Лабораторна робота № 4. Створення запитів на групування, сортування, використання вбудованих функцій. Створення та керування представленнями.

Виконавець: ІП-11 Прищепа Владислав Станіславович

Варіант: 25

Мета: − Вивчити оператор, котрий використовується в реляційних СУБД, для вибірки даних з таблиць, групування та сортування даних

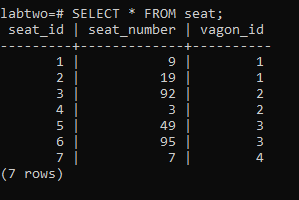
− Навчитись використовувати вбудовані функції в запитах

− Вивчити призначення представлень (view) баз даних, синтаксису та семантики команд SQL для їх створення, зміни та видалення, системних збережених процедур для отримання інформації про представлення

1) Створити наступні запити:

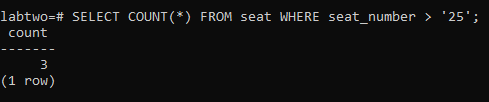
a. запит з використанням функції COUNT;

Початкова таблиця seat:



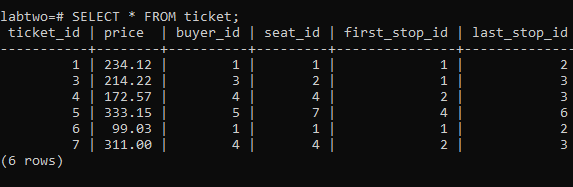
Підрахуємо кількість рядків таблиці seat, де seat\_number > 25:

SELECT COUNT(\*) FROM seat WHERE seat\_number > '25';



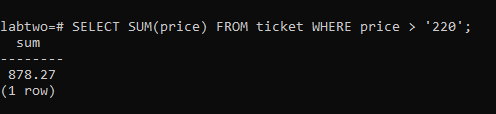
b. запит з використанням функції SUM;

Початкова таблиця ticket:



Підрахуємо суму значень стовпця price таблиці ticket, що білше 220:

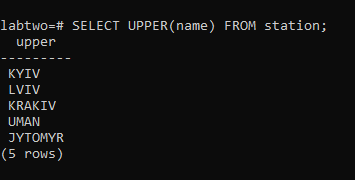
SELECT SUM(price) FROM ticket WHERE price > '220';



c. запит з використанням функцій UPPER, LOWER;

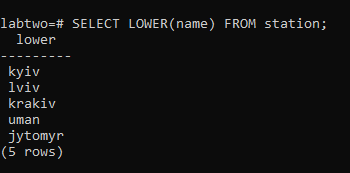
Виведемо стовпець name таблиці station, але всі літери великі:

SELECT UPPER(name) FROM station;



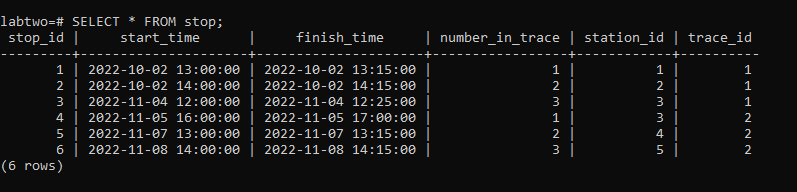
Виведемо стовпець name таблиці station, але всі літери маленькі:

SELECT LOWER(name) FROM station;



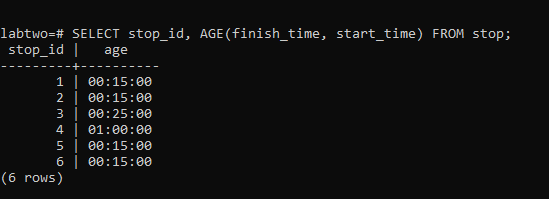
d. запит з використанням функцій для роботи з датами;

Початкова таблиця stop:



Виведемо для кожної зупинки тривалість самої зупинки:

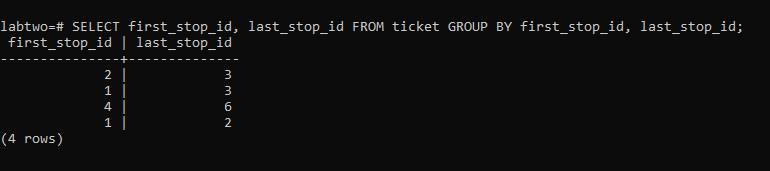
SELECT stop\_id, AGE(finish\_time, start\_time) FROM stop;



e. запит з використанням групування по декільком стовпцям;

Згрупуємо рядки таблиці ticket по first\_stop\_id, last\_stop\_id та виведемо утворені унікальні пари значень first\_stop\_id, last\_stop\_id:

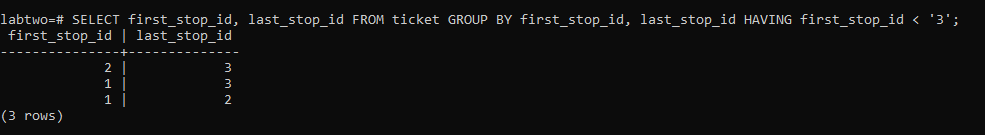
SELECT first\_stop\_id, last\_stop\_id FROM ticket GROUP BY first\_stop\_id, last\_stop\_id;



f. запит з використанням умови відбору груп HAVING;

Згрупуємо рядки таблиці ticket по first\_stop\_id, last\_stop\_id та виведемо утворені унікальні пари значень first\_stop\_id, last\_stop\_id, де first\_stop\_id < 3:

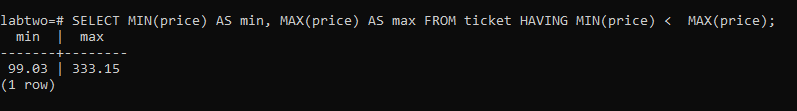
SELECT first\_stop\_id, last\_stop\_id FROM ticket GROUP BY first\_stop\_id, last\_stop\_id HAVING first\_stop\_id < '3';



g. запит з використанням HAVING без GROUP BY;

Виведемо максимальне й мінімальне значення price таблиці ticket за умови, що мінімальне значення менше максимального:

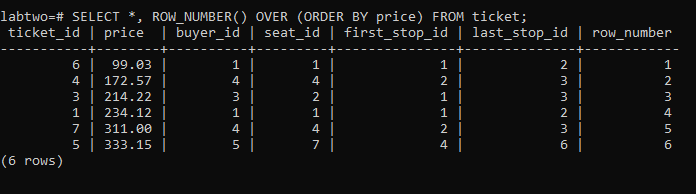
SELECT MIN(price) AS min, MAX(price) AS max FROM ticket HAVING MIN(price) < MAX(price);



h. запит з використанням функцій row\_number() over ….;

Виведемо відсортовані рядки за price рядки таблиці ticket та пронумеруємо їх:

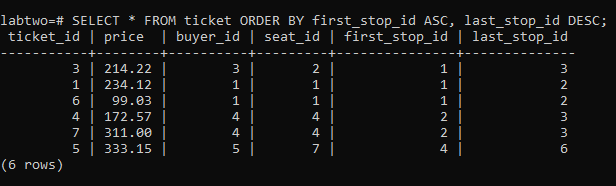
SELECT \*, ROW\_NUMBER() OVER (ORDER BY price) FROM ticket;



i. запит з використанням сортування по декільком стовпцям.

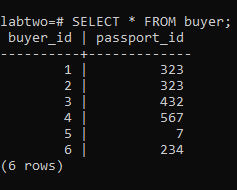
Відсортуємо рядки таблиці ticket за стовбцями first\_stop\_id (за зростанням), last\_stop\_id (за спаданням):

SELECT \* FROM ticket ORDER BY first\_stop\_id ASC, last\_stop\_id DESC;



2) Робота з представленнями (view):

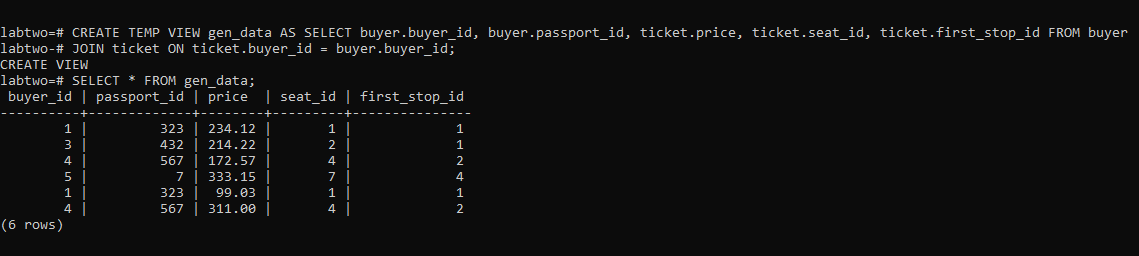
Початкова таблиця buyer:



a. створити представлення, котре містить дані з декількох таблиць;

Створимо представлення gen\_data, що містить значення з стовпців buyer.buyer\_id, buyer.passport\_id, ticket.price, ticket.seat\_id, ticket.first\_stop\_id з таблиць buyer та ticket, для рядків яких виконується рівність ticket.buyer\_id = buyer.buyer\_id:

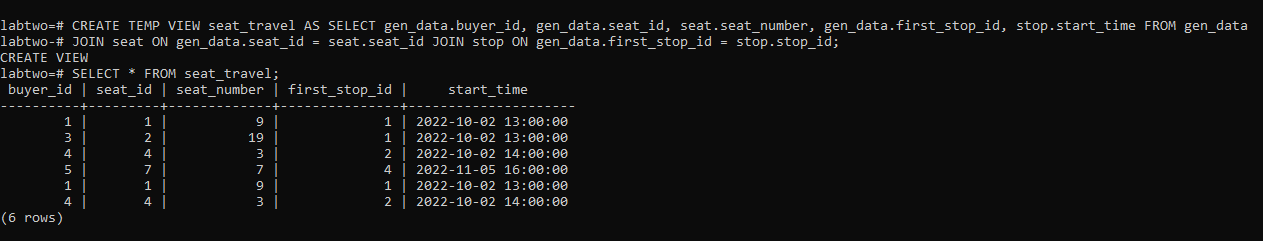
CREATE TEMP VIEW gen\_data AS SELECT buyer.buyer\_id, buyer.passport\_id, ticket.price, ticket.seat\_id, ticket.first\_stop\_id FROM buyer JOIN ticket ON ticket.buyer\_id = buyer.buyer\_id;



b. створити представлення, котре містить дані з декількох таблиць та використовує представлення, котре створене в п.a;

Створимо представлення seat\_travel, що містить значення з стовпців gen\_data.buyer\_id, gen\_data.seat\_id, seat.seat\_number, gen\_data.first\_stop\_id, stop.start\_time з представлення gen\_data та таблиць seat та stop, для рядків яких виконуються рівності gen\_data.seat\_id = seat.seat\_id та gen\_data.first\_stop\_id = stop.stop\_id:

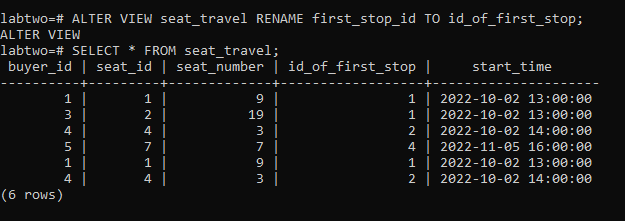
CREATE TEMP VIEW seat\_travel AS SELECT gen\_data.buyer\_id, gen\_data.seat\_id, seat.seat\_number, gen\_data.first\_stop\_id, stop.start\_time FROM gen\_data JOIN seat ON gen\_data.seat\_id = seat.seat\_id JOIN stop ON gen\_data.first\_stop\_id = stop.stop\_id;



c. модифікувати представлення з використанням команди ALTER VIEW;

Перейменуємо у представленні seat\_travel стовпець first\_stop\_id на id\_of\_first\_stop:

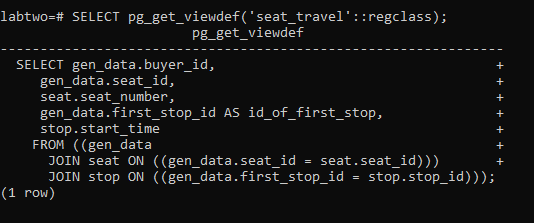
ALTER VIEW seat\_travel RENAME first\_stop\_id TO id\_of\_first\_stop;



d. отримати довідникову інформацію про ці представлення з використанням вбудованих процедур (наприклад в MsSQL sp\_help, sp\_helptext та sp\_depends).

Виведемо інформацію про представлення 'seat\_travel':

SELECT pg\_get\_viewdef('seat\_travel'::regclass);



Висновок: під час виконання лабораторної роботи я навчився створювати запити на вибірки даних, їх обробку, створювати представлення.