', 'Olena', 'Volodymyrivna', '2023-10-14', '2053-10-14'),

('UBL640882', 'Tkachenko', 'Anna', 'Sergiyivna', '2023-10-16', '2053-10-16'),

('QIS616181', 'Tarashchuk', 'Marina', 'Yevheniyivna', '2023-10-23', '2053-10-23'),

('KRK967838', 'Ivachenko', 'Denis', 'Mikolayovich', '2024-01-02', '2054-01-02'),

('CAU185664', 'Kravchenko', 'Lubov', 'Fedorivna', '2024-01-02', '2026-01-02'),

('VMV596859', 'Boiko', 'Dmytro', 'Pavlovych', '2024-01-01', '2034-01-01'),

('HRE662993', 'Kovalenko', 'Kateryna', 'Yosypivna', '2024-01-02', '2054-01-02');

-- Вставка отриманих категорій --

INSERT INTO received\_categories (category\_of\_transport, drivers\_license\_number, type\_of\_gearbox)

VALUES ('B', 'DEO607623', 'AT'),

('B', 'MLQ650656', 'AT'),

('B', 'UBL640882', 'MT'),

('B', 'GHC629684', 'AT'),

('A', 'IEC746207', 'MT'),

('T', 'KOX781795', 'AT'),

('C', 'OCV821326', 'MT'),

('A1', 'TDD337514', 'MT'),

('B', 'VAN404609', 'AT'),

('B', 'WPT356695', 'MT'),

('B', 'WYT949478', 'MT'),

('A', 'YBT046268', 'MT'),

('B', 'IEC746207', 'AT'),

('A', 'YUD505187', 'AT'),

('B1', 'YUD505187', 'MT'),

('C1', 'YPY243387', 'MT'),

('B', 'YPY243387', 'MT'),

('C1', 'GGQ027807', 'MT'),

('A1', 'SPE103618', 'MT'),

('A', 'SPE103618', 'MT'),

('B1', 'SPE103618', 'MT'),

('B', 'SPE103618', 'AT'),

('B', 'CSY833559', 'AT'),

('B1', 'WZT497204', 'MT'),

('D1', 'WZT497204', 'MT'),

('B', 'UQY043227', 'MT'),

('B', 'JLT842865', 'MT'),

('B', 'QIS616181', 'MT'),

('B', 'KRK967838', 'MT'),

('C1', 'CAU185664', 'MT'),

('B', 'VMV596859', 'MT'),

('A1', 'HRE662993', 'MT');

# ДОДАТОК Б КОД ВСТАВКИ ДАНИХ ЗА ДОПОМОГОЮ КОМАНД ТИПУ COPY FROM

Заповнення бази даних можна організувати за допомогою імпортування з файлів. Нижче наведені команди для заповнення тестовими даними бази даних service\_center\_mia за допомогою імпортування з файлів, що розташовані на фізичному носії за адресою C:/files/.

COPY type\_of\_transport FROM 'C:/files/type\_of\_transport.csv' WITH CSV HEADER;

COPY category\_of\_transport FROM 'C:/files/category\_of\_transport.csv' WITH CSV HEADER;

COPY transport FROM 'C:/files/transport.csv' WITH CSV HEADER;

COPY practical\_exam\_inspector FROM 'C:/files/practical\_exam\_inspector.csv' WITH CSV HEADER;

COPY theoretical\_exam\_inspector FROM 'C:/files/theoretical\_exam\_inspector.csv' WITH CSV HEADER;

COPY applicant FROM 'C:/files/applicant.csv' WITH CSV HEADER;

COPY medical\_certificates FROM 'C:/files/medical\_certificates.csv' WITH CSV HEADER;

COPY log\_of\_theoretical\_exams (date\_of\_passing, time\_of\_passing, applicant\_id, inspector\_id, success\_marker, exam\_grade)

FROM 'C:/files/log\_of\_theoretical\_exams.csv' WITH CSV HEADER;

COPY log\_of\_practical\_exams (date\_of\_passing, time\_of\_passing, transport\_id, inspector\_id, applicant\_id, success\_marker, amount\_of\_mistakes, category\_of\_transport)

FROM 'C:/files/log\_of\_practical\_exams.csv' WITH CSV HEADER;

COPY drivers\_license FROM 'C:/files/drivers\_license.csv' WITH CSV HEADER;

COPY received\_categories FROM 'C:/files/received\_categories.csv' WITH CSV HEADER;