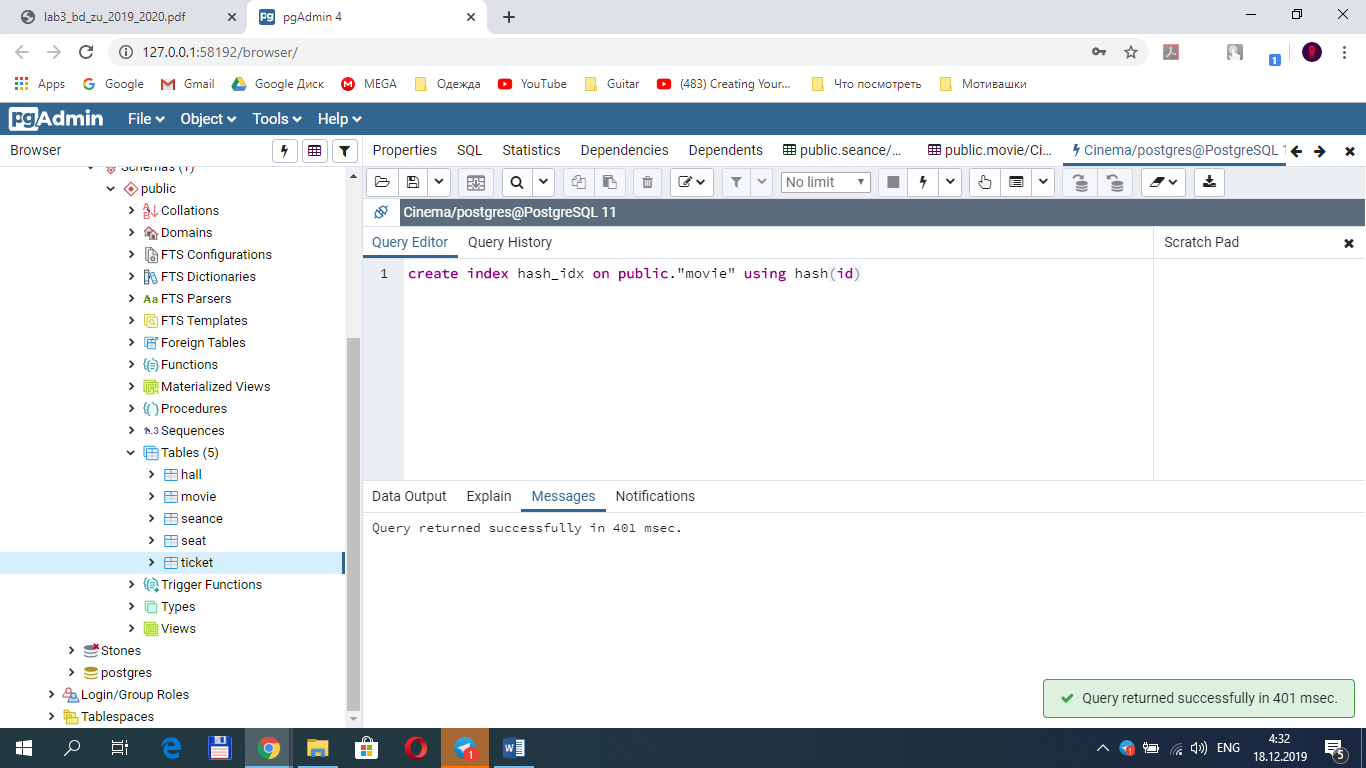
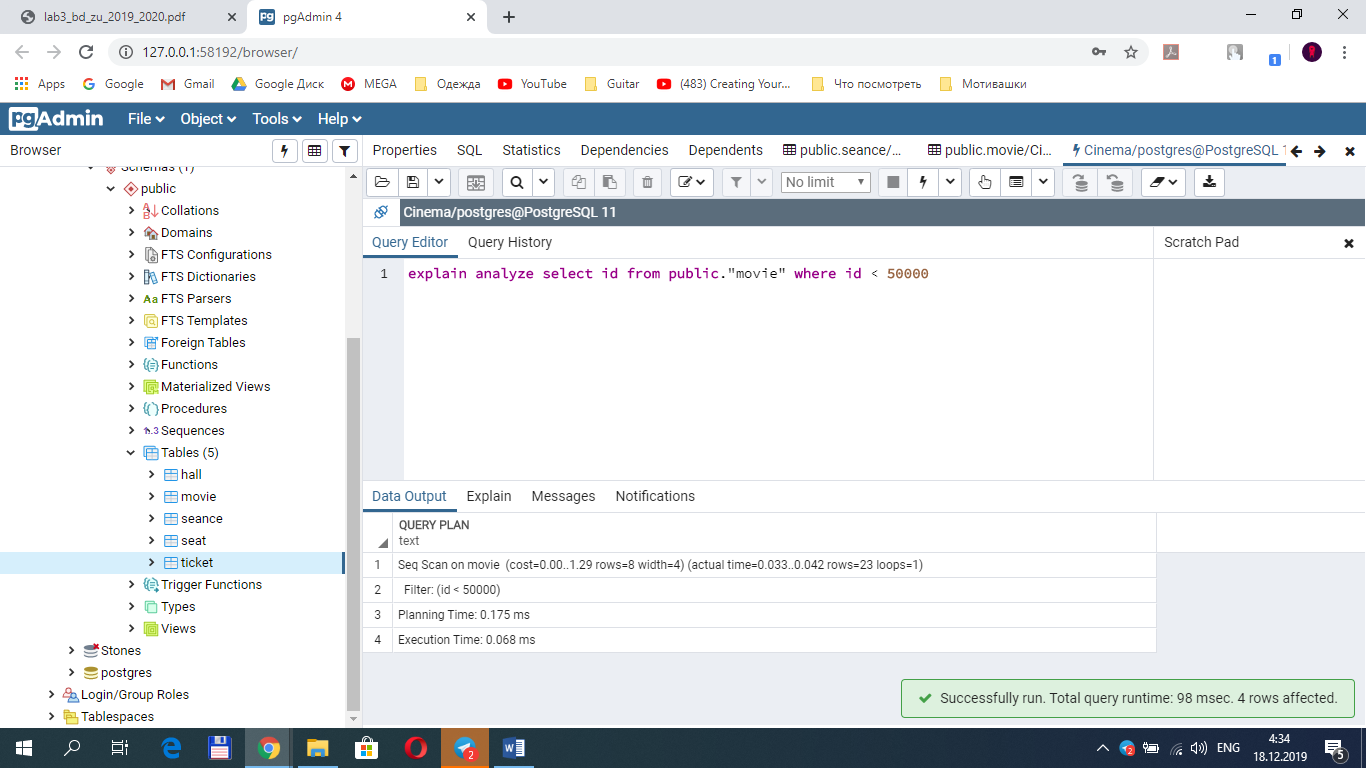
**Завдання 2:**

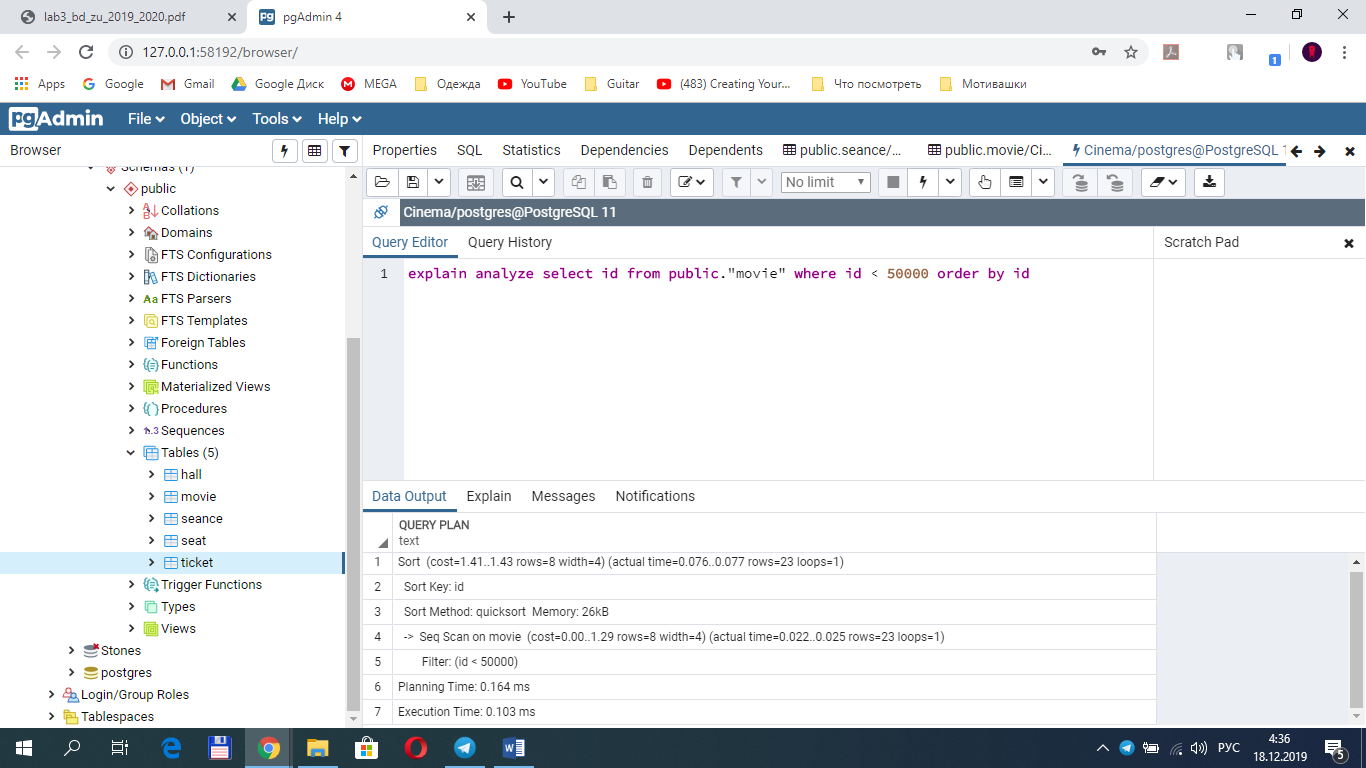
1. Створимо Hash-індекс



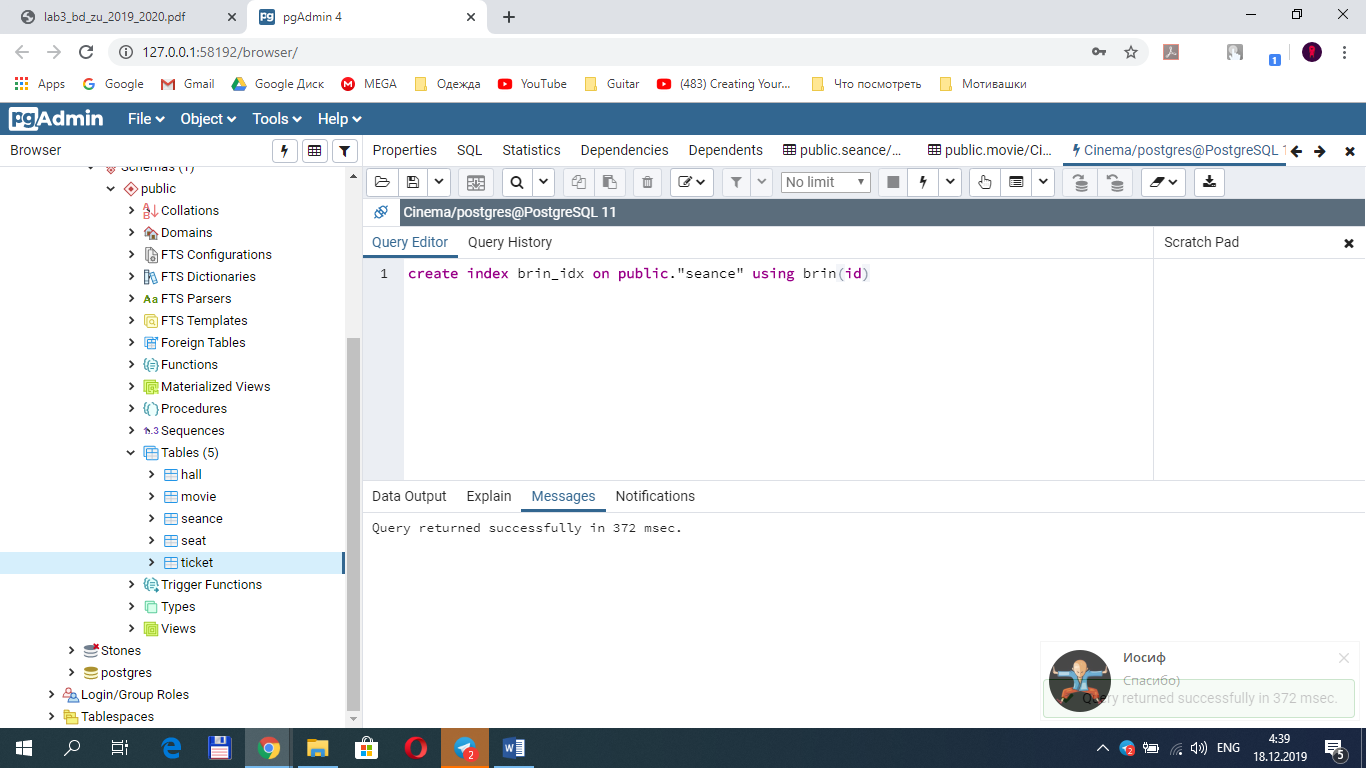
Запит з фільтрацією



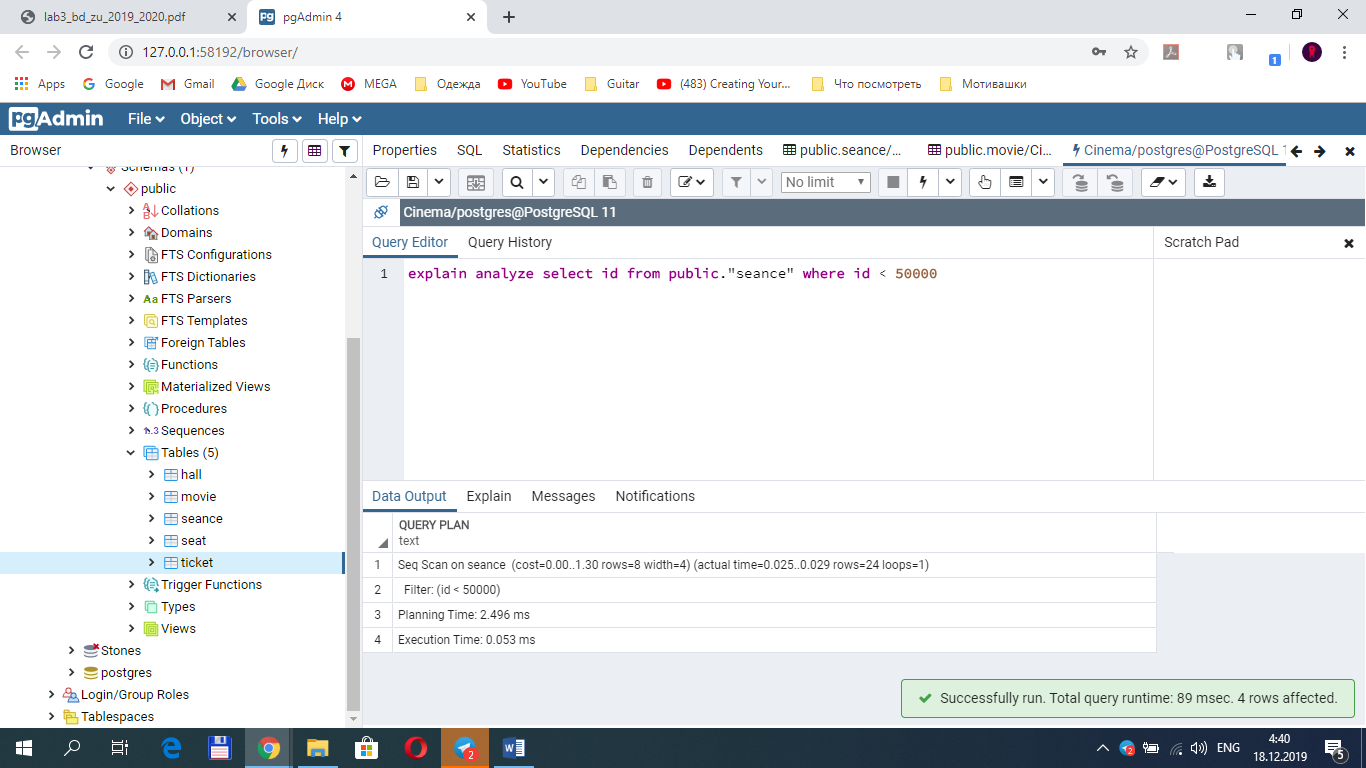
Запит з фільтрацією та сортуванням



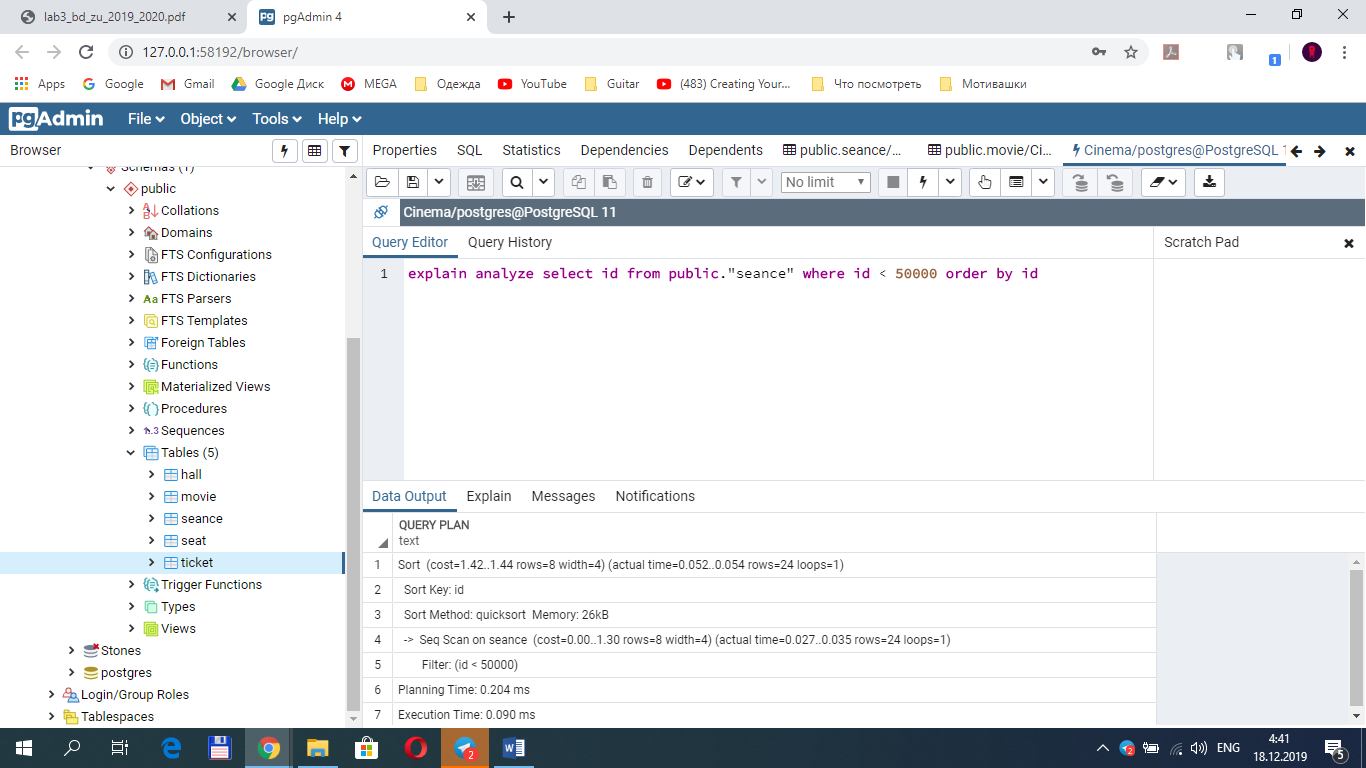
1. Створимо BRIN-індекс



Запит з фільтрацією



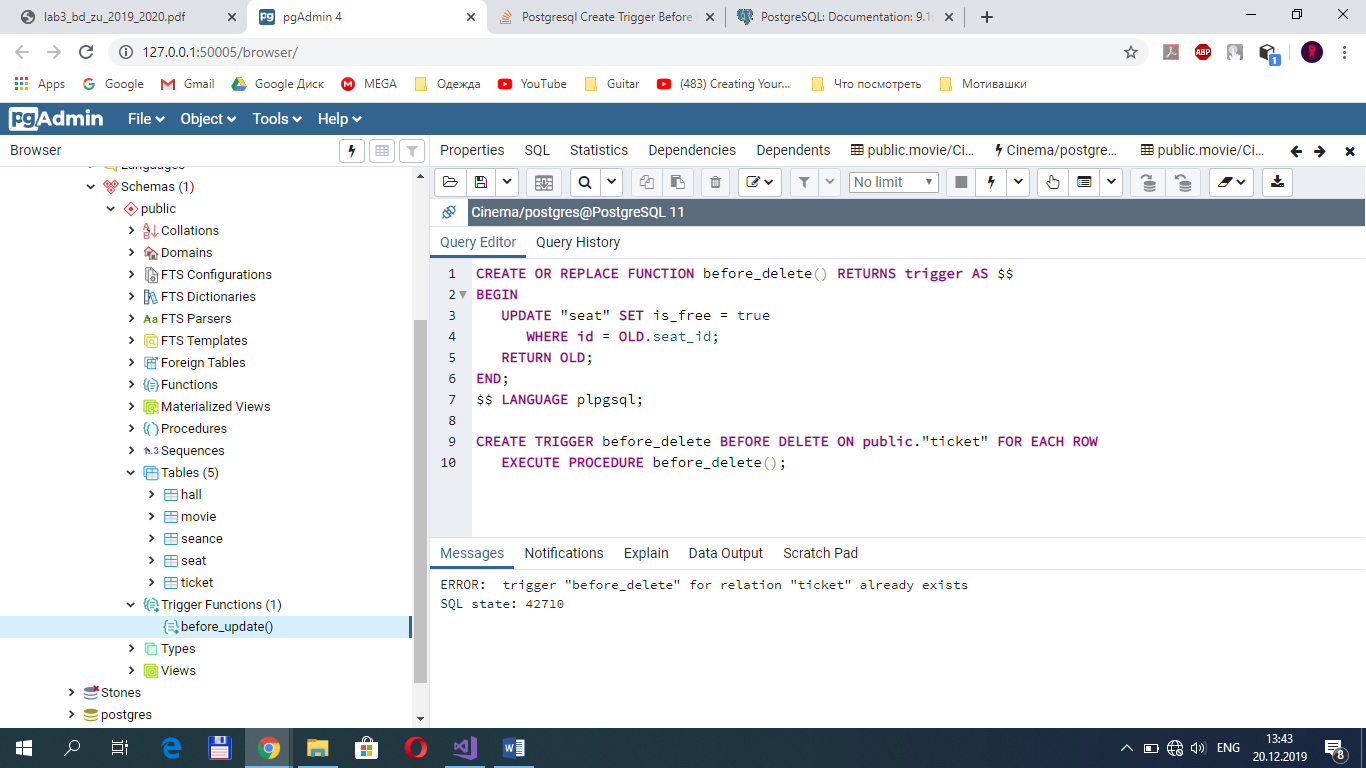
Запит з фільтрацією та сортуванням



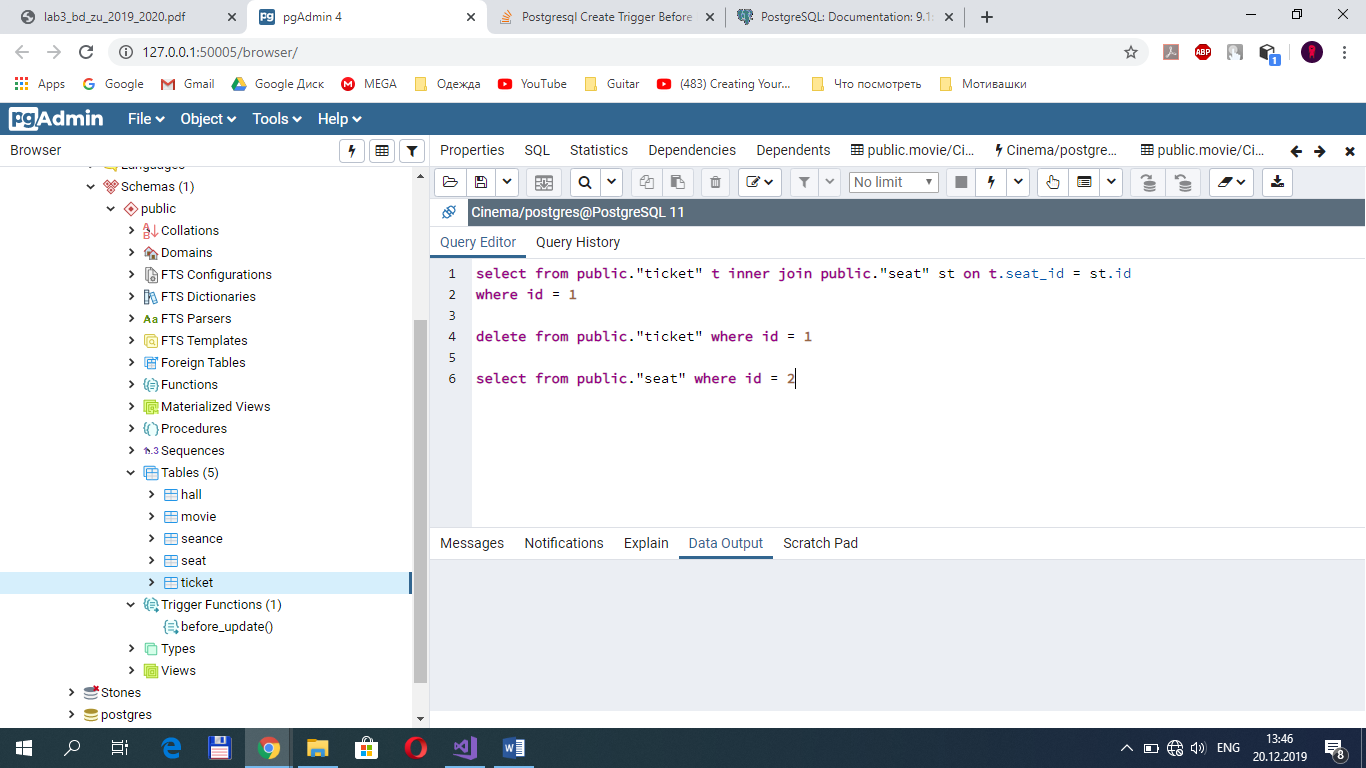
Отже, для того, щоб індекс зпрацював, має бути досить маленька відносно кількості записів вибірка. По-друге, зі скріншотів видно, що BRIN-індекс працює швидше Hash-індексу.

**Завдання 3:**

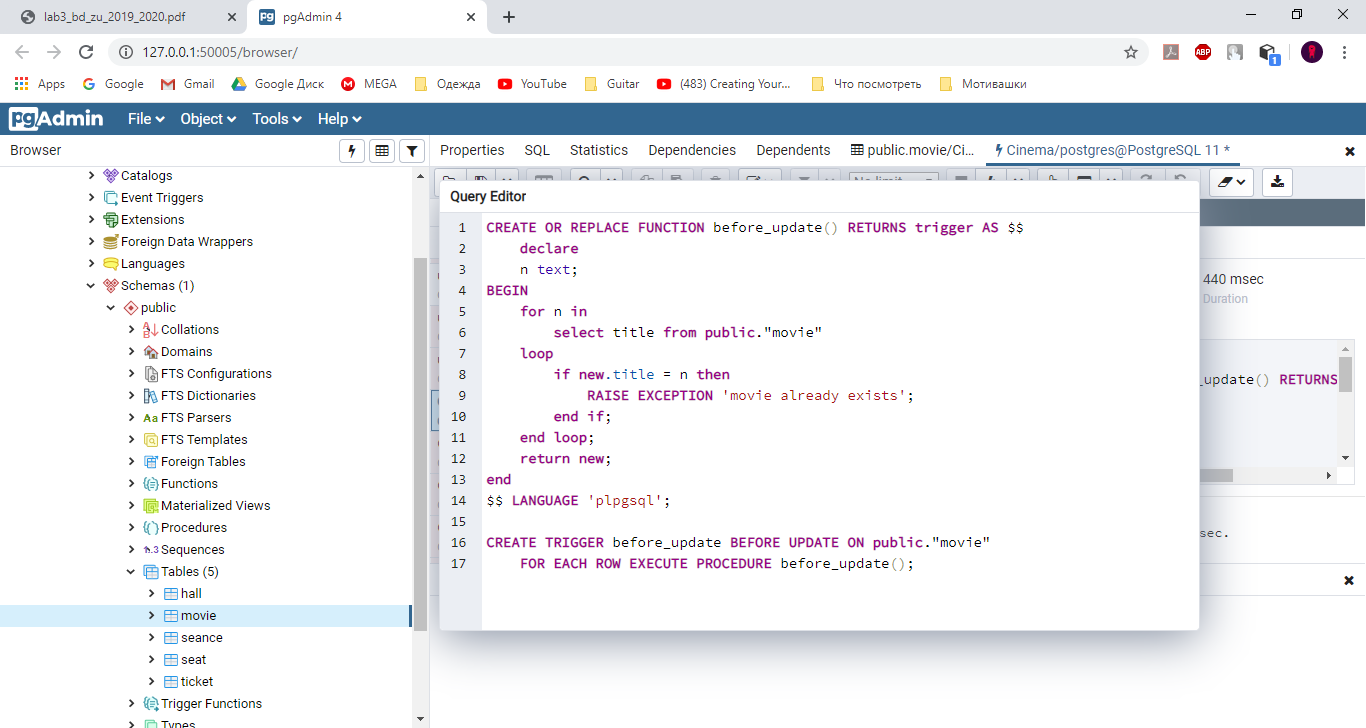
1. Before delete



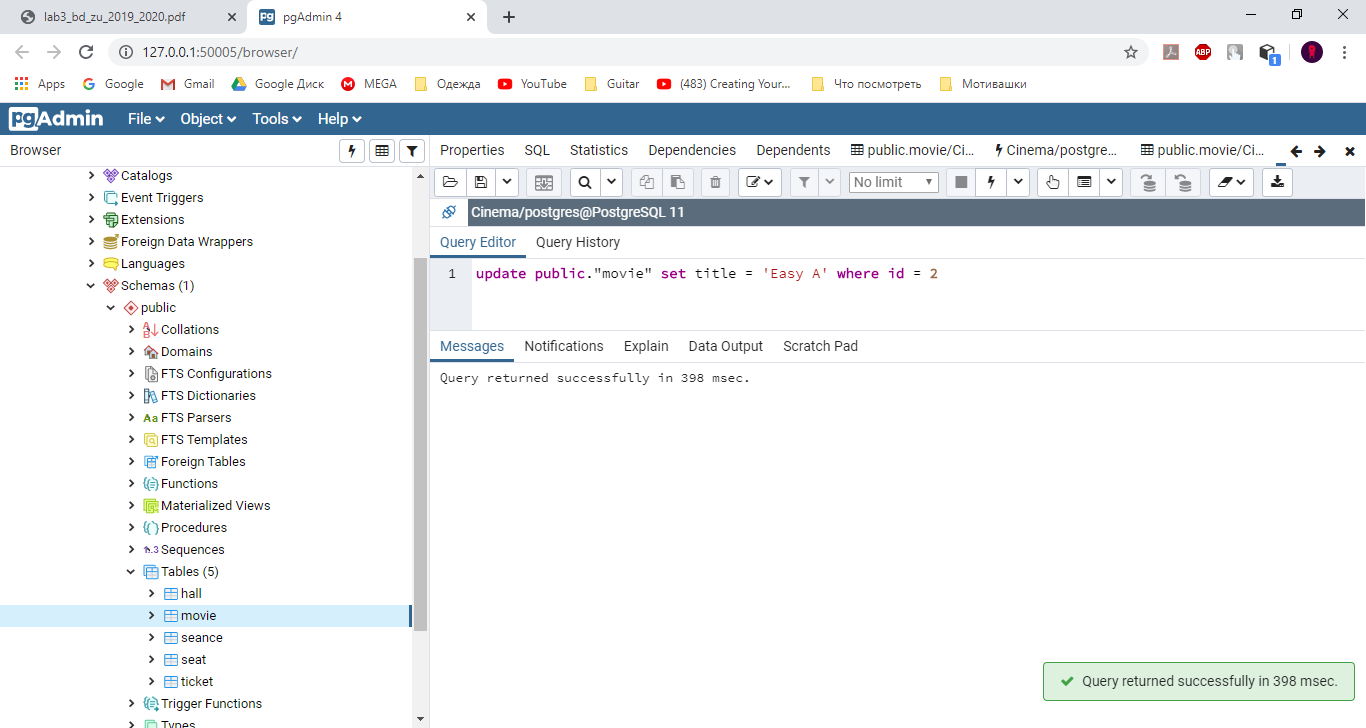
Операція



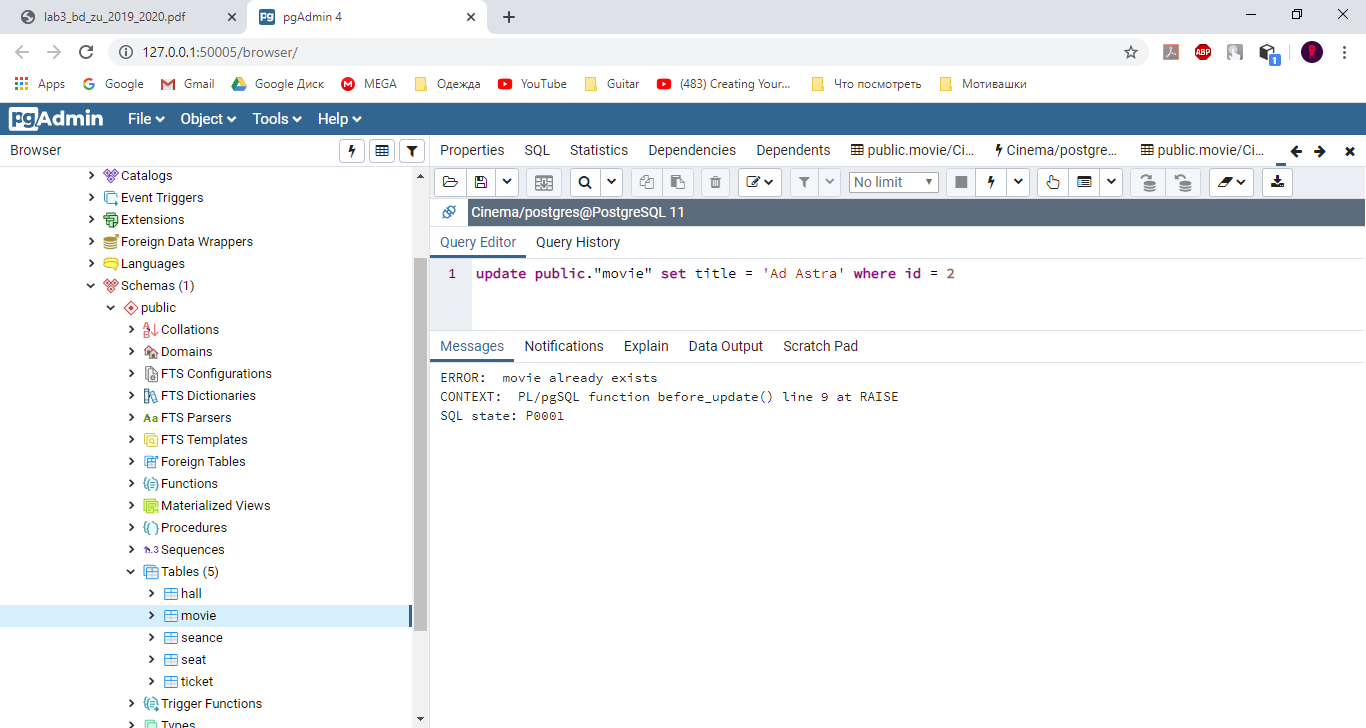
Before update



Коректна операція



Некоректна операція

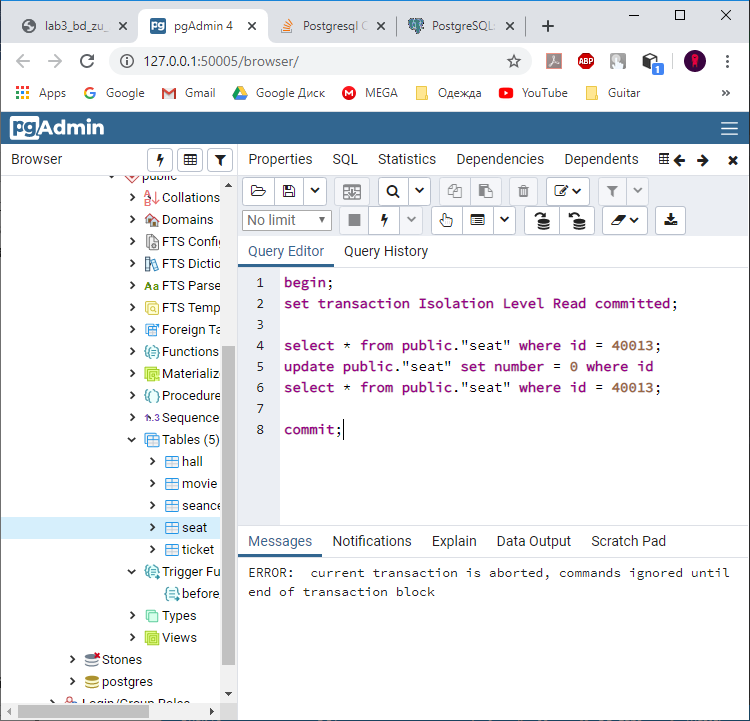


**Завдання 4:**

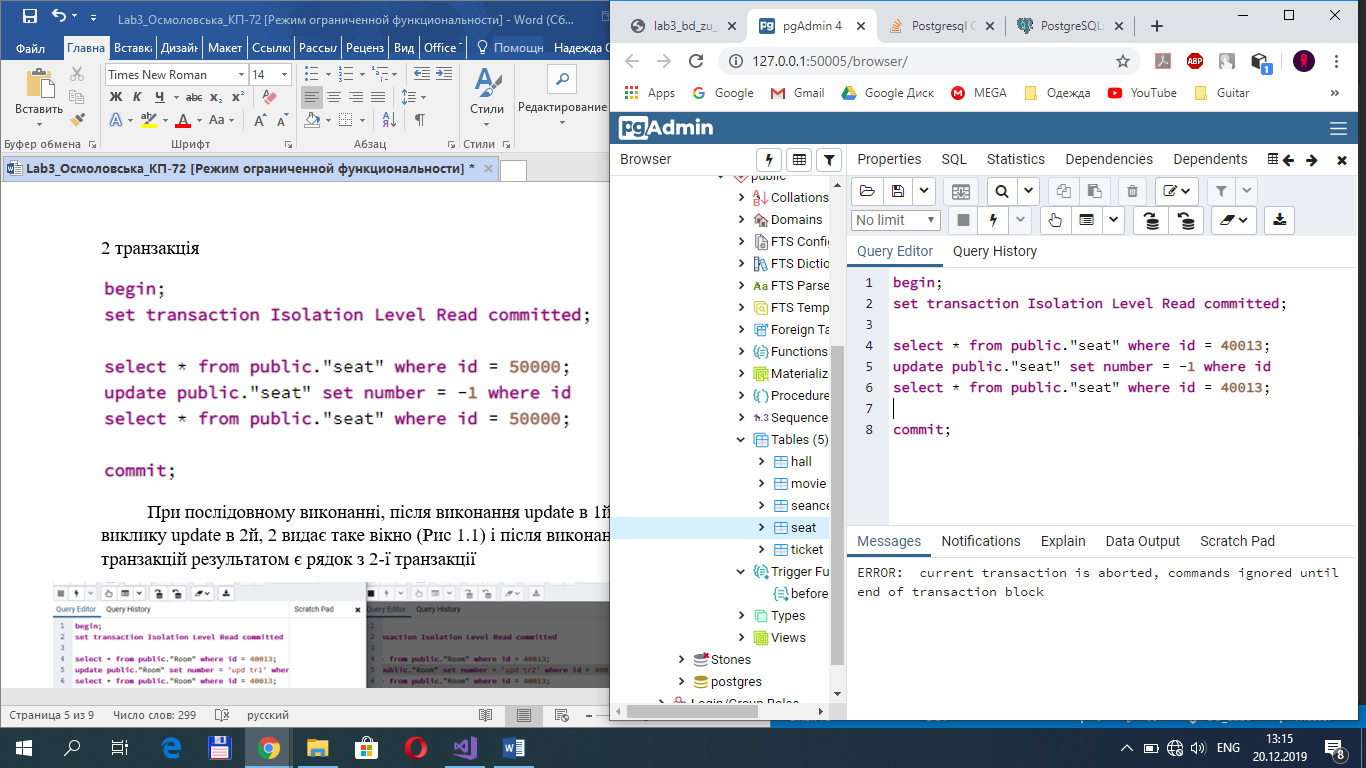
1. Read Uncommitted

Якщо кілька паралельних транзакцій намагаються змінювати один і той же рядок таблиці, то в остаточному варіанті рядок буде мати значення, визначене останньою успішно виконаною транзакцією - **Втрачене оновлення (Lost Update)**

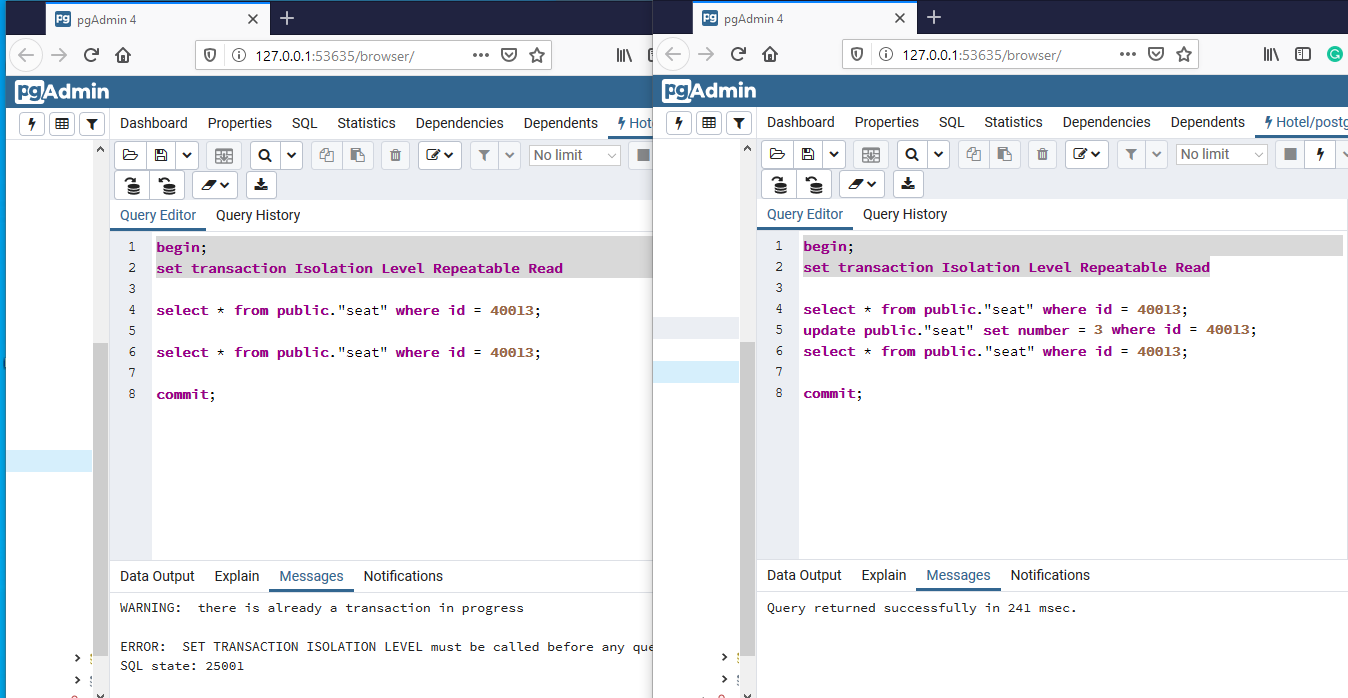
1 транзакція



2 транзакція



При послідовному виконанні, після виконання update в 1й і виклику update в 2й, 2 видає таке вікно (Рис 1.1) і після виконання обох транзакцій результатом є рядок з 2-ї транзакції

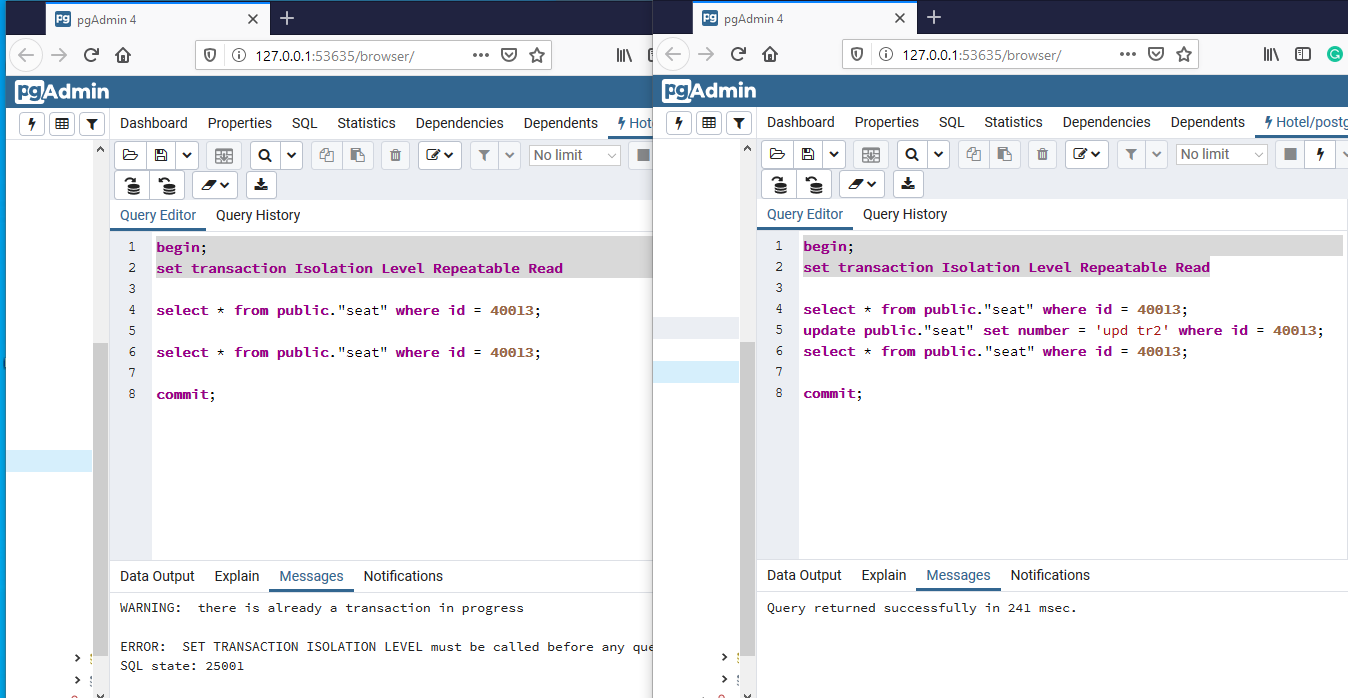


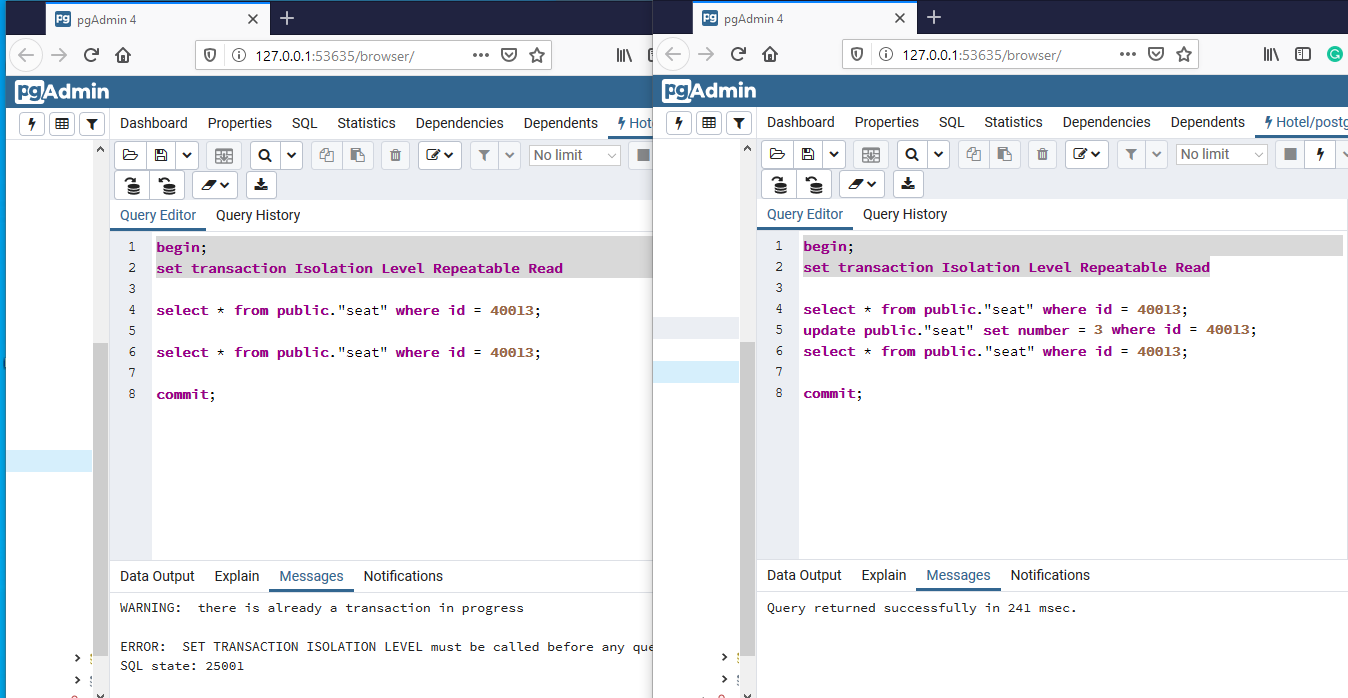
1. Repeatable Read

Всі оператори поточної транзакції бачать тільки ті рядки, які були зафіксовані перед першим запитом на вибірку або зміну даних, виконаним в цій транзакції.

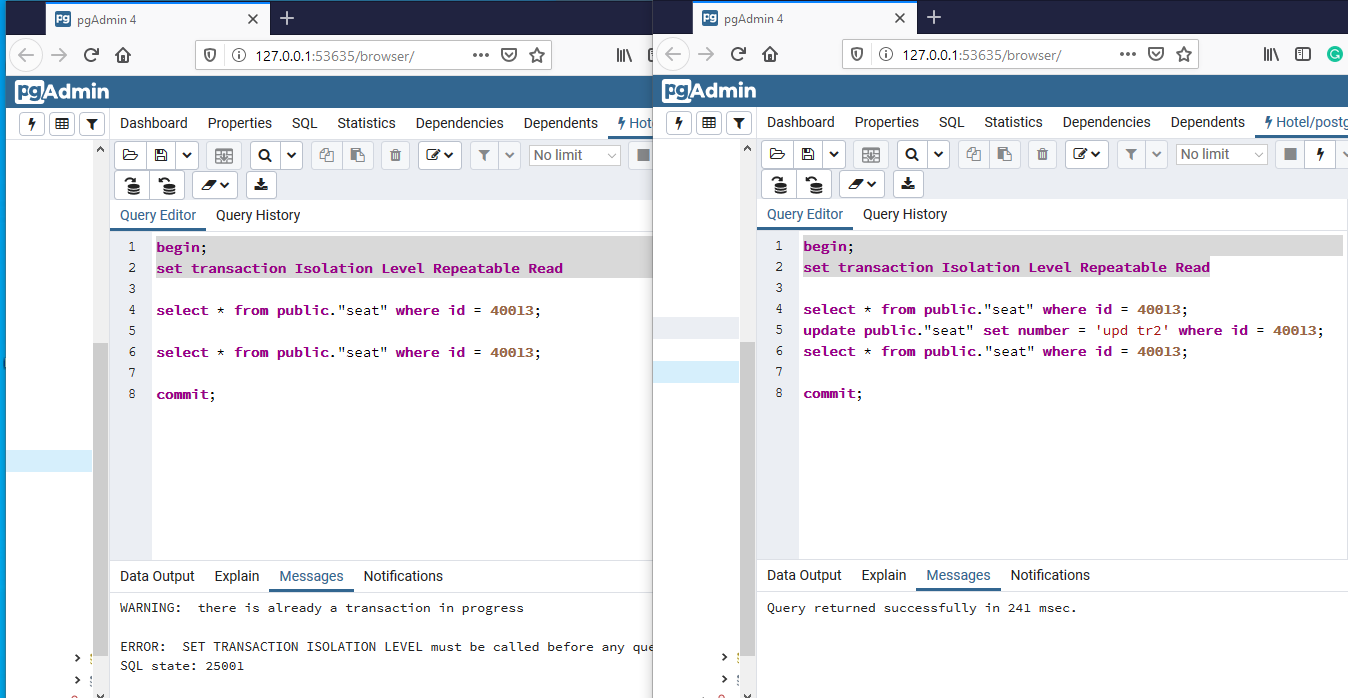
Виконаємо дві наступні транзакції, і побачимо, що по завершенню 2-ї, перша все ще не «бачитиме» змін, що відбулись.

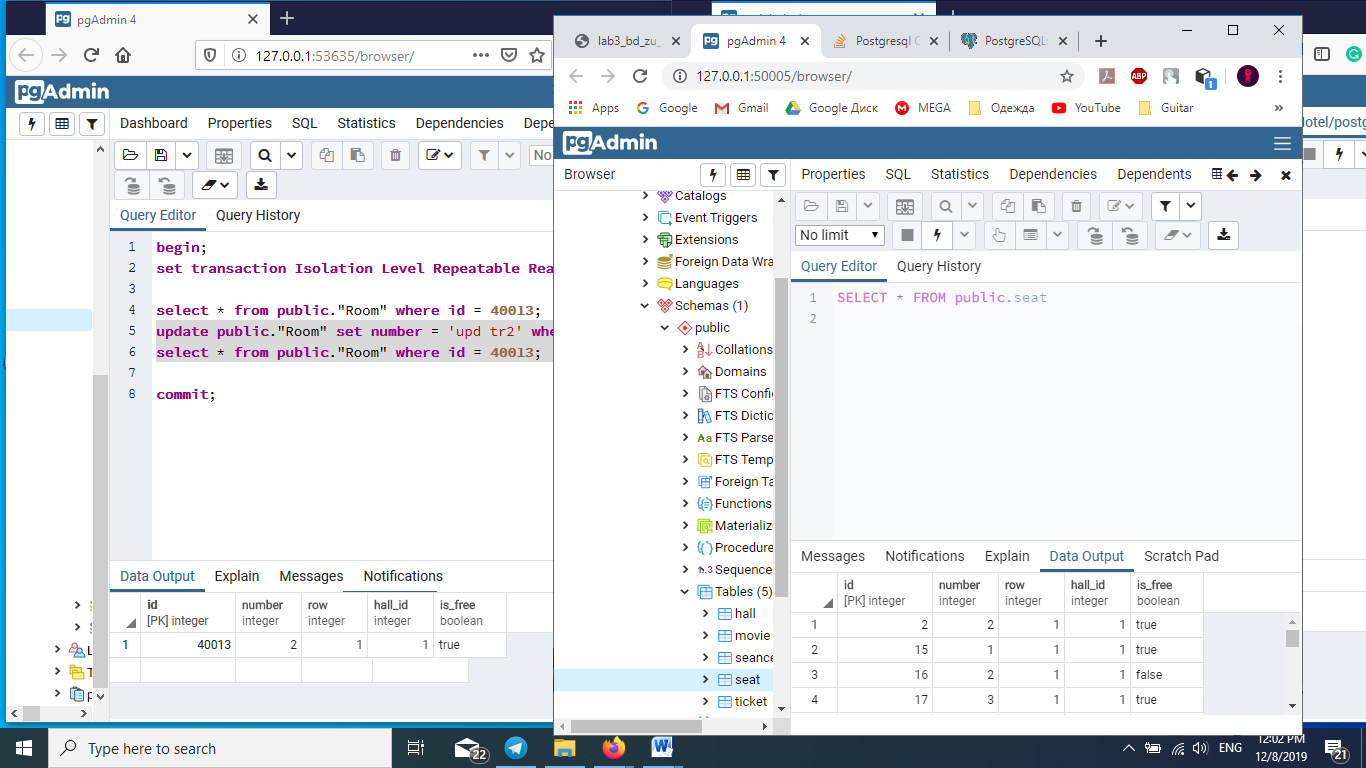
**1 крок :**

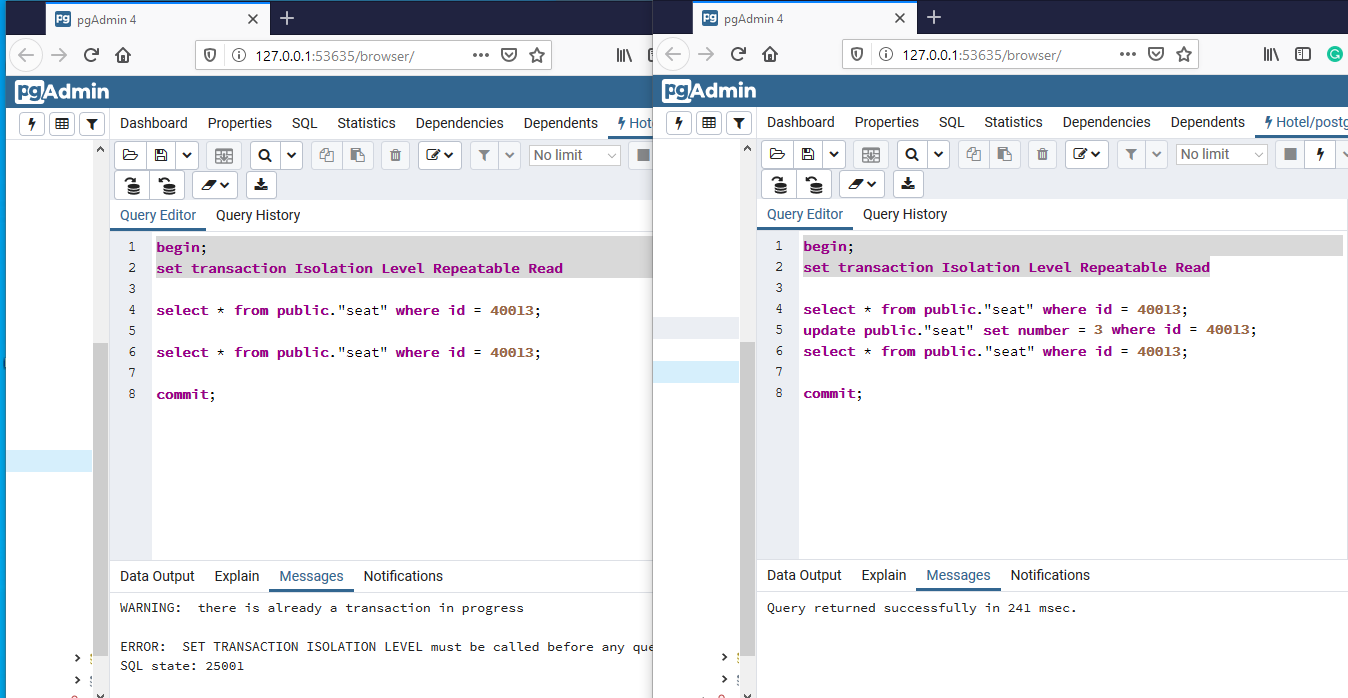
****

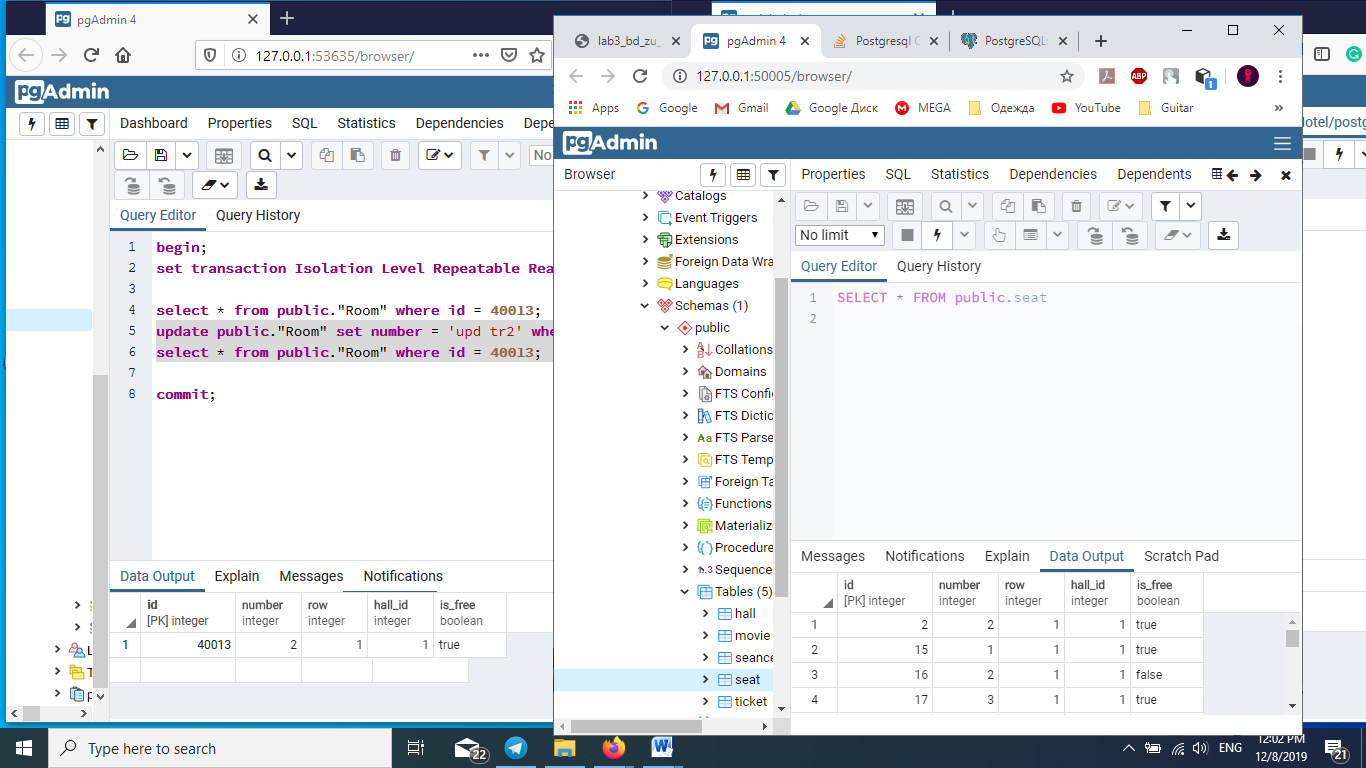


**2 крок:**

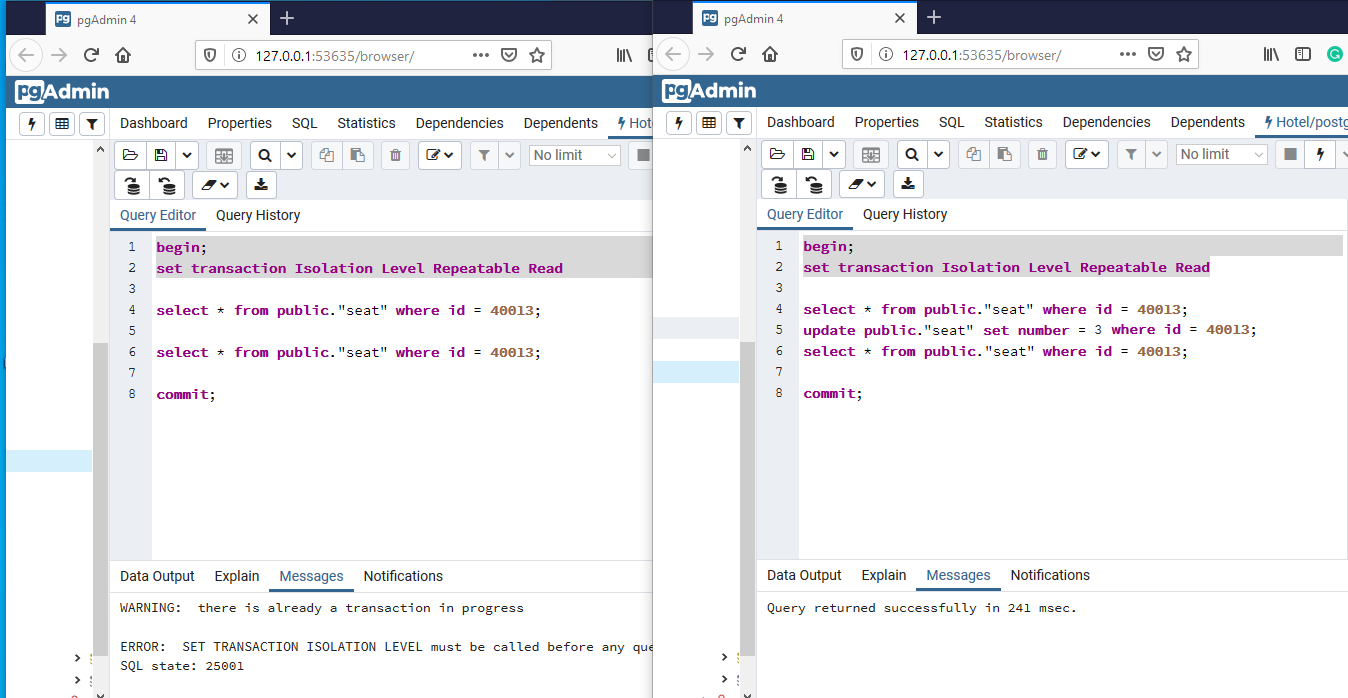
****

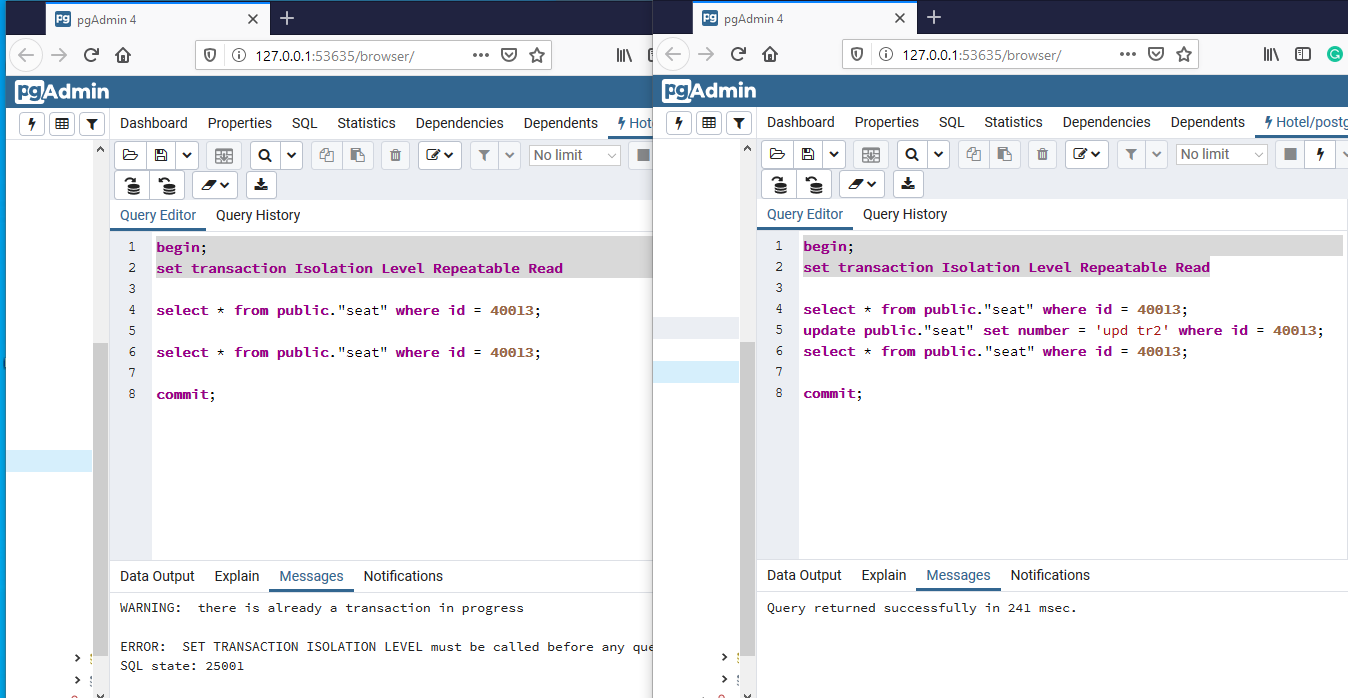


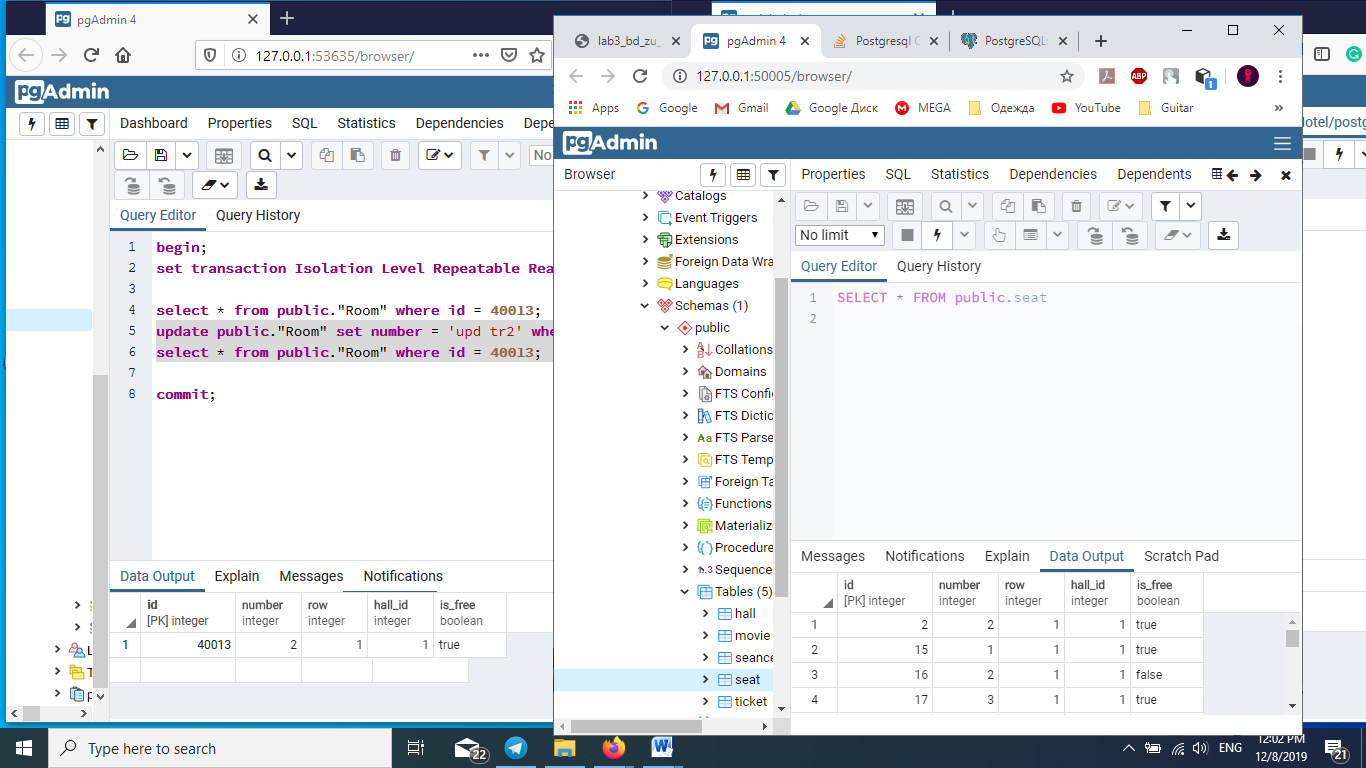




**3 крок:**



****



Тобто 1-ша транзакція не бачить змін, які відбулись у 2-й