

Лабораторна робота

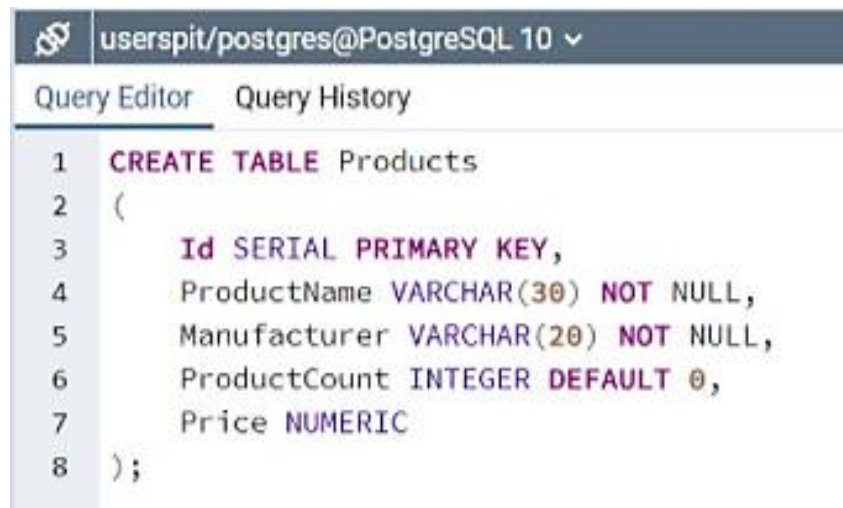
Тема: Операції з даними в PostgreSQL.

Мета: Операції з даними.

Хід роботи

1. Додавання даних. Команда Insert:

1) Припустимо, у нас в базі даних є наступна таблиця:



```
userspit/postgres@PostgreSQL 10 ▾
Query Editor  Query History
1  CREATE TABLE Products
2  (
3      Id SERIAL PRIMARY KEY,
4      ProductName VARCHAR(30) NOT NULL,
5      Manufacturer VARCHAR(20) NOT NULL,
6      ProductCount INTEGER DEFAULT 0,
7      Price NUMERIC
8  );
```

Рисунок 1.1 – Таблиця Products

2) Додамо в неї один рядок за допомогою команди INSERT:

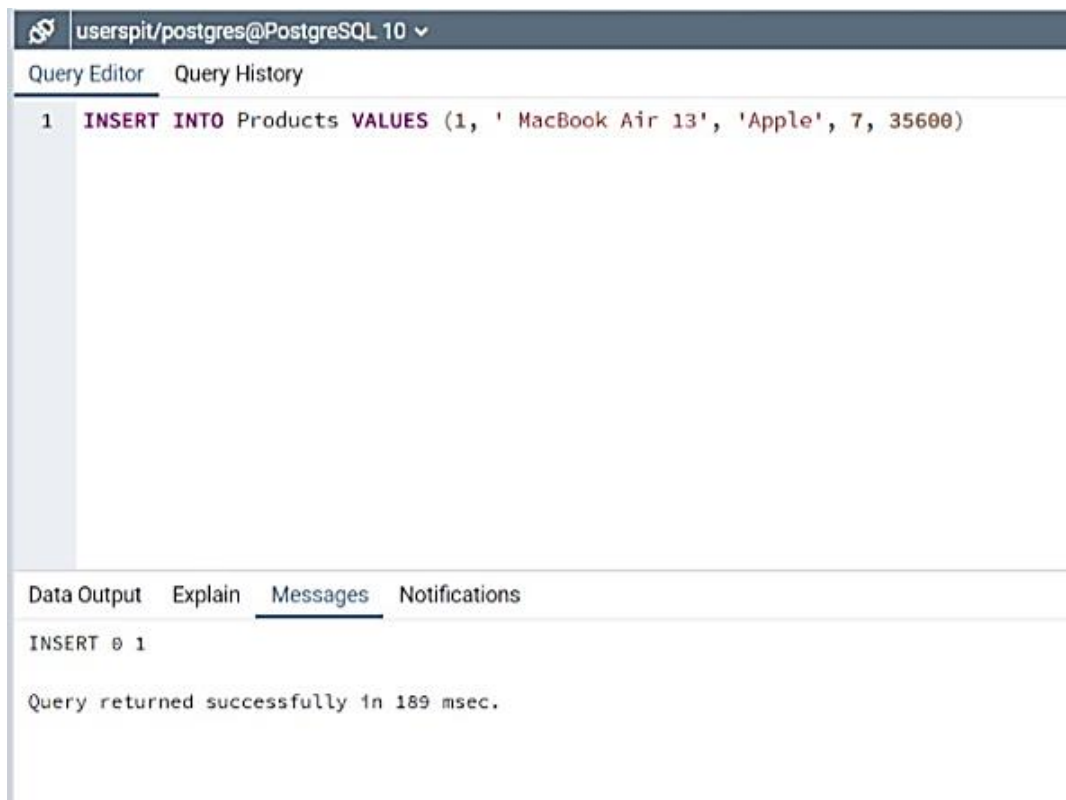


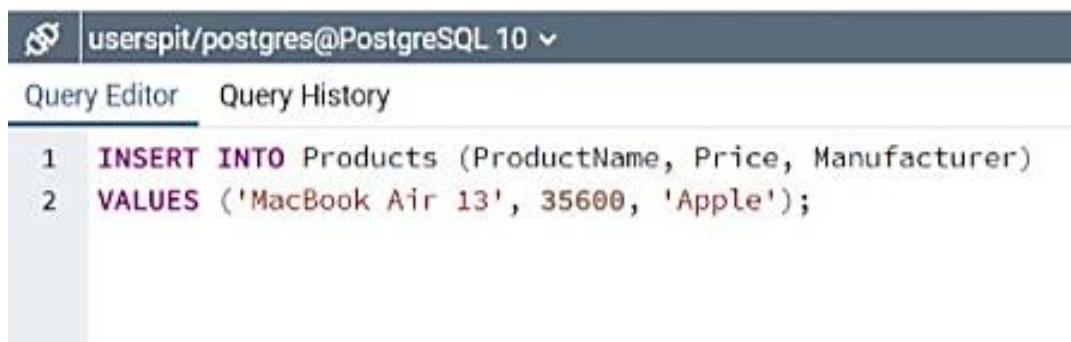
Рисунок 1.2 – Команди INSERT

Після вдалого виконання в pgAdmin в поле повідомлень має з'явитися повідомлення "INSERT 0 1":

Варто враховувати, що значення для стовпців в дужках після ключового слова VALUES передаються по порядку їх оголошення. Наприклад, у виразі CREATE TABLE вище можна побачити, що першим стовпцем йде Id, тому цьому стовпцю передається 1. Другий стовпець називається ProductName, тому друге значення - рядок "MacBook Air 13" буде передано саме цьому стовпцю і так далі. Тобто значення передаються стовпцям наступним чином:

- Id: 1
- ProductName: MacBook Air 13'
- Manufacturer: 'Apple'
- Productscount: 7
- Price: 35600

Також при введенні значень можна вказати безпосередні стовпці, в які будуть додаватися значення:



```
userspit/postgres@PostgreSQL 10 ▾  
Query Editor  Query History  
1  INSERT INTO Products (ProductName, Price, Manufacturer)  
2  VALUES ('MacBook Air 13', 35600, 'Apple');
```

Рисунок 1.3 – Стовпці

Тут значення вказується тільки для трьох стовпців. Причому тепер значення передаються в порядку проходження стовпців:

ProductName: 'MacBook Air 13'

Manufacturer: 'Apple'

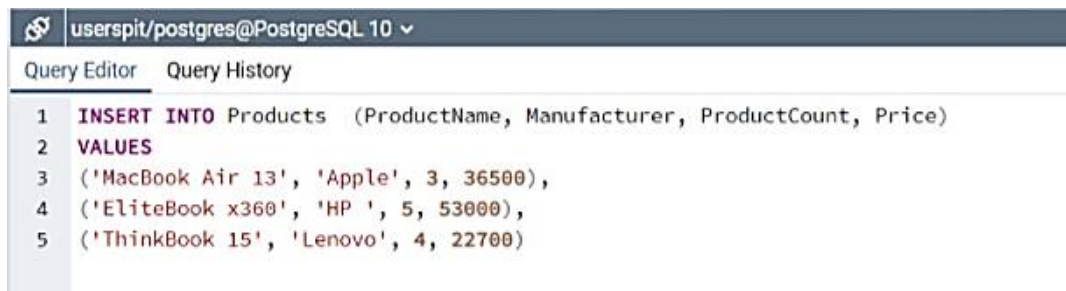
Price: 35600

Для стовпця Id значення буде генеруватися автоматично базою даних, так як він представляє тип Serial. Тобто до значення з останнього рядка буде додаватися одиниця.

Для інших стовпців буде додаватися значення за замовчуванням, якщо заданий атрибут DEFAULT (наприклад, для стовпця ProductCount), значення NULL. При цьому не зазначені стовпці (за винятком тих, які мають тип Serial) повинні допускати значення NULL або мати атрибут DEFAULT.

Якщо конкретні стовпці не вказуються, як у першому прикладі, тоді ми повинні передати значення для всіх стовпців у таблиці.

Також ми можемо додати відразу кілька рядків:



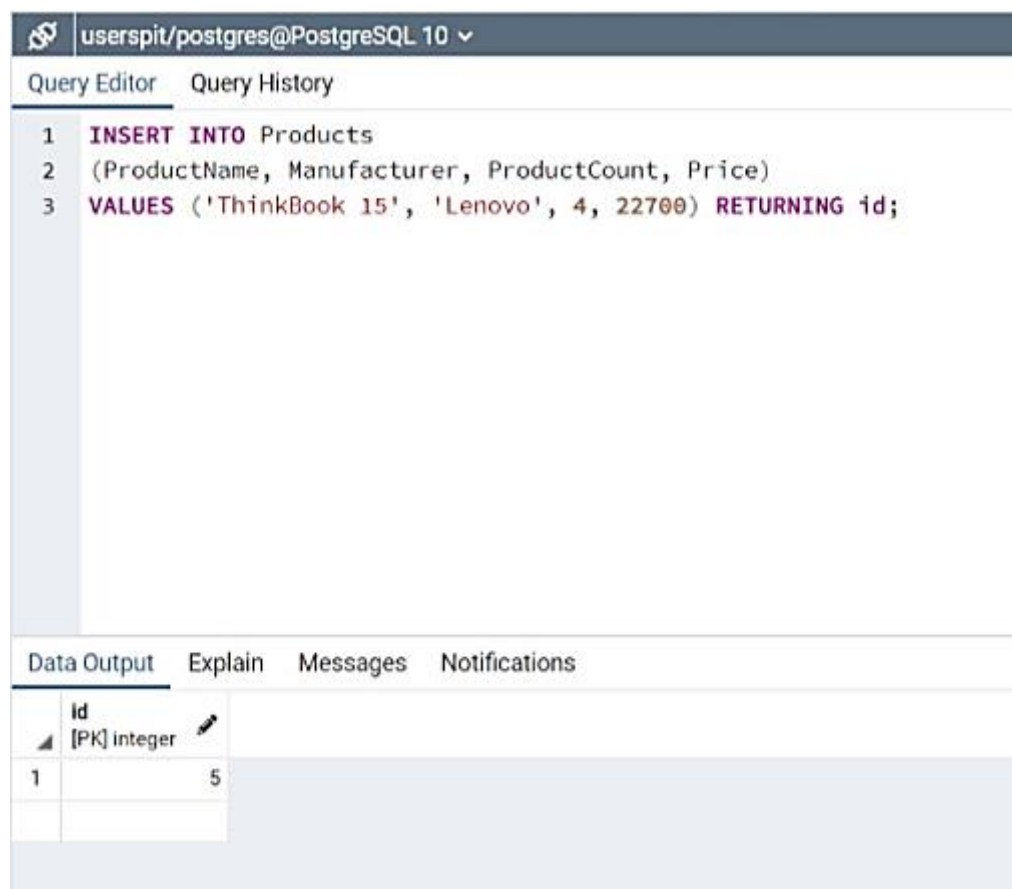
```
userspit/postgres@PostgreSQL 10
Query Editor  Query History
1  INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)
2  VALUES
3  ('MacBook Air 13', 'Apple', 3, 36500),
4  ('EliteBook x360', 'HP ', 5, 53000),
5  ('ThinkBook 15', 'Lenovo', 4, 22700)
```

Рисунок 1.4 – Додаємо відразу кілька рядків

В даному випадку в таблицю будуть додані три рядки.

3) Повернення значень

Якщо ми додаємо значення тільки для частини стовпців, то ми можемо не знати, які значення будуть у інших стовпців. Наприклад, яке значення отримає стовець Id у товару. За допомогою оператора RETURNING ми можемо отримати це значення:



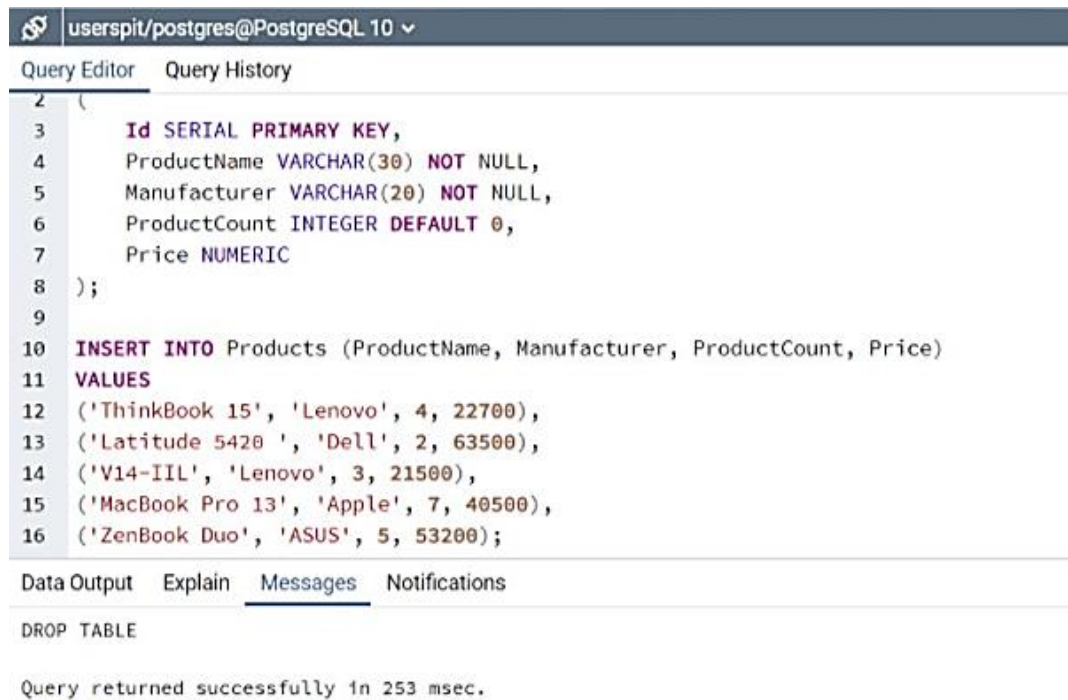
```
userspit/postgres@PostgreSQL 10
Query Editor  Query History
1  INSERT INTO Products
2  (ProductName, Manufacturer, ProductCount, Price)
3  VALUES ('ThinkBook 15', 'Lenovo', 4, 22700) RETURNING id;
```

Data Output		Explain	Messages	Notifications
	Id [PK] integer			
1	5			

Рисунок 1.5 – Повернення значень

2. Отримання даних. Команда Select

1) Наприклад, нехай раніше була створена таблиця Products, і в неї додані деякі початкові Дані:



```
userspit/postgres@PostgreSQL 10
Query Editor  Query History
2  (
3      Id SERIAL PRIMARY KEY,
4      ProductName VARCHAR(30) NOT NULL,
5      Manufacturer VARCHAR(20) NOT NULL,
6      ProductCount INTEGER DEFAULT 0,
7      Price NUMERIC
8  );
9
10 INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)
11 VALUES
12 ('ThinkBook 15', 'Lenovo', 4, 22700),
13 ('Latitude 5420 ', 'Dell', 2, 63500),
14 ('V14-IIL', 'Lenovo', 3, 21500),
15 ('MacBook Pro 13', 'Apple', 7, 40500),
16 ('ZenBook Duo', 'ASUS', 5, 53200);

Data Output  Explain  Messages  Notifications
DROP TABLE

Query returned successfully in 253 msec.
```

Рисунок 2.1 – Таблиця Products

Отримаємо всі об'єкти з цієї таблиці:

userspit/postgres@PostgreSQL 10 ▾						
Query Editor Query History						
1 SELECT * FROM Products;						
Data Output Explain Messages Notifications						
	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric	
1	1	ThinkBook 15	Lenovo	4	22700	
2	2	Latitude 5420	Dell	2	63500	
3	3	V14-IIL	Lenovo	3	21500	
4	4	MacBook Pro 13	Apple	7	40500	
5	5	ZenBook Duo	ASUS	5	53200	

Рисунок 2.2 – Команда Select

Символ Зірочка * вказує, що нам треба отримати всі стовпці.

Однак використання символу зірочки * вважається не дуже хорошою практикою, так як, як правило, не всі стовпці бувають потрібні. І більш оптимальний підхід полягає у вказівці всіх необхідних стовпців після слова SELECT. Виняток становить той випадок, коли треба отримати дані по абсолютно всіх стовпцях таблиці. Також використання символу * може бути переважно в таких ситуаціях, коли в точності не відомі назви стовпців.

Якщо нам треба отримати дані не по всіх, а по якихось конкретних стовпцях, то тоді всі ці специфікації стовпців перераховуються через кому після SELECT:

userspit/postgres@PostgreSQL 10

Query Editor

Query History

1

SELECT ProductName, Price FROM Products;

Data Output

Explain


Messages

Notifications

	productname character varying (30)	price numeric
1	ThinkBook 15	22700
2	Latitude 5420	63500
3	V14-IIL	21500
4	MacBook Pro 13	40500
5	ZenBook Duo	53200

Рисунок 2.3 – Конкретні стовпці Select

Специфікація стовпця необов'язково повинна представляти його назву. Це може бути будь-який вираз, наприклад, результат арифметичної операції. Так, виконаємо наступний запит:

userspit/postgres@PostgreSQL 10 ▾

Query Editor

Query History

1

SELECT ProductCount **AS** Title,

2

Manufacturer,

3

Price * ProductCount **AS** TotalSum

4

FROM Products;

5

Data Output

Explain

Messages

Notifications

	<div>title</div> <div>integer</div>	<div>manufacturer</div> <div>character varying (20)</div>	<div>totalsum</div> <div>numeric</div>
1	4	Lenovo	90800
2	2	Dell	127000
3	3	Lenovo	64500
4	7	Apple	283500
5	5	ASUS	266000

Рисунок 2.5 – Оператор AS

3. Фільтрація. WHERE

Для фільтрації даних застосовується оператор WHERE, після якого вказується умова, на підставі якого проводиться фільтрація:

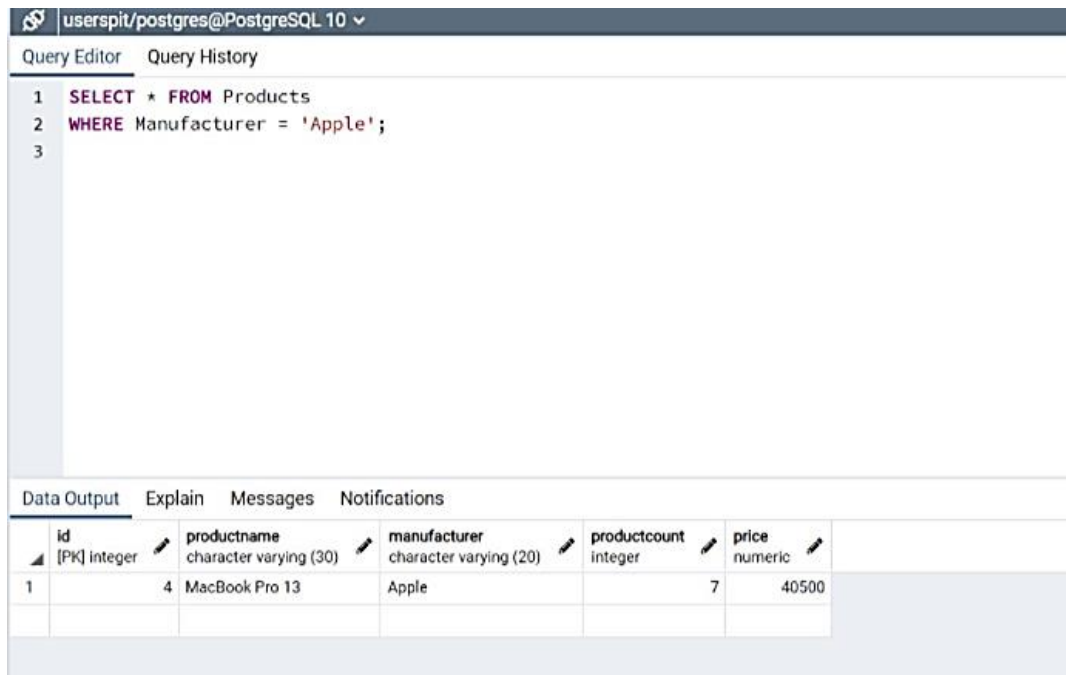
WHERE умова

Якщо умова істинна, то рядок потрапляє в результуючу вибірку. Як можна використовувати операції порівняння. Ці операції порівнюють два вирази. У PostgreSQL можна застосовувати такі операції порівняння:

- = : порівняння на рівність
- < > : порівняння на нерівність
- < : менше ніж
- : більше ніж
- !< : не менше ніж

- !> : не більше ніж
- < = : менше ніж або дорівнює
- = : більше ніж або дорівнює

1) Наприклад, знайдемо всі товари, виробником яких є компанія Apple:



userspit/postgres@PostgreSQL 10				
Query Editor		Query History		
1	SELECT * FROM Products			
2	WHERE Manufacturer = 'Apple';			
3				

Data Output					Explain	Messages	Notifications
	id	productname	manufacturer	productcount	price		
	[PK] integer	character varying (30)	character varying (20)	integer	numeric		
1	4	MacBook Pro 13	Apple	7	40500		

Рисунок 3.1 – Товари Apple

Варто відзначити, що в даному випадку велике значення має регістр символів, наприклад, рядок "Apple" не еквівалентна рядку "APPLE" або "apple".

Інший приклад-знайдемо всі товари, у яких ціна менше 37000:

userspit/postgres@PostgreSQL 10 ▾

Query Editor Query History

```
1 SELECT * FROM Products
2 WHERE Price < 37000;
3
```

Data Output Explain Messages Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric	
1	1	ThinkBook 15	Lenovo	4	22700	
2	3	V14-IIL	Lenovo	3	21500	

Рисунок 3.2 – Товари у яких ціна менше 37000

В якості умови можуть використовуватися і більш складні вирази. Наприклад, знайдемо всі товари, у яких сукупна вартість більше 80 000:

userspit/postgres@PostgreSQL 10

Query Editor

Query History

1

SELECT * FROM Products

2

WHERE Price * ProductCount > 80000;

3

Data Output

Explain

Messages

Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	ThinkBook 15	Lenovo	4	22700
2	2	Latitude 5420	Dell	2	63500
3	4	MacBook Pro 13	Apple	7	40500
4	5	ZenBook Duo	ASUS	5	53200

Рисунок 3.2 – Знайдемо всі товари, у яких сукупна вартість більше 80 000

Первинний ключ унікально ідентифікує рядок у таблиці. В якості первинного ключа необов'язково повинні виступати стовпці з типом SERIAL, вони можуть представляти будь-який інший тип.

2) Логічний оператор

Щоб об'єднати декілька умов в одне, в PostgreSQL можна використовувати логічні оператори:

- AND: операція логічного і. вона об'єднує два вирази:
вираз1 AND вираз2

Тільки якщо обидва ці вирази одночасно істинні, то і загальна умова оператора AND також буде істинно. Тобто якщо і перша умова істинно, і друге.

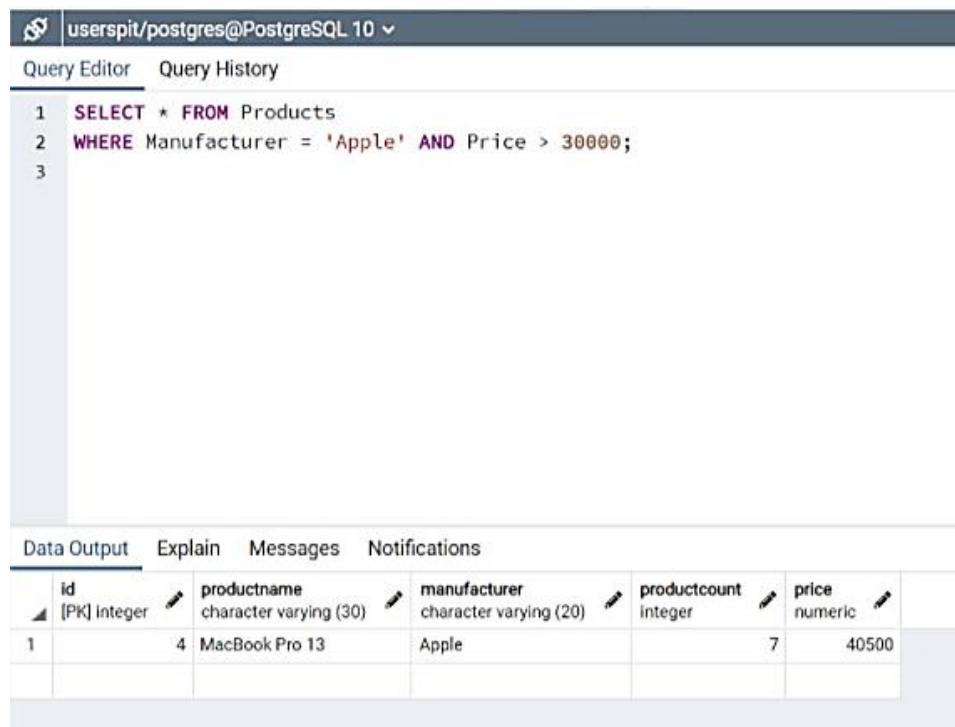
- OR: операція логічного або. Вона також об'єднує два вирази:
вираз1 OR вираз2

Якщо хоча б одне з цих виразів істинно, то загальна умова оператора OR також буде істинно. Тобто якщо або перша умова істинно, або друге.

- NOT: операція логічного заперечення. Якщо вираз в цій операції помилково, то загальна умова істинно.

NOT вираз

Наприклад, виберемо всі товари, у яких виробник Apple і одночасно Ціна більше 30000:



The screenshot shows a PostgreSQL query editor interface. The top bar indicates the user is 'userspit/postgres@PostgreSQL 10'. Below the bar, there are tabs for 'Query Editor' and 'Query History'. The 'Query Editor' tab is active, displaying a SQL query:


```
1 SELECT * FROM Products
2 WHERE Manufacturer = 'Apple' AND Price > 30000;
3
```

Below the query editor, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the following data:

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	4	MacBook Pro 13	Apple	7	40500

Рисунок 3.3 – Виберемо всі товари, у яких виробник Apple і одночасно Ціна більше 30000

Тепер змінимо оператор на OR. Тобто виберемо всі товари, у яких або виробник Apple, або ціна більше 30000:


userspit/postgres@PostgreSQL 10 ▾

Query Editor
Query History

```

1 SELECT * FROM Products
2 WHERE Manufacturer = 'Apple' OR Price > 30000;
3

```

Data Output
Explain
Messages
Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	2	Latitude 5420	Dell	2	63500
2	4	MacBook Pro 13	Apple	7	40500
3	5	ZenBook Duo	ASUS	5	53200

Рисунок 3.4 – Товари, у яких або виробник Apple, або ціна більше 30000

Застосування оператора NOT-виберемо всі товари, у яких виробник не Apple:

userspit/postgres@PostgreSQL 10

Query Editor

Query History

1

SELECT * FROM Products

2

WHERE NOT Manufacturer = 'Apple';

3

Data Output

Explain

Messages

Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	ThinkBook 15	Lenovo	4	22700
2	2	Latitude 5420	Dell	2	63500
3	3	V14-IIL	Lenovo	3	21500
4	5	ZenBook Duo	ASUS	5	53200

Рисунок 3.5 – Виберемо всі товари, у яких виробник не Apple

Але в більшості випадків цілком можна обійтися без оператора NOT. Так, в попередній приклад ми можемо переписати наступним чином:

userspit/postgres@PostgreSQL 10

Query Editor

Query History

1

SELECT * FROM Products

2

WHERE Manufacturer <> 'Apple';

3

Data Output

Explain

Messages

Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	ThinkBook 15	Lenovo	4	22700
2	2	Latitude 5420	Dell	2	63500
3	3	V14-IIL	Lenovo	3	21500
4	5	ZenBook Duo	ASUS	5	53200

Рисунок 3.6 – NOT (< >) Apple

Також в одній команді SELECT можна використовувати відразу кілька операторів:

userspit/postgres@PostgreSQL 10 ▾						
Query Editor Query History						
<pre> 1 SELECT * FROM Products 2 WHERE Manufacturer = 'Apple' OR Price > 30000 AND ProductCount > 2; 3 </pre>						
Data Output Explain Messages Notifications						
	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric	
1	4	MacBook Pro 13	Apple	7	40500	
2	5	ZenBook Duo	ASUS	5	53200	

Рисунок 3.7 – Кілька операторів SELECT

Так як оператор AND має більш високий пріоритет, то спочатку буде виконуватися підвираження Price > 30000 AND ProductCount > 2, і тільки потім оператор OR. Тобто тут вибираються товари, які на складі більше 2 і у яких одночасно Ціна більше 30000, або ті товари, виробником яких є Apple.

За допомогою дужок ми також можемо перевизначити порядок операцій:

userspit/postgres@PostgreSQL 10

Query Editor

Query History

1

SELECT * FROM Products

2

WHERE (Manufacturer = 'Apple' OR Price > 30000) AND ProductCount > 2;

3

Data Output

Explain

Messages

Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	4	MacBook Pro 13	Apple	7	40500
2	5	ZenBook Duo	ASUS	5	53200

Рисунок 3.8 – Кілька операторів дужки SELECT

3) IS NULL

Ряд стовпців може допускати значення NULL. Це значення не еквівалентно порожньому рядку". NULL представляє повну відсутність будь-якого значення. І для перевірки на наявність подібного значення застосовується оператор IS NULL.

Наприклад, виберемо всі товари, у яких не встановлено поле Product Count:

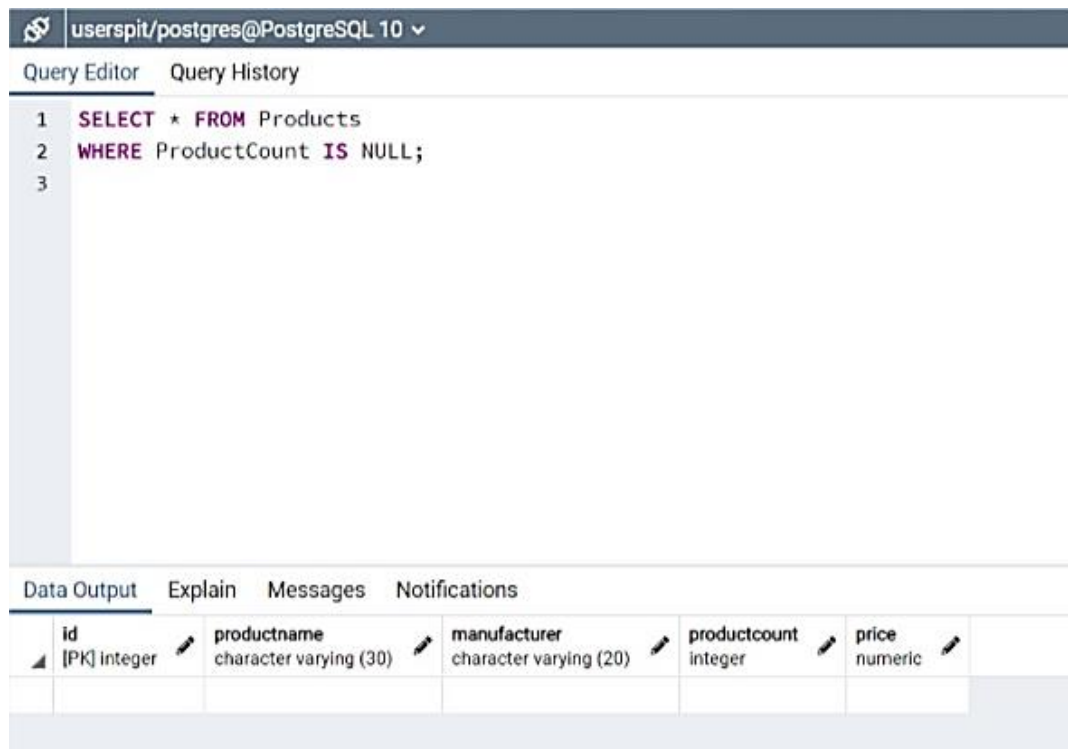


Рисунок 3.9 – Оператор IS NULL

Якщо, навпаки, необхідно отримати рядки, у яких поле Product Count не дорівнює NULL, то можна використовувати оператор NOT:

userspit/postgres@PostgreSQL 10

Query Editor

Query History

1

SELECT * FROM Products

2

WHERE ProductCount IS NOT NULL;

3

Data Output

Explain

Messages

Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	ThinkBook 15	Lenovo	4	22700
2	2	Latitude 5420	Dell	2	63500
3	3	V14-IIL	Lenovo	3	21500
4	4	MacBook Pro 13	Apple	7	40500
5	5	ZenBook Duo	ASUS	5	53200

Рисунок 3.10 – Оператор IS NOT NULL


4. Оновлення даних. Команда UPDATE

Для оновлення даних в базі даних PostgreSQL застосовується команда UPDATE.

Наприклад, збільшимо у всіх товарів ціну на 3000:

userspit/postgres@PostgreSQL 10	
Query Editor	Query History
<pre> 1 UPDATE Products 2 SET Price = Price + 3000; 3 </pre>	

Рисунок 4.1 – UPDATE


userspit/postgres@PostgreSQL 10 ▾

Query Editor
Query History

```

1 UPDATE Products
2 SET Manufacturer = 'Lenovo Ins'
3 WHERE Manufacturer = 'Lenovo';
4
5 SELECT * FROM Products;


```

Data Output
Explain
Messages
Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	2	Latitude 5420	Dell	2	69500
2	4	MacBook Pro 13	Apple	7	46500
3	5	ZenBook Duo	ASUS	5	59200
4	1	ThinkBook 15	Lenovo Ins	4	28700
5	3	V14-IIL	Lenovo Ins	3	27500

Рисунок 4.3 – Змінюємо назву виробника

Також можна оновлювати відразу кілька стовпців:


userspit/postgres@PostgreSQL 10

Query Editor
Query History

```

1 UPDATE Products
2 SET Manufacturer = 'Lenovo',
3     ProductCount = ProductCount + 3
4 WHERE Manufacturer = 'Lenovo Ins';
5
6
7 SELECT * FROM Products;

```

Data Output
Explain
Messages
Notifications


	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	2	Latitude 5420	Dell	2	69500
2	4	MacBook Pro 13	Apple	7	46500
3	5	ZenBook Duo	ASUS	5	59200
4	1	ThinkBook 15	Lenovo	7	28700
5	3	V14-IIL	Lenovo	6	27500

Рисунок 4.4 – Змінюємо назву виробника в кількох стовпцях

5. Видалення даних. Команда DELETE

1) Для видалення даних в PostgreSQL застосовується команда DELETE.

Наприклад, видалимо рядки, у яких виробник-Dell:


userspit/postgres@PostgreSQL 10

Query Editor
Query History

```

1 DELETE FROM Products
2 WHERE Manufacturer='Dell';
3
4
5 SELECT * FROM Products;


```

Data Output
Explain
Messages
Notifications

	id [PK] Integer	productname character varying (30)	manufacturer character varying (20)	productcount Integer	price numeric
1	4	MacBook Pro 13	Apple	7	46500
2	5	ZenBook Duo	ASUS	5	59200
3	1	ThinkBook 15	Lenovo	7	28700
4	3	V14-IIL	Lenovo	6	27500

Рисунок 5.1 – Змінюємо назву виробника в кількох стовпцях

Або видалимо всі товари, виробником яких є Lenovo і які мають ціну менше 28000:


userspit/postgres@PostgreSQL 10 ▾

Query Editor
Query History

```

1 DELETE FROM Products
2 WHERE Manufacturer='Lenovo' AND Price < 28000;
3
4
5 SELECT * FROM Products;

```

Data Output
Explain
Messages
Notifications

	id [PK] integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	4	MacBook Pro 13	Apple	7	46500
2	5	ZenBook Duo	ASUS	5	59200
3	1	ThinkBook 15	Lenovo	7	28700

Рисунок 5.2 – Видалення товарів

Якщо необхідно зовсім видалити всі рядки незалежно від умови, то умова можна не вказувати:

 userspit/postgres@PostgreSQL 10 ▾

Query Editor

Query History

1

DELETE FROM Products;

2

3

4

SELECT * FROM Products;

Data Output

Explain

Messages

Notifications

	Id	productname	manufacturer	productcount	price
	[PK] integer	character varying (30)	character varying (20)	integer	numeric

Рисунок 5.3 – Видалимо всі рядки