Зміст

1. SELECT

2.DML (Data Manipulation Language)

* **INSERT**
* **UPDATE**
* **DELETE**

3. DDL (Data Definition Language)

* **CREATE TABLE**
* **ALTER TABLE**
* **DROP TABLE**

4. JOINS (Об'єднання)

* **INNER JOIN (ВНУТРІШНЄ об'єднання)**
* **LEFT JOIN (ЛІВЕ З'ЄДНАННЯ)**
* **RIGHT JOIN (ПРАВЕ З'ЄДНАННЯ)**
* **FULL OUTER JOIN (ПОВНЕ ЗОВНІШНЄ З'ЄДНАННЯ)**

5. DISTINCT

6. IN та NOT IN

7. BETWEEN

8. ORDER BY Clause

9. GROUP BY

10. HAVING

11. Агрегатні функції - **Avg, Min, Max, Sum, Count**

12. =, >, AND, OR, NOT

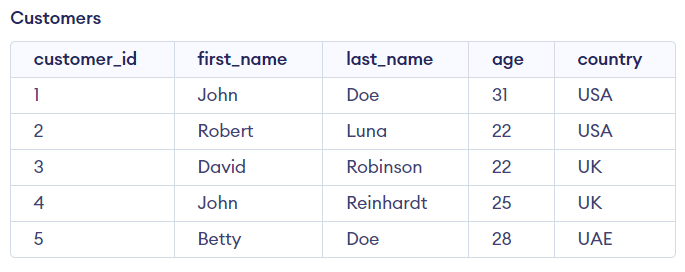
13. Alias - AS

14. LIKE та NOT LIKE

15. Wildcard (символи)

**SELECT**

**SELECT** – найбільш важливий оператор для тестування бази даних. Він повертає рядки даних з однієї або декількох таблиць, які задовольняють заданий набір критеріїв.



**1. Вибрати стовпчик first\_name з таблиці Customers**

**SELECT** first\_name

**FROM** Customers;

2**. Вибрати стовпці first\_name та last\_name з таблиці Customers**

**SELECT** first\_name, last\_name

**FROM** Customers;

**3. Вибрати всі стовпці з таблиці Customers**

**SELECT** \*

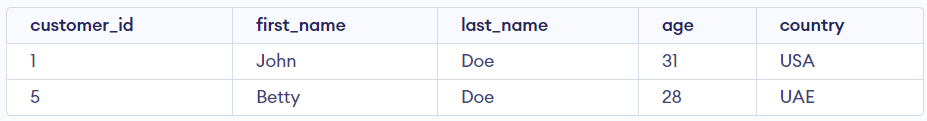
**FROM** Customers;

**4. Вибрати: всі стовпці з таблиці Customers з last\_name 'Doe'**

SELECT \*

FROM Customers

WHERE last\_name = 'Doe';



**5. Виберіть стовпці віку та країни з таблиці клієнтів, де країною є "USA"**

SELECT age, country

FROM Customers

WHERE country = 'USA';



**DML (Data Manipulation Language)**

INSERT, UPDATE, DELETE

**DML (Data Manipulation Language)** – оператори маніпулювання даними (INSERT, UPDATE, DELETE)

**INSERT**

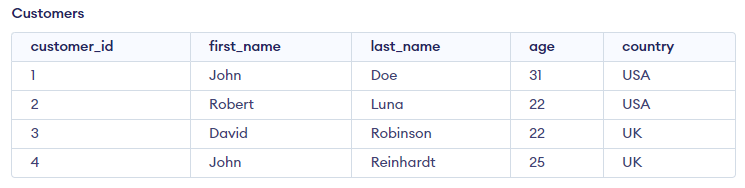
INSERT INTO - використовується для вставки нових рядків у таблицю бази даних.

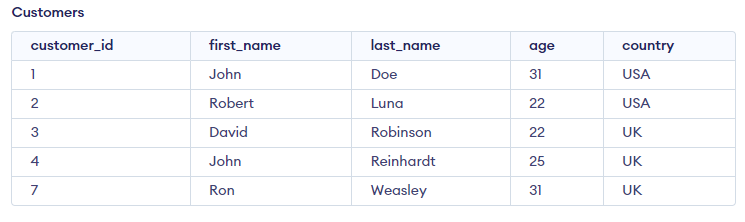
**1. Вставити рядок в таблицю Customers**

INSERT INTO Customers(customer\_id, first\_name, last\_name, age, country)

VALUES

(7, 'Ron', 'Weasley', 31, 'UK');





**2. Вставити декілька рядків в таблицю бази даних одночасно**

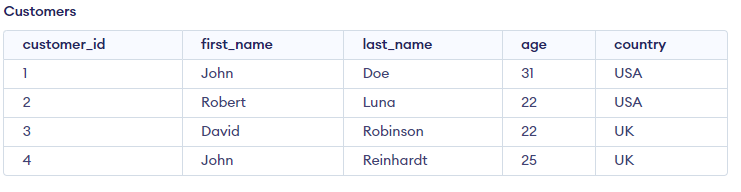
INSERT INTO Customers(first\_name, last\_name, age, country)

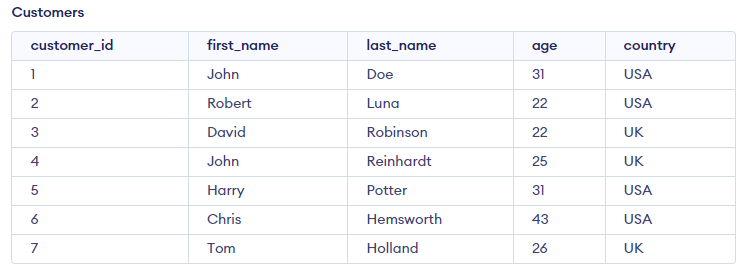
VALUES

('Harry', 'Potter', 31, 'USA'),

('Chris', 'Hemsworth', 43, 'USA'),

('Tom', 'Holland', 26, 'UK');





**UPDATE**

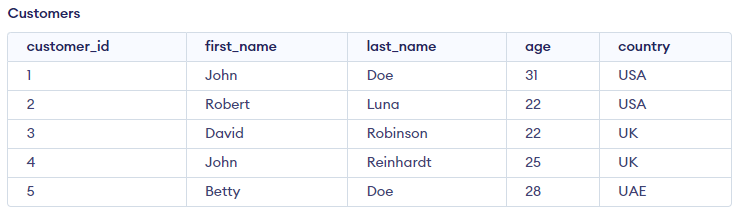
**UPDATE** - використовується для редагування існуючого рядка в таблиці бази даних

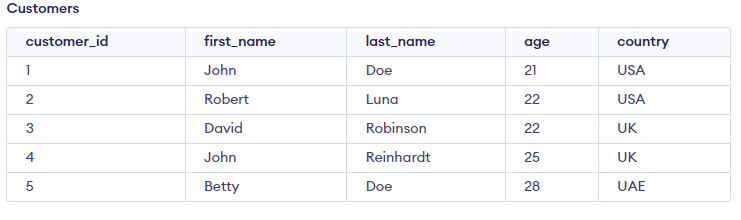
**1. Оновити єдине значення в заданому рядку**

UPDATE Customers

SET age = 21

WHERE customer\_id = 1;



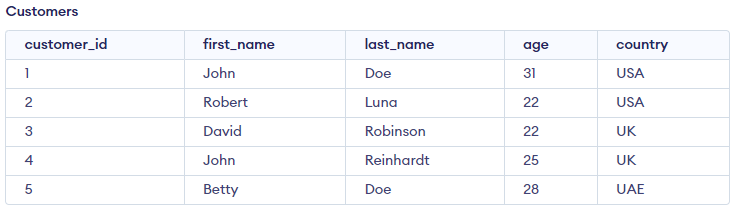


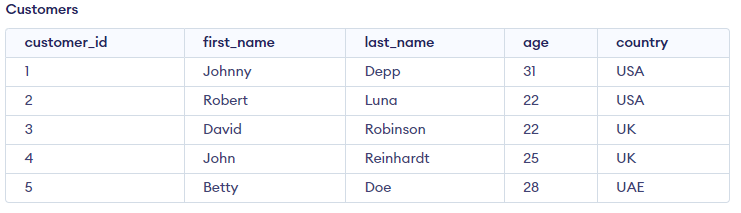
**2. Оновити декілька значень у заданому рядку**

UPDATE Customers

SET first\_name = 'Johnny', last\_name = 'Depp'

WHERE customer\_id = 1;



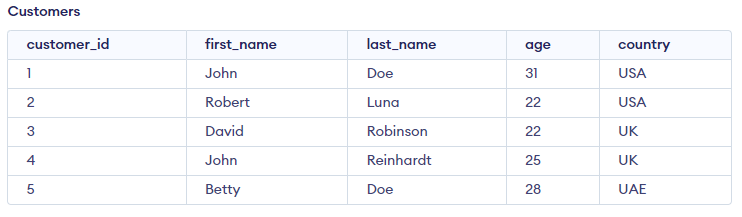


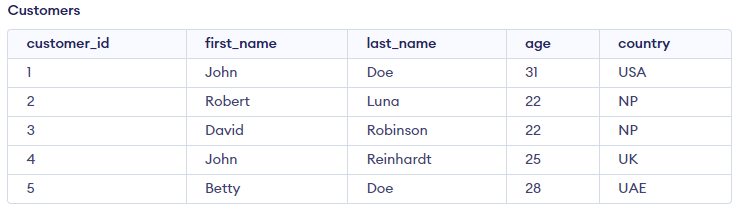
**3. Оновити декілька рядків, що задовольняють умові**

UPDATE Customers

SET country = 'NP'

WHERE age = 22;

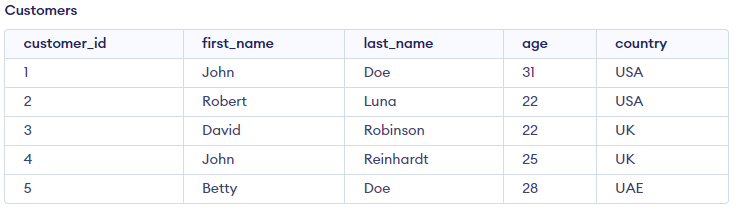


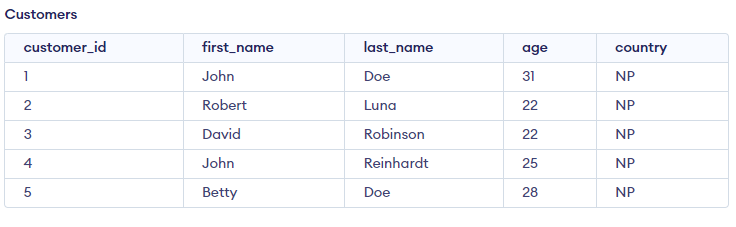


**4. Оновити всі рядки**

UPDATE Customers

SET country = 'NP';





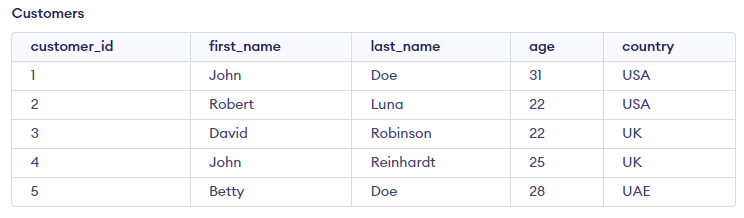
**DELETE**

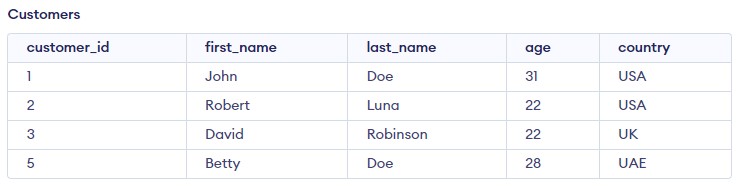
**DELETE** - використовується для видалення рядка(ів) з таблиці бази даних

**1. Видалити рядок з таблиці Customers, якщо його customer\_id дорівнює 4**

DELETE FROM Customers

WHERE customer\_id = 4;





**2. Видалити всі рядки з таблиці Customers**

DELETE FROM Customers;

**DDL (Data Definition Language)**

CREATE TABLE, ALTER TABLE, DROP TABLE

**DDL (Data Definition Language) –** оператори для управління тестовими таблицями

**CREATE TABLE**

**CREATE TABLE -** для створення таблиці бази даних

|  |  |  |
| --- | --- | --- |
| **Data Type (Тип даних)** | **Description (Опис)** | **Example (Приклад)** |
| **int** | може зберігати числа | 400, -300 |
| **varchar(x)** | може зберігати змінні символи з максимальною довжиною x | John Doe, United States of America |
| **text** | може зберігати тексти до **65535** символів | Це дуже довгий абзац, який може переходити через рядки. |

**1. Створити таблицю Companies з id, name, address, email, phone**

CREATE TABLE Companies (

id int,

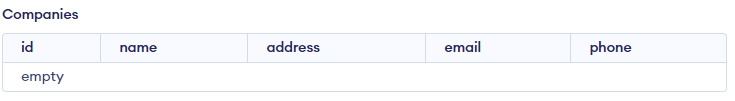
name varchar(50),

address text,

email varchar(50),

phone varchar(10)

);



**ALTER TABLE**

**ALTER TABLE** - використовується для зміни структури існуючої таблиці, наприклад, додавання, видалення, перейменування стовпців тощо

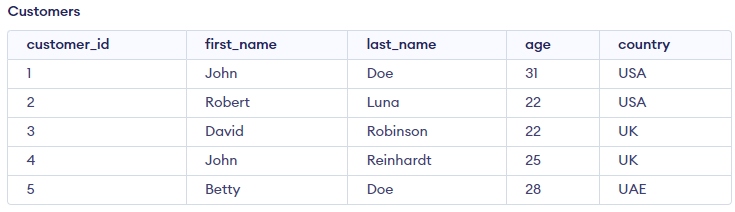
За допомогою команди ALTER TABLE можна виконати такі операції над таблицею:

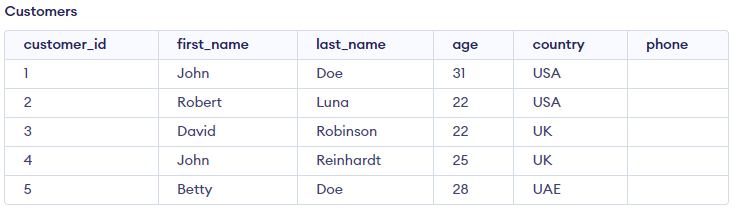
* **Add a column** - Додати стовпець
* **Rename a column** - Перейменування стовпця
* **Modify a column** - Змінити стовпець
* **Delete a column** - Видалити стовпець
* **Rename a table** - Перейменування таблиці

**1. Додати стовпець телефон до таблиці Customers**

ALTER TABLE Customers

ADD phone varchar(10);





**2. Додати до таблиці Customers стовпці phone та age**

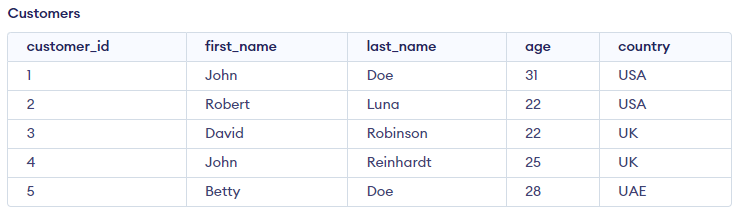
ALTER TABLE Customers

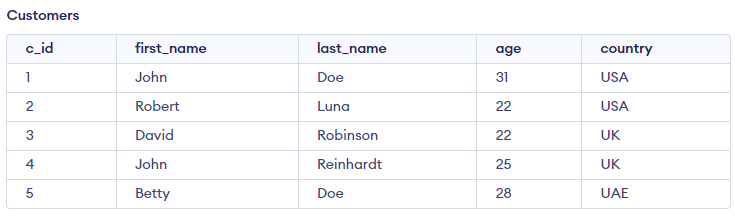
ADD phone varchar(10), age int;

**3. Перейменувати стовпець customer\_id на c\_id**

ALTER TABLE Customers

RENAME COLUMN customer\_id TO c\_id;

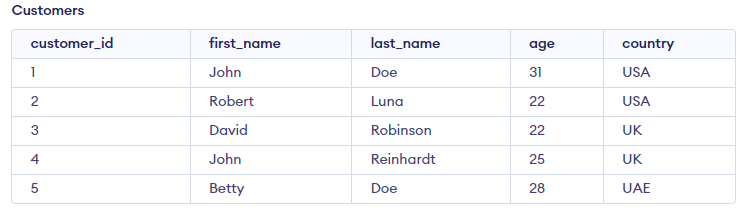


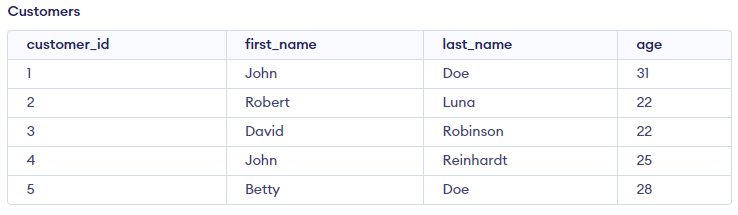


**4. Видалити стовпець** country **з таблиці Customers**

ALTER TABLE Customers

DROP COLUMN country;

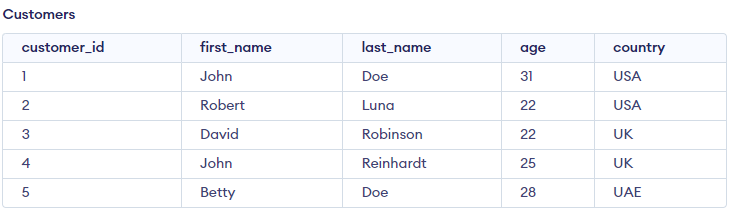


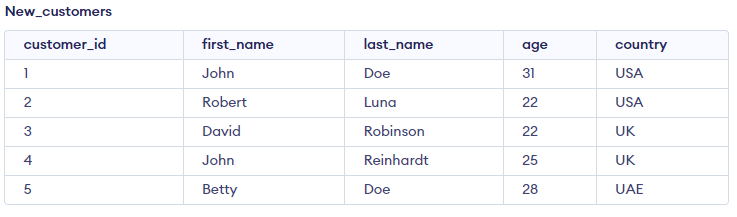


**5. Перейменувати таблицю Customers на New\_customers**

ALTER TABLE Customers

RENAME TO New\_customers;





**DROP TABLE**

**DROP TABLE** - використовується для видалення таблиць в нашій базі даних

**1. Видалити таблицю Companies**

DROP TABLE Companies;

**JOINS (Об'єднання)**

У мові SQL існує чотири основних типи об'єднань:

* **INNER JOIN** (ВНУТРІШНЄ об'єднання)
* **LEFT JOI**N (ЛІВЕ З'ЄДНАННЯ)
* **RIGHT JOIN** (ПРАВЕ З'ЄДНАННЯ)
* **FULL OUTER JO**IN (ПОВНЕ ЗОВНІШНЄ З'ЄДНАННЯ)

**SQL JOIN (основи)**

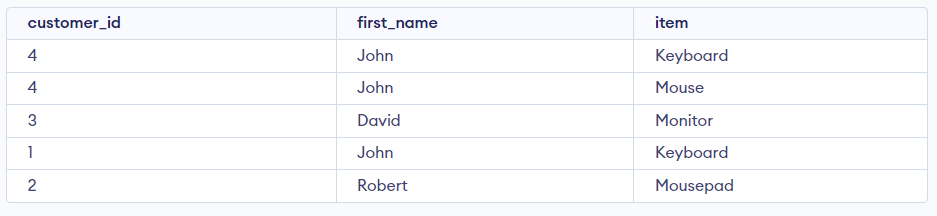
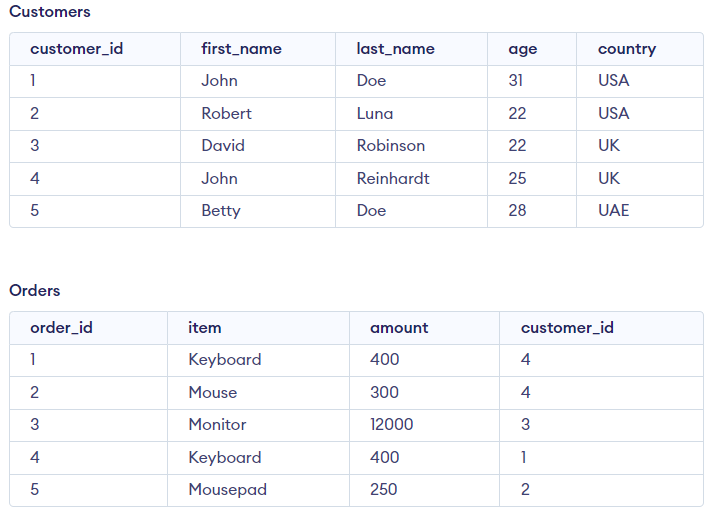
1. Об'єднати таблиці Customers та Orders на основі спільних значень стовпців customer\_id

SELECT Customers.customer\_id, Customers.first\_name, Orders.item

FROM Customers

JOIN Orders

ON Customers.customer\_id = Orders.customer\_id;



**INNER JOIN**

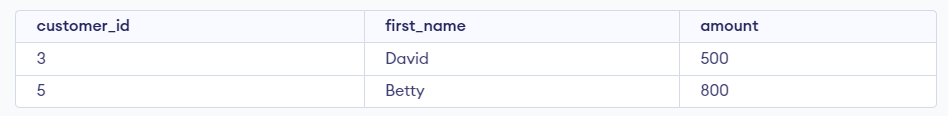
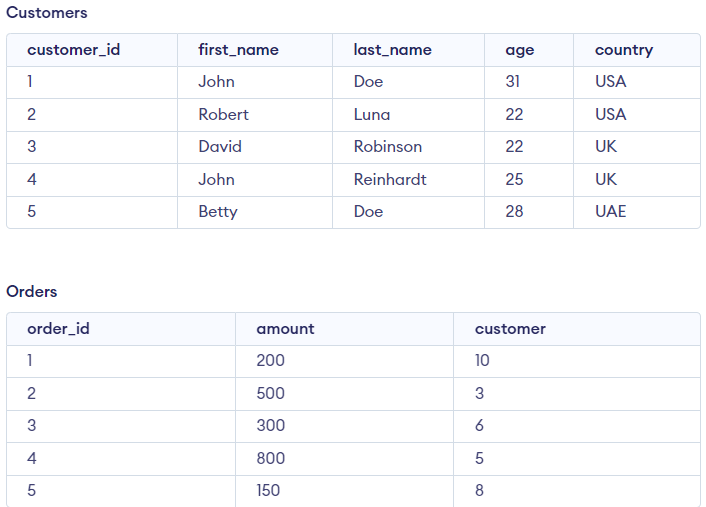
**1. Об'єднати таблиці Customers та Orders, вибрати стовпці customer\_id та first\_name з таблиці Customers, вибрати суму з таблиці Orders**

SELECT Customers.customer\_id, Customers.first\_name, Orders.amount

FROM Customers

INNER JOIN Orders

ON Customers.customer\_id = Orders.customer;



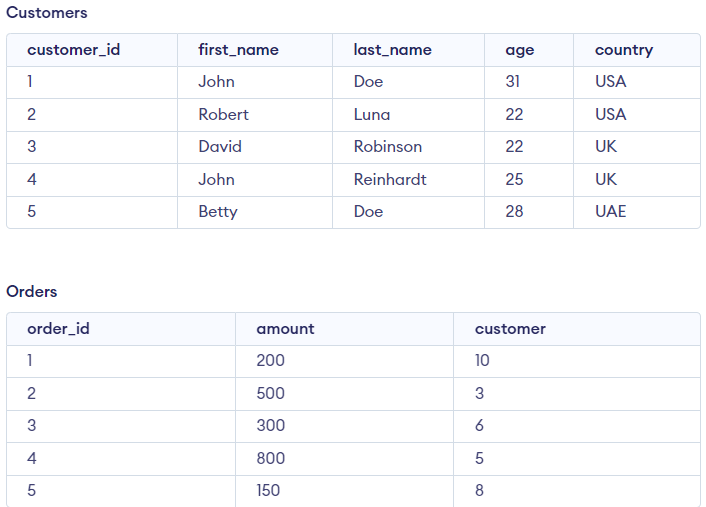
**2. Об'єднати таблиці Customers та Orders, коли ідентифікатор клієнта з таблиці Customers співпадає зі стовпчиком клієнта в таблиці Orders**

SELECT Customers.customer\_id, Customers.first\_name, Orders.amount

FROM Customers

JOIN Orders

ON Customers.customer\_id = Orders.customer;



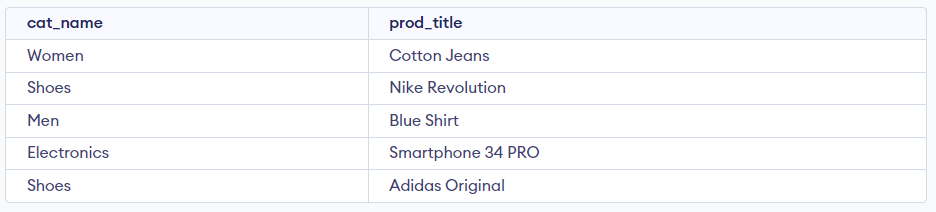
**3. Вибрати спільні рядки між таблицями Categories і Products на основі стовпця cat\_id, який присутній в обох таблицях**

SELECT Categories.cat\_name, Products.prod\_title

FROM Categories

INNER JOIN Products

ON Categories.cat\_id = Products.cat\_id;



**4. Об'єднати таблиці Customers та Orders та повернути рядки, якщо сума 500 або більше**

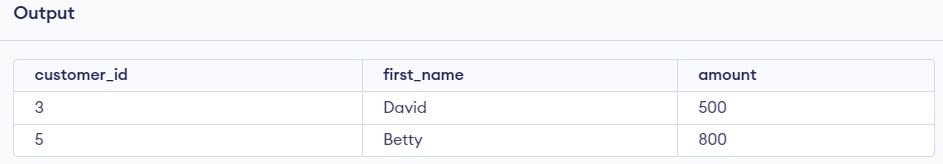
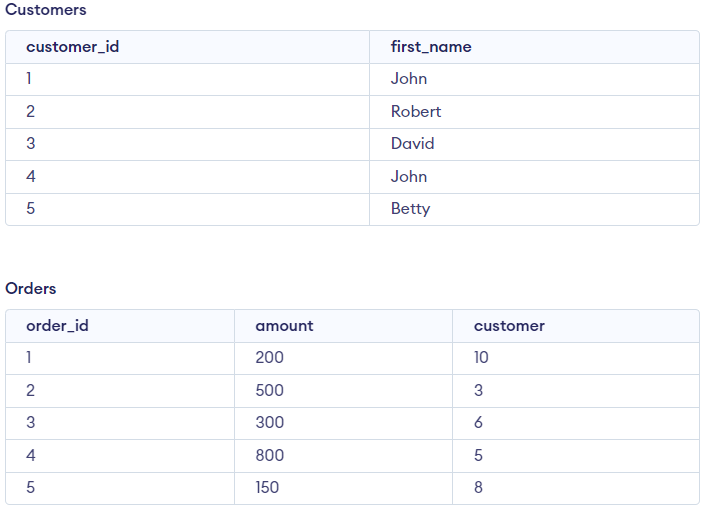
SELECT Customers.customer\_id, Customers.first\_name, Orders.amount

FROM Customers

INNER JOIN Orders

ON Customers.customer\_id = Orders.customer

WHERE Orders.amount >= 500;



**5. Приєднатися до трьох таблиць: Customers, Orders, Shippings**

SELECT C.customer\_id, C.first\_name, O.amount, S.status

FROM Customers AS C

INNER JOIN Orders AS O

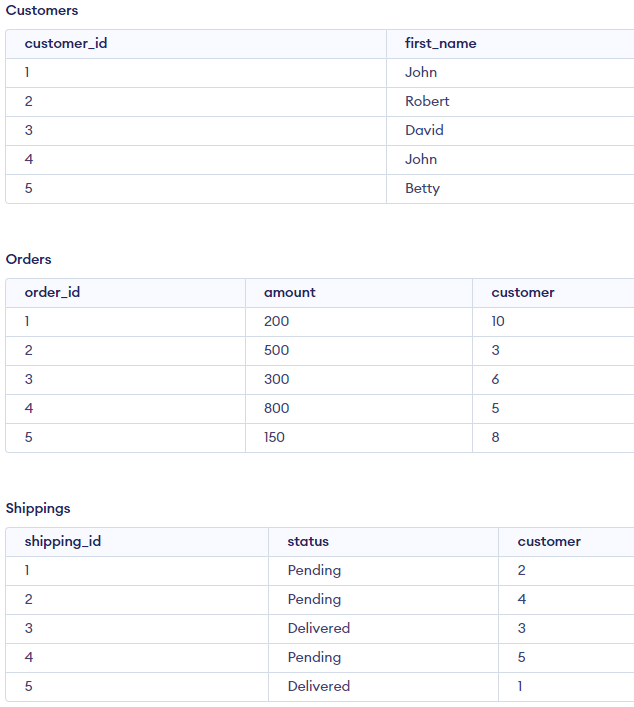
ON C.customer\_id = O.customer

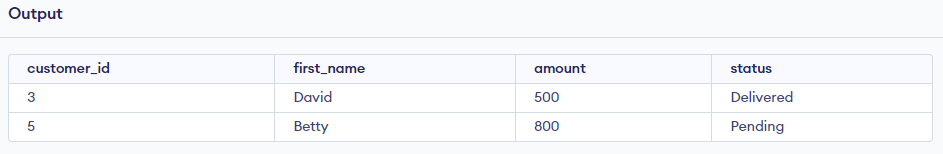
INNER JOIN Shippings AS S

ON C.customer\_id = S.customer;

Тут команда SQL:

* об'єднує таблиці Customers і Orders на основі customer\_id (з таблиці Customers) і customer (з таблиці Orders)
* і об'єднує таблиці Customers і Shippingss на основі customer\_id (з таблиці Customers) і customer (з таблиці Shippings)
* Команда повертає ті рядки, в яких значення стовпців збігаються за обома умовами об'єднання.





**LEFT JOIN**

**SQL LEFT JOIN** об'єднує дві таблиці на основі спільного стовпця. Він вибирає записи, які мають співпадаючі значення у цих стовпцях, та решту рядків з лівої таблиці

**1. Об'єднати таблиці Customers і Orders на основі їх спільних стовпців customer\_id (Customers - ліва таблиця, Orders - права таблиця)**

SELECT Customers.customer\_id, Customers.first\_name, Orders.item

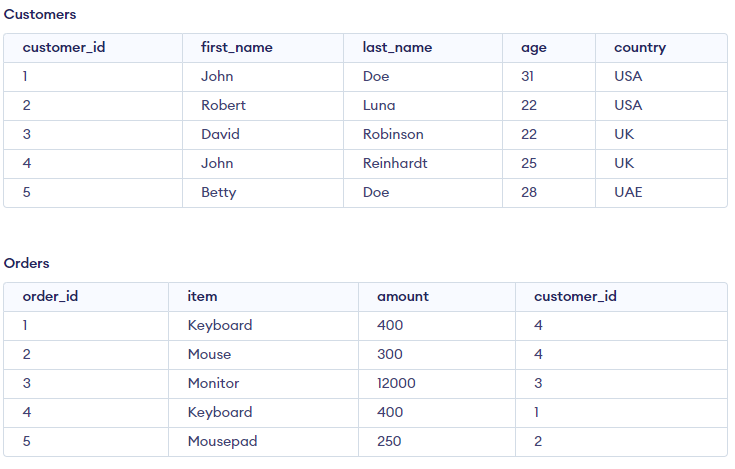
FROM Customers

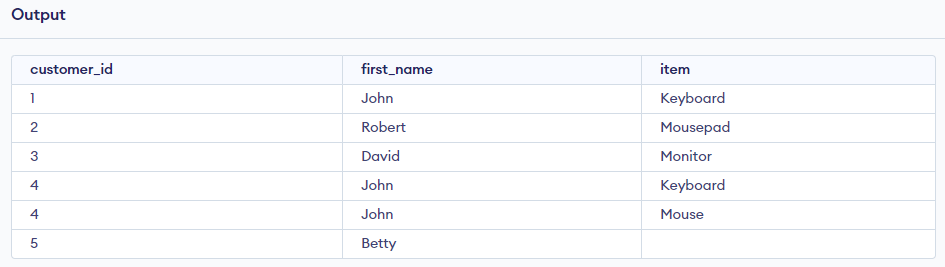
LEFT JOIN Orders

ON Customers.customer\_id = Orders.customer\_id;

Тут код зліва об'єднує таблиці Customers і Orders на основі customer\_id, який є спільним для обох таблиць. Результуючий набір містить:

* стовпці customer\_id та first\_name з таблиці Customers (включно з тими, чиє значення customer\_id відсутнє в таблиці Orders)
* стовпець item з таблиці Orders





**2. Об'єднати таблиці Customers та Orders**

SELECT Customers.customer\_id, Customers.first\_name, Orders.amount

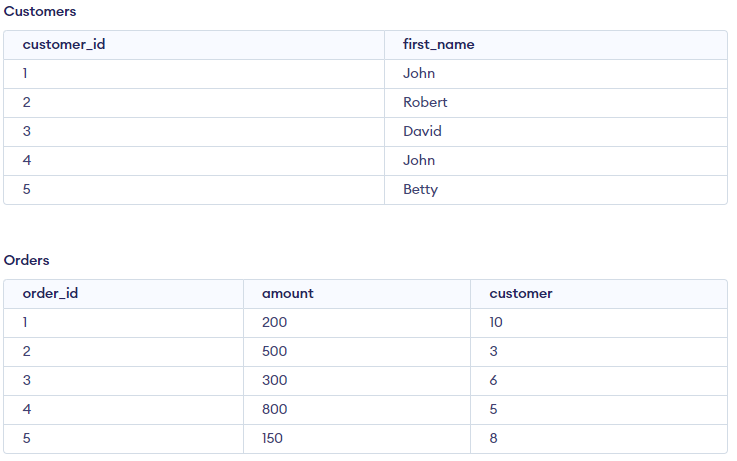
FROM Customers

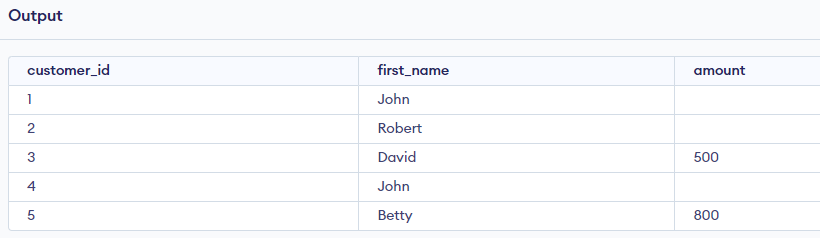
LEFT JOIN Orders

ON Customers.customer\_id = Orders.customer;

Тут команда SQL вибирає стовпці customer\_id та first\_name (з таблиці Customers) і стовпець amount (з таблиці Orders).

Результатом буде набір рядків, в яких є збіг між customer\_id (з таблиці Customers) і customer (з таблиці Orders), а також всі інші рядки з таблиці Customers.





**3. Об'єднати таблиці Customers і Orders і вибирати рядки, в яких сума більше або дорівнює 500**

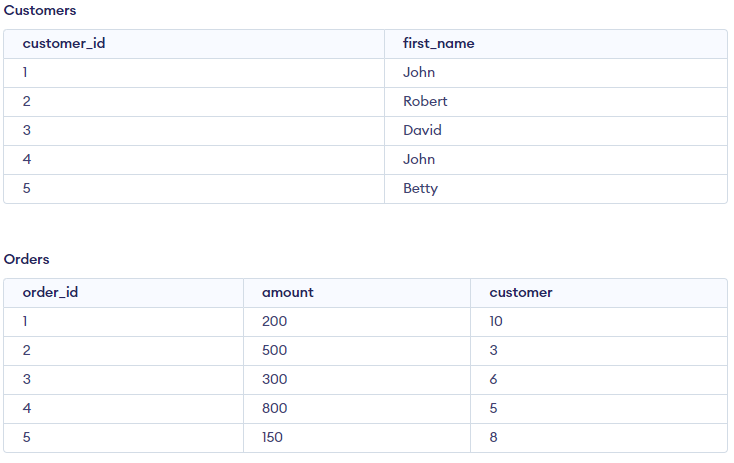
SELECT Customers.customer\_id, Customers.first\_name, Orders.amount

FROM Customers

LEFT JOIN Orders

ON Customers.customer\_id = Orders.customer

WHERE Orders.amount >= 500;





**SQL RIGHT JOIN**

**SQL RIGHT JOIN** з'єднує дві таблиці на основі спільного стовпця. Він вибирає записи, які мають відповідні значення в цих стовпцях, і решту рядків з правої таблиці

**1. Об'єднати таблиці Customers і Orders на основі їх спільних стовпців customer\_id**

**(Customers - ліва таблиця, Orders - права таблиця)**

SELECT Customers.customer\_id, Customers.first\_name, Orders.item

FROM Customers

RIGHT JOIN Orders

ON Customers.customer\_id = Orders.customer\_id;

Тут код з'єднує таблиці Customers і Orders на основі ідентифікатора customer\_id, який є спільним для обох таблиць. Результуючий набір містить:

* стовпці customer\_id та first\_name з таблиці Customers
* стовпець item з таблиці Orders (включно з тими, чиє значення customer\_id відсутнє в таблиці Customers)

**2. Об'єднати таблиці Customers і Orders і вибрати рядки, в яких сума більша або дорівнює 500**

SELECT Customers.customer\_id, Customers.first\_name, Orders.amount

FROM Customers

RIGHT JOIN Orders

ON Customers.customer\_id = Orders.customer

WHERE Orders.amount >= 500;

**FULL OUTER JOIN (повне об'єднання)**

**SQL FULL OUTER JOIN** об'єднує дві таблиці на основі спільного стовпця. Він вибирає записи, які мають співпадаючі значення в цих стовпцях, а також решту рядків з обох таблиць

**1. Виконати повне об'єднання таблиць Customers і Orders на основі їх спільних стовпців customer\_id (Customers - ліва таблиця, Orders - права таблиця)**

SELECT Customers.customer\_id, Customers.first\_name, Orders.item

FROM Customers

FULL OUTER JOIN Orders

ON Customers.customer\_id = Orders.customer\_id;

Тут SQL-запит виконує ПОВНЕ ОБ'ЄДНАННЯ двох таблиць, Customers і Orders. Це означає, що результуючий набір містить всі рядки з обох таблиць, включаючи ті, що не мають спільних значень customer\_id.

**3. Об'єднати дві таблиці Customers та Orders і вибрати рядки, в яких сума більше або дорівнює 500.**

SELECT Customers.customer\_id, Customers.first\_name, Orders.amount

FROM Customers

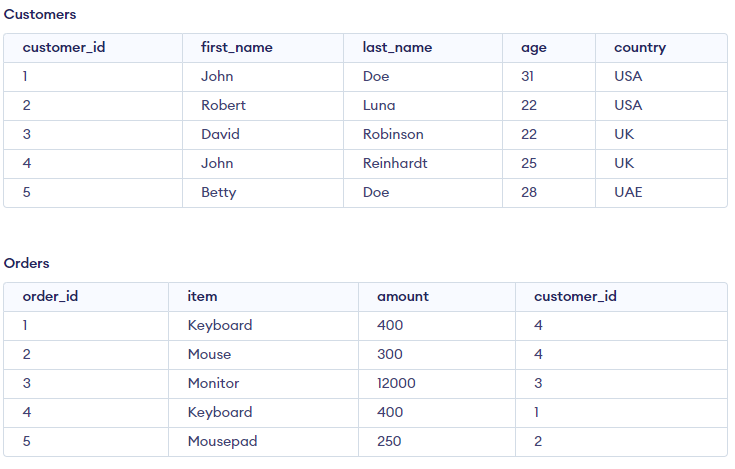
FULL OUTER JOIN Orders

ON Customers.customer\_id = Orders.customer

WHERE Orders.amount >= 500;

**DISTINCT**

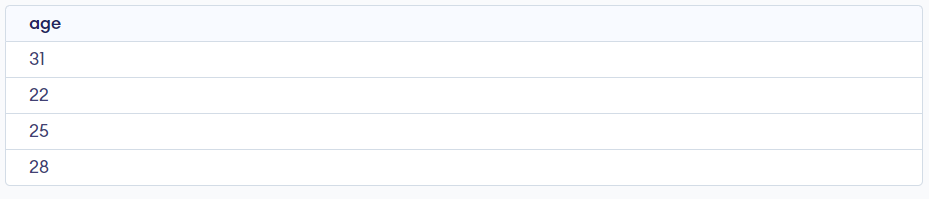
**SELECT DISTINCT** - отримує окремі значення з таблиці бази даних.



**1. Вибрати унікальний вік з таблиці Customers**

SELECT DISTINCT age

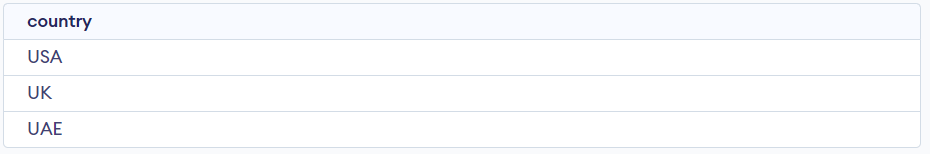
FROM Customers;



**2. Вибрати унікальні країни з таблиці Customers**

SELECT DISTINCT country

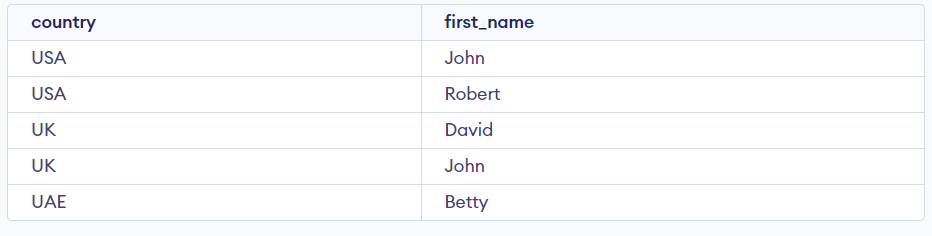
FROM Customers;



**2. Вибрати рядки, якщо ім'я та країна клієнта є унікальними**

SELECT DISTINCT country, first\_name

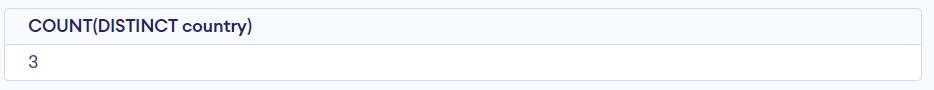
FROM Customers;



**3. Підрахувати унікальні країни, з яких походять клієнти**

SELECT COUNT(DISTINCT country)

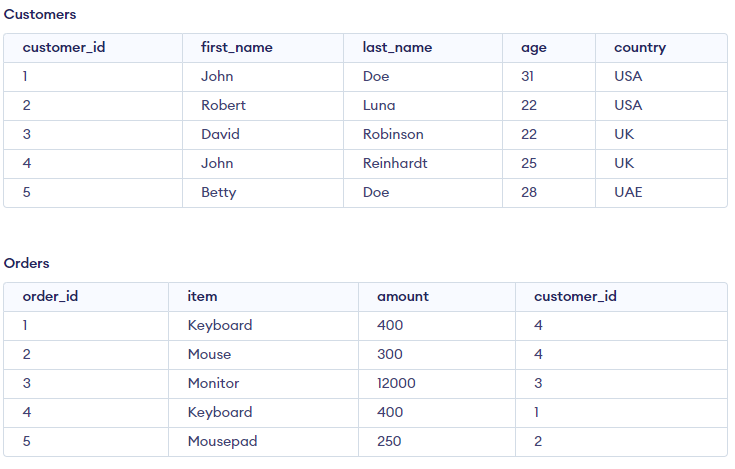
FROM Customers;



**IN та NOT IN**

**IN** - використовується з оператором WHERE для порівняння значень у списку

**NOT IN** - використовується для виключення рядків, які збігаються зі значеннями у списку. Він повертає всі рядки, крім виключених

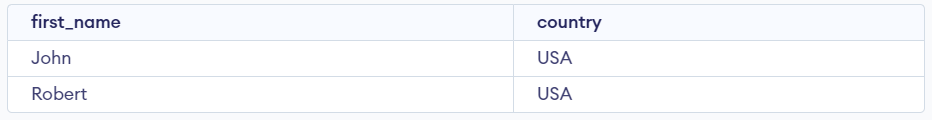


**1. Вибрати клієнтів з USA**

SELECT first\_name, country

FROM Customers

WHERE country IN ('USA');

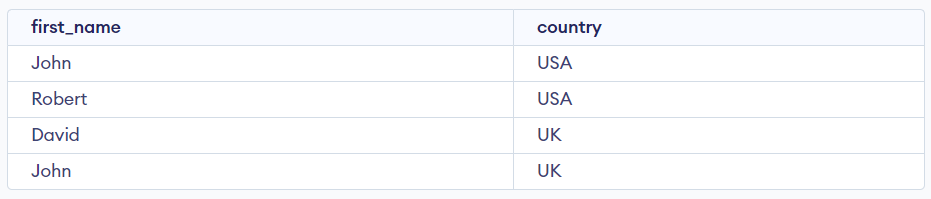


2. Виділити рядки, якщо країною є USA або UK

SELECT first\_name, country

FROM Customers

WHERE country IN ('USA', 'UK');

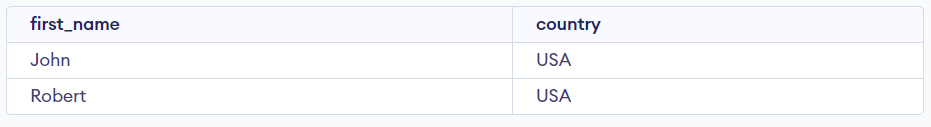


**2. Виділити рядки зі значенням 'USA' у стовпчику country**

SELECT first\_name, country

FROM Customers

WHERE 'USA' IN (country);

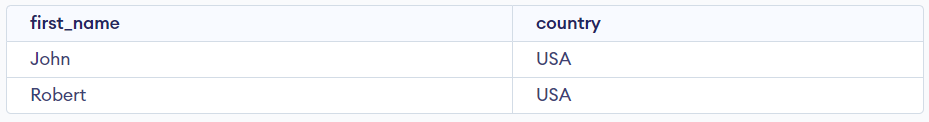


**3. Виберіть рядки, де країна не є UK або UAE**

SELECT first\_name, country

FROM Customers

WHERE country NOT IN ('UK', 'UAE');



**4. Вибрати тільки тих клієнтів, які оформили замовлення**

SELECT customer\_id, first\_name

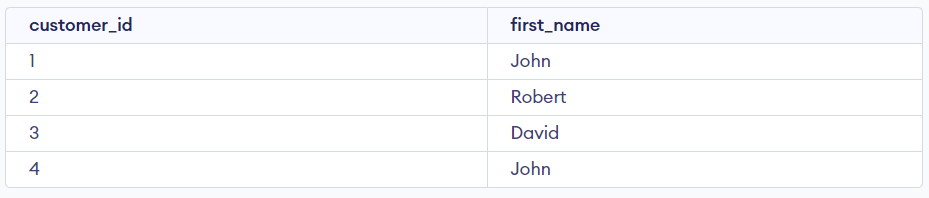
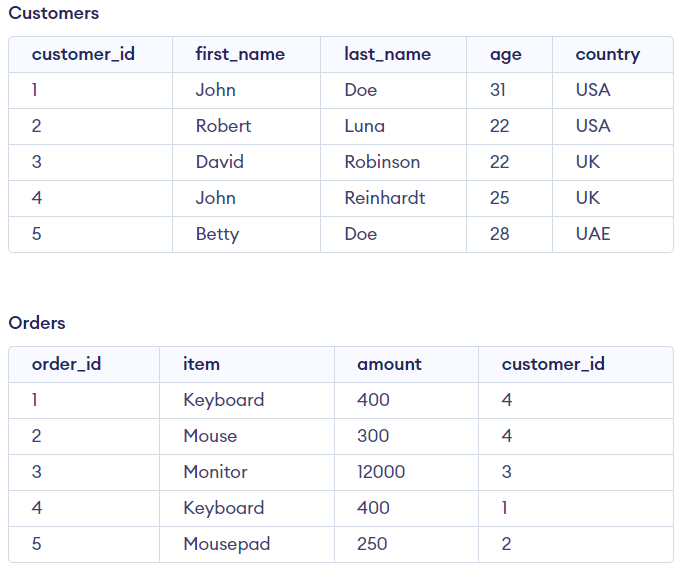
FROM Customers

WHERE customer\_id IN (

SELECT customer\_id

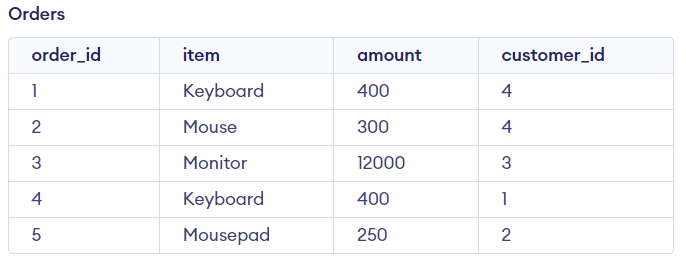
FROM Orders

);



**BETWEEN**

**BETWEEN** - оператор, який використовується для вказівки діапазону значень для стовпця (BETWEEN використовується з реченням WHERE)

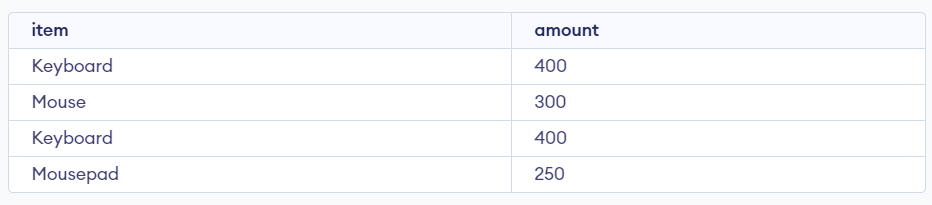


**1. Вибрати стовпці item і amount з таблиці Orders, в яких сума знаходиться в діапазоні від 200 до 600**

SELECT item, amount

FROM Orders

WHERE amount BETWEEN 200 AND 600;

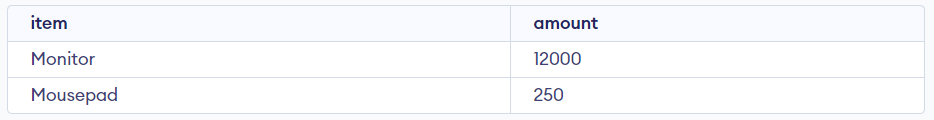


**2. Виключити рядки з сумою від 300 до 500**

SELECT item, amount

FROM Orders

WHERE amount NOT BETWEEN 300 AND 500;

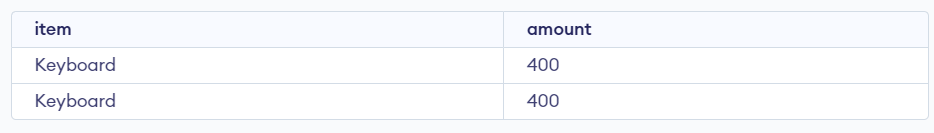


**3. Виділити рядки, в яких елементи починаються з літер від 'I' до 'L', виключити всі елементи, що починаються з 'L', за якими йдуть інші символи**

SELECT item, amount

FROM Orders

WHERE item BETWEEN 'I' AND 'L';



**ORDER BY Clause**

**ORDER BY** в SQL використовується для сортування набору результатів за зростанням або спаданням

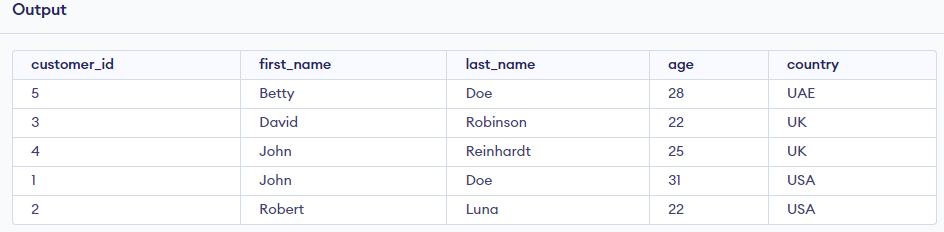
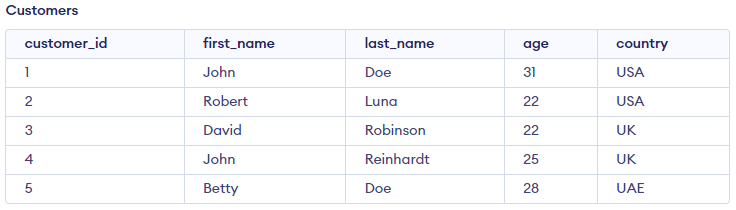
Ми можемо використовувати ключове слово **ASC** для явного сортування вибраних записів за зростанням

**1. Вибрати всі рядки з таблиці Customers, а потім відсортувати їх у порядку зростання за країнами**

SELECT \*

FROM Customers

ORDER BY country;



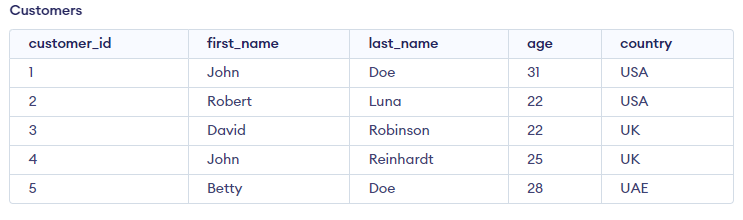
­­­­­­­­­­­­­­­­­

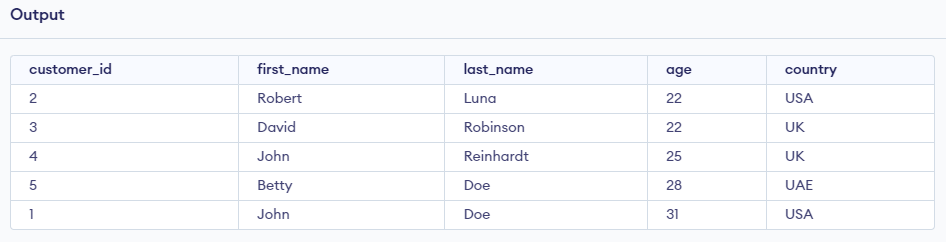
**2. Впорядкувати всі рядки таблиці Customers у порядку зростання за віком**

SELECT \*

FROM Customers

ORDER BY age ASC;



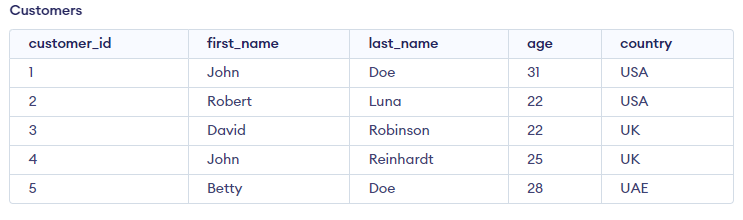


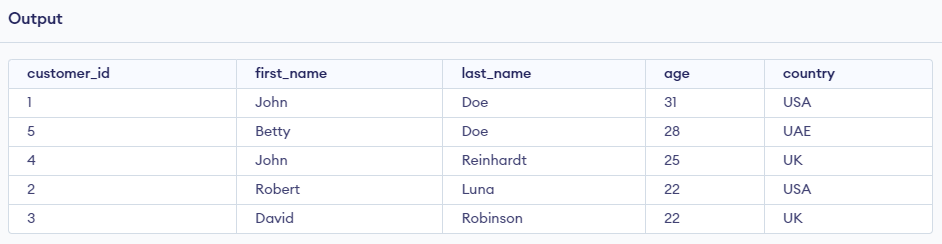
**3. Впорядкувати всі рядки таблиці Customers у порядку спадання за віком**

SELECT \*

FROM Customers

ORDER BY age DESC;



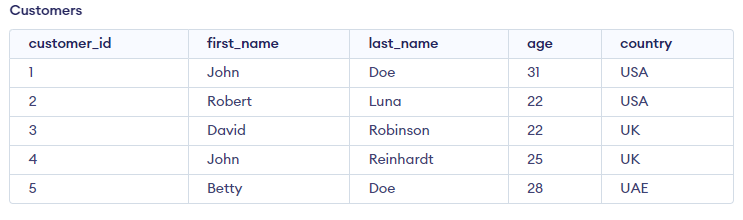


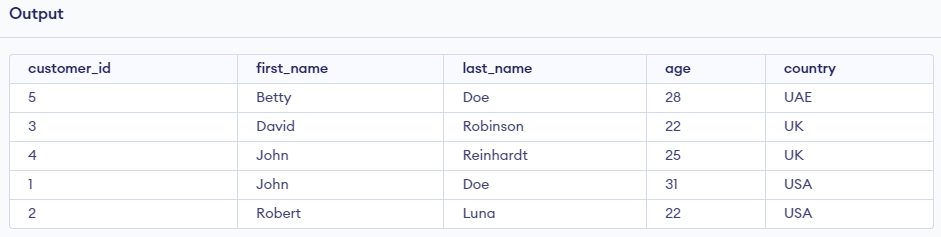
**4. Відсортувати всі рядки таблиці Customers спочатку за first\_name, а потім за віком**

SELECT \*

FROM Customers

ORDER BY first\_name, age;





**GROUP BY**

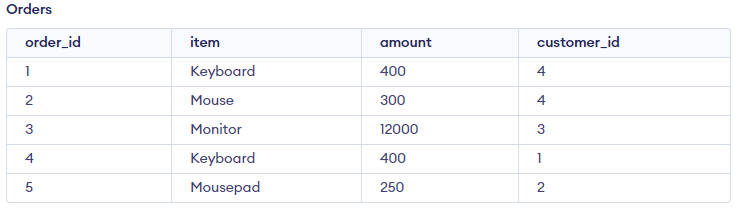
**GROUP BY** - використовується для групування рядків за одним або декількома стовпчиками

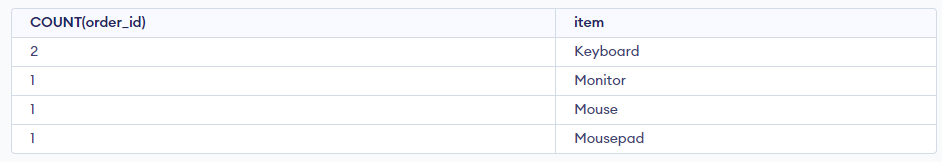
**1. Виберіть стовпець item і кількість ідентифікаторів замовлень з таблиці Orders, згрупувати їх за колонкою товару**

SELECT COUNT(order\_id), item

FROM Orders

GROUP BY item;



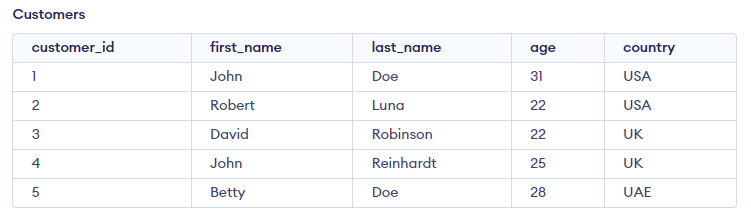


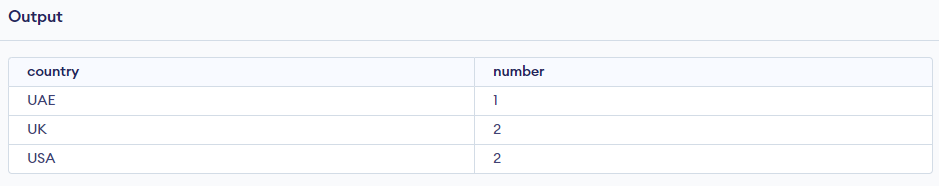
**2. Підрахувати кількість країн і згрупувати рядки за країнами**

SELECT country, COUNT(\*) AS number

FROM Customers

GROUP BY country;



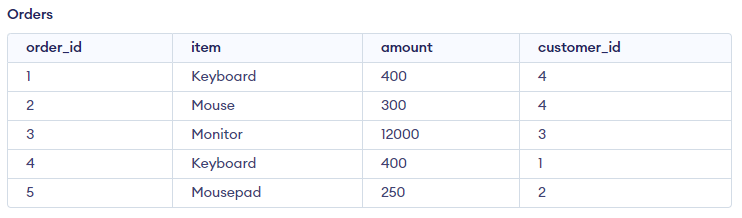


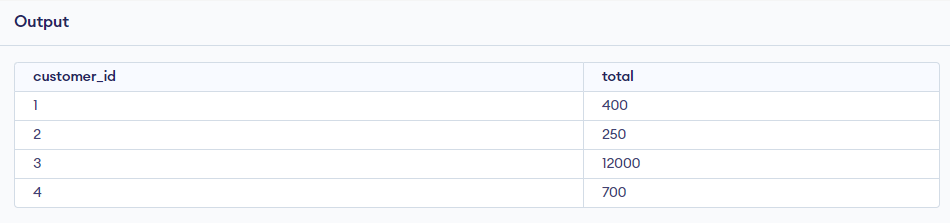
**3. Знайти загальну суму, витрачену кожним клієнтом, який зробив замовлення. Вибрати id\_клієнта та суму з Orders, згрупувати результат за customer\_id**

SELECT customer\_id, SUM(amount) AS total

FROM Orders

GROUP BY customer\_id;



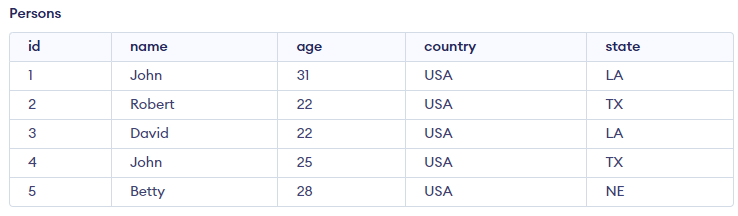


**4. Вибрати країну, штат і мінімальний вік з таблиці Persons, згрупувати за країною та штатом**

SELECT country, state, MIN(age) AS min\_age

FROM Persons

GROUP BY country, state;





**HAVING**

**HAVING** - використовується, якщо нам потрібно відфільтрувати набір результатів на основі агрегатних функцій, таких як MIN() і MAX(), SUM() і AVG(), а також COUNT()

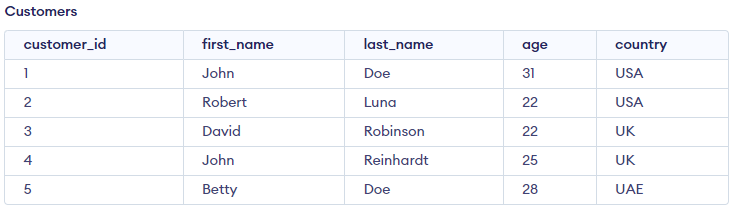
**1. Відбирати клієнтів з однаковим ім'ям на основі їхнього віку**

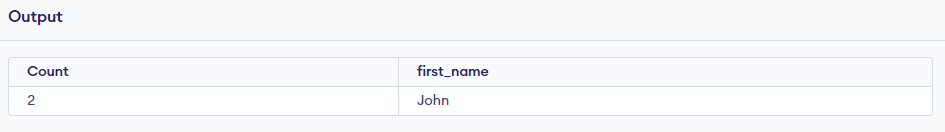
SELECT COUNT(age) AS Count, first\_name

FROM Customers

GROUP BY first\_name

HAVING COUNT(age) > 1;





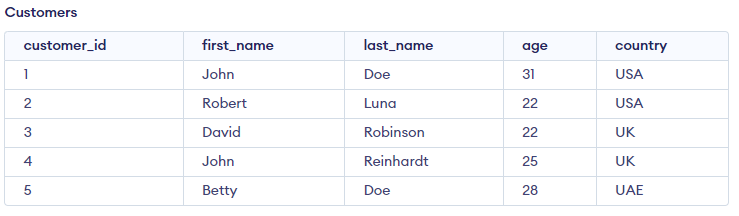
**2. Виберіть кількість ідентифікаторів клієнтів, більшу за одиницю, та відповідну країну**

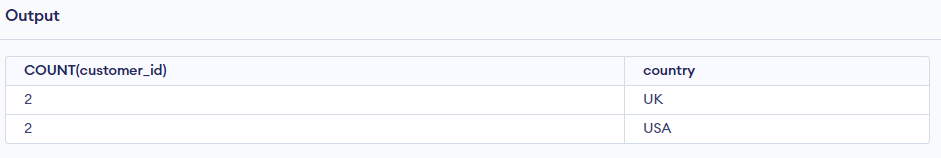
SELECT COUNT(customer\_id), country

FROM Customers

GROUP BY country

HAVING COUNT(customer\_id) > 1;





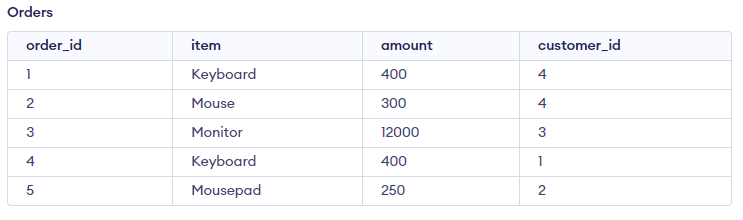
**2. Обчислити суму сум у таблиці Orders і отримати загальну вартість замовлення, меншу за 500 для кожного клієнта**

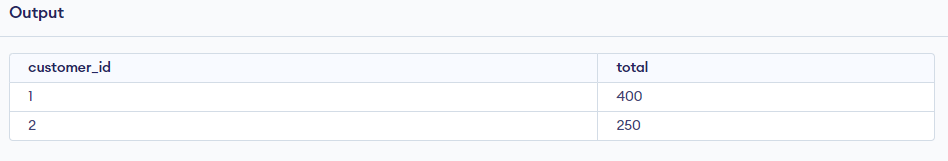
SELECT customer\_id, SUM(amount) AS total

FROM Orders

GROUP BY customer\_id

HAVING SUM(amount) < 500;





**Агрегатні функції**

Avg, Min, Max, Sum, Count

**SUM() і AVG()**

**SUM() і AVG()** - використовуються для обчислення загальних і середніх значень у числових стовпчиках

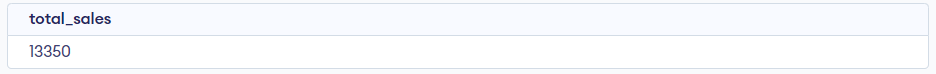
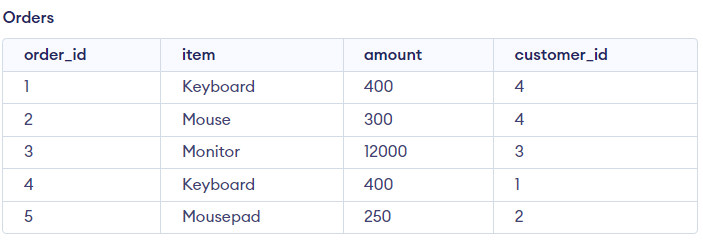
**SUM()** - використовується для обчислення кумулятивної суми числових значень у стовпці

**AVG()** - використовується для обчислення середнього арифметичного числових значень у стовпці

**1. Виберіть суму з таблиці Orders**

SELECT SUM(amount) AS total\_sales

FROM Orders;

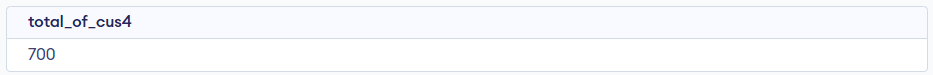


**2. Вибиремо суму суми id 4 з замовлень**

SELECT SUM(amount) AS total\_of\_cus4

FROM Orders

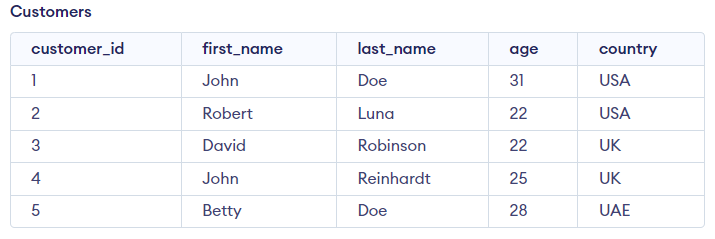
WHERE customer\_id = 4;

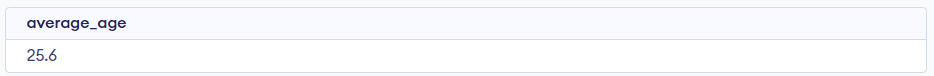


**3. Отримати середній вік клієнтів**

SELECT AVG(age) AS average\_age

FROM Customers;



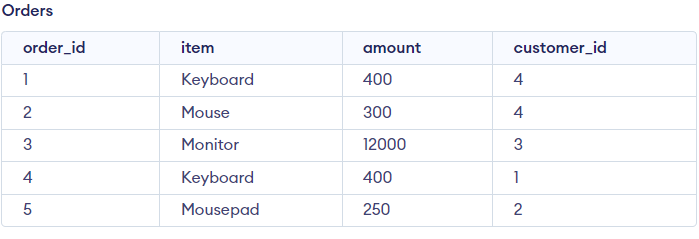


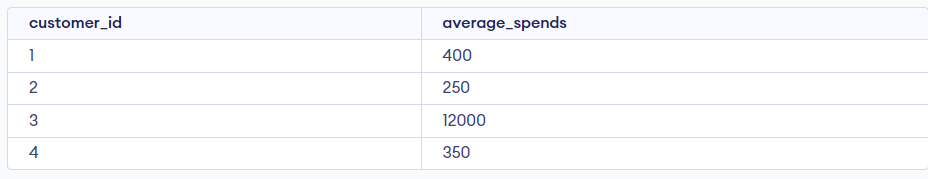
**3. Вибрати середню суму, витрачену кожним клієнтом, з таблиці Orders**

SELECT customer\_id, AVG(amount) AS average\_spends

FROM Orders

GROUP BY customer\_id;

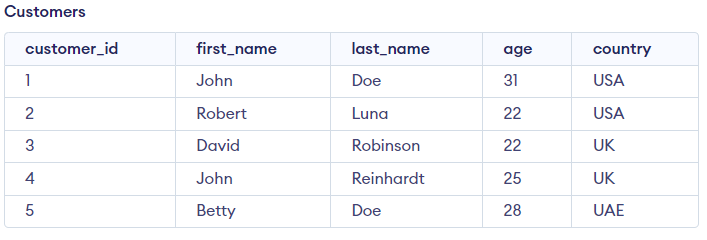




**MAX() та MIN()**

**MAX()** - повертає максимальне значення стовпця

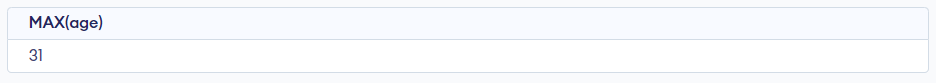
**MIN()** - повертає мінімальне значення стовпця



**1. Вибрати найбільше значення зі стовпчика age**

SELECT MAX(age)

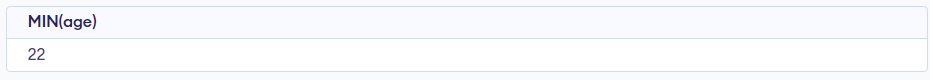
FROM Customers;



**2. Вибрати найменше значення зі стовпчика age**

SELECT MIN(age)

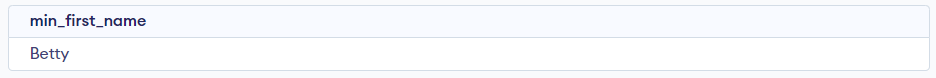
FROM Customers;



**3. Вибрати мінімальне значення first\_name з Customers**

SELECT MIN(first\_name) AS min\_first\_name

FROM Customers;



**4. Обрати всіх клієнтів з найменшим віком**

SELECT \*

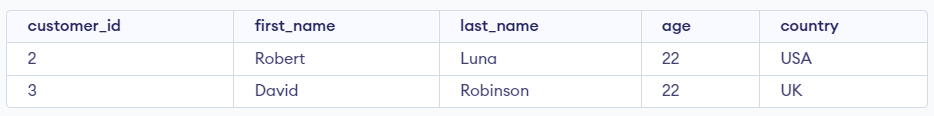
FROM Customers

WHERE age = (

SELECT MIN(age)

FROM Customers

);



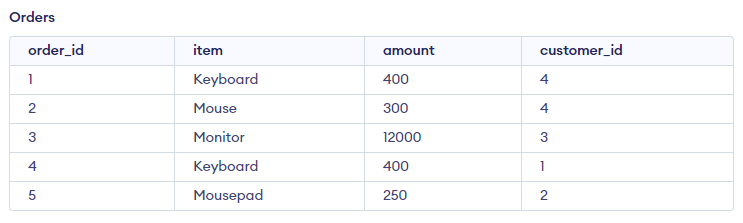
**COUNT()**

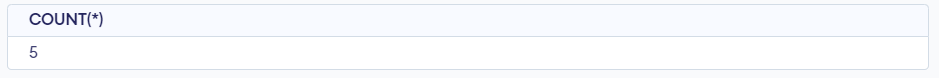
**COUNT()** - повертає кількість записів, повернених запитом

**1. Показати скільки всього створено рядків у таблиці Orders**

SELECT COUNT(\*)

FROM Orders;



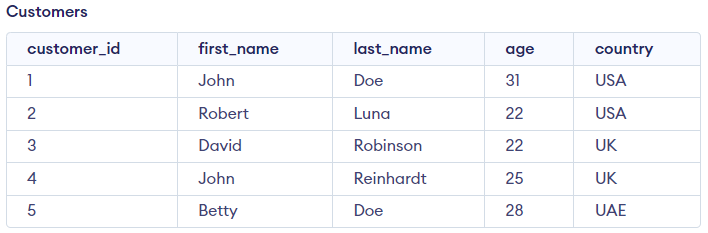


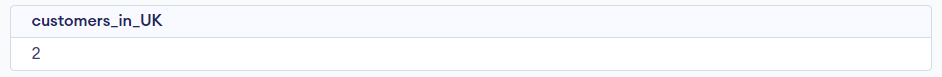
**2. Показати кількість клієнтів, які проживають у UK**

SELECT COUNT(country) AS customers\_in\_UK

FROM Customers

WHERE country = 'UK';

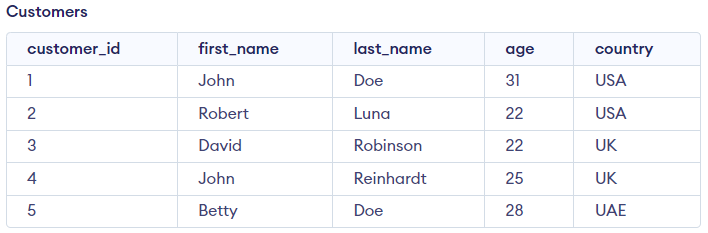


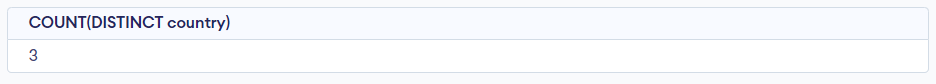


**3. Підрахувати унікальні країни в таблиці Customers**

SELECT COUNT(DISTINCT country)

FROM Customers;



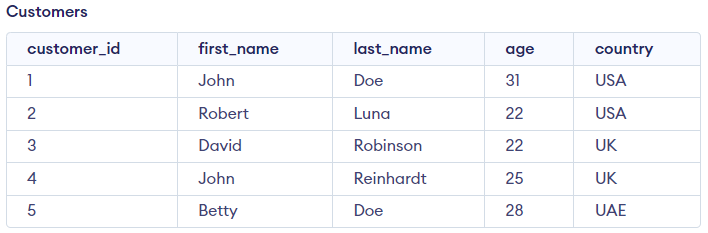


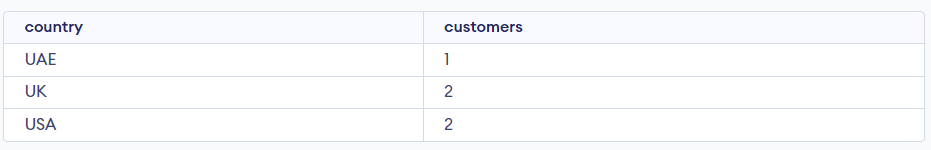
**4. Підрахувати кількість клієнтів у кожній країні**

SELECT country, COUNT(\*) AS customers

FROM Customers

GROUP BY country;





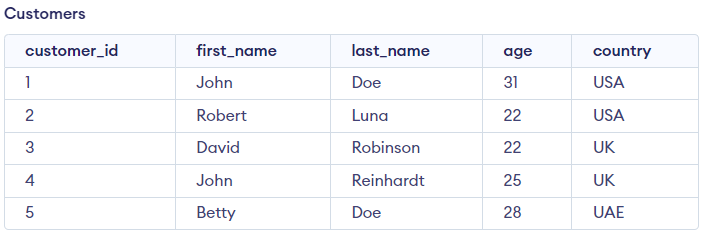
**5. Підрахувати кількість рядків за країнами і повернути результат, якщо він більший за одиницю**

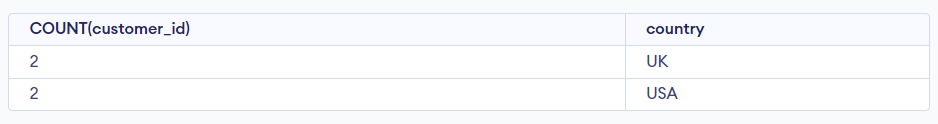
SELECT COUNT(customer\_id), country

FROM Customers

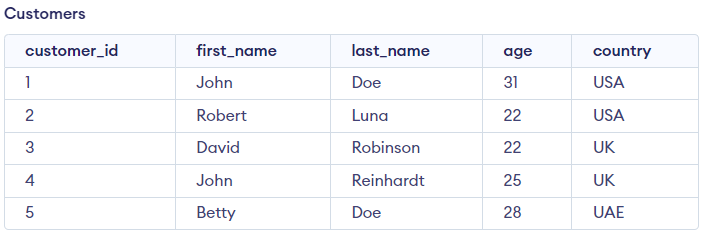
GROUP BY country

HAVING COUNT(customer\_id) > 1;





**=, >, AND, OR, NOT**

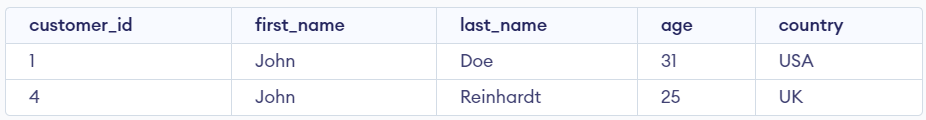


**1. *Дорівнює (=).* Вибрати всі стовпці з таблиці Customers з іменем 'John'**

SELECT \*

FROM Customers

WHERE first\_name = 'John';

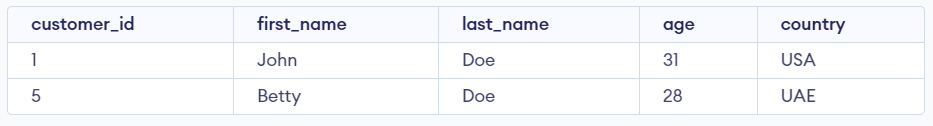


**2. *Більше, ніж (>).* Вибрати всі стовпці з таблиці Customers, вік яких більший за 25 років**

SELECT \*

FROM Customers

WHERE age > 25;

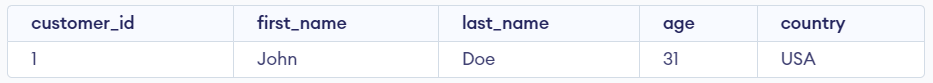


**3. *Оператор AND (І).* Вибрати всі стовпці з таблиці Customers з прізвищем 'Doe' та країною 'USA'**

SELECT \*

FROM Customers

WHERE last\_name = 'Doe' AND country = 'USA';

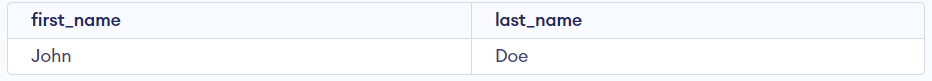


**4. Вибрати first\_name, last\_name всіх клієнтів, які живуть у 'США' і мають прізвище '**Doe**'**

SELECT first\_name, last\_name

FROM Customers

WHERE country = 'USA' AND last\_name = 'Doe';

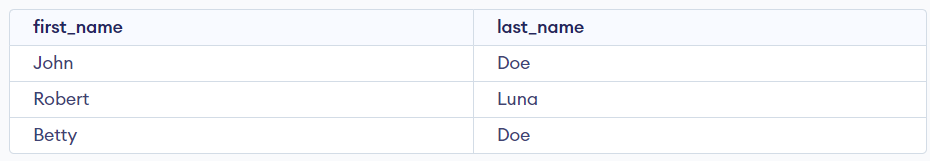


**5. *Оператор OR*. Виберіть first\_name, last\_name клієнтів, які або живуть у USA, або мають прізвище 'Doe'**

SELECT first\_name, last\_name

FROM Customers

WHERE country = 'USA' OR last\_name = 'Doe';

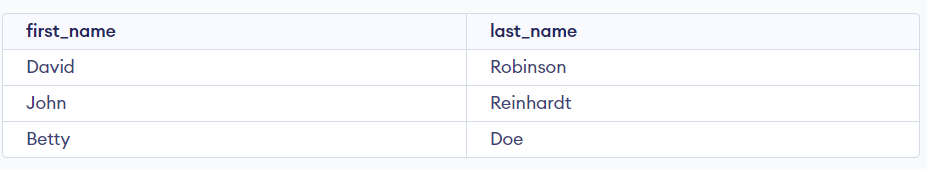


**6. *Оператор SQL NOT*. Вибрати клієнтів, які не живуть у USA**

SELECT first\_name, last\_name

FROM Customers

WHERE NOT country = 'USA';

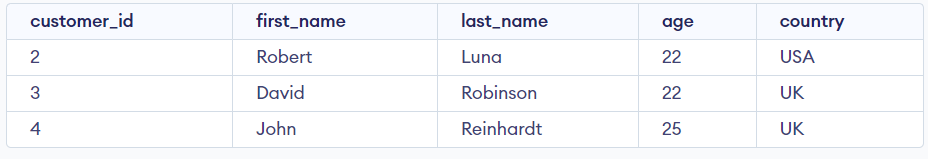


**7. Вибирати клієнтів, які проживають у USA або UK і вік яких не перевищує 26 років**

SELECT \*

FROM Customers

WHERE (country = 'USA' OR country = 'UK') AND age < 26;

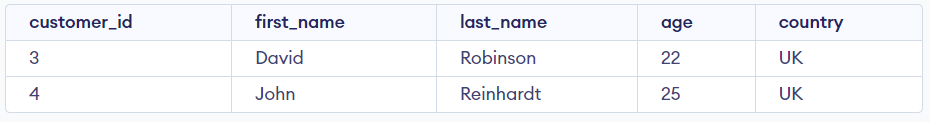


8. Вибрати клієнтів не з USA, прізвище яких не " Doe ".

SELECT \*

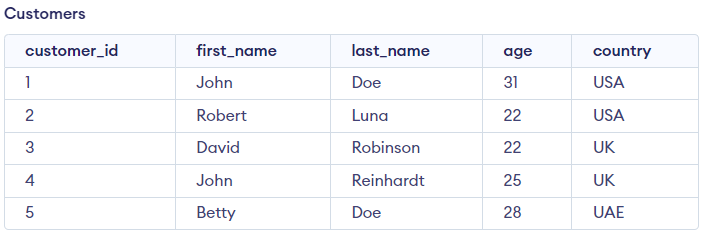
FROM customers

WHERE NOT country = 'USA' AND NOT last\_name = 'Doe';



**Alias - AS**

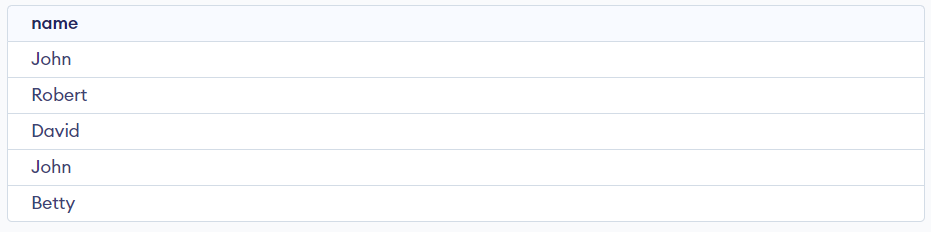
Ключове слово **AS** використовується для надання стовпчикам або таблицям тимчасової назви, яка може бути використана для ідентифікації цього стовпчика або таблиці пізніше



**1. Вибрати стовпець first\_name з таблиці Customers, змінити на name у результуючому наборі**

SELECT first\_name AS name

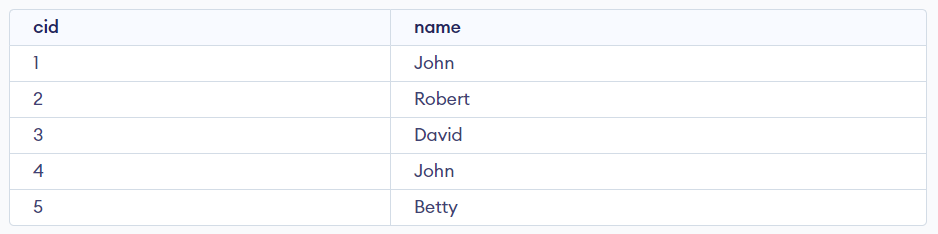
FROM Customers;



**2. Використовувати псевдоніми з більш ніж одним стовпчиком**

SELECT customer\_id AS cid, first\_name AS name

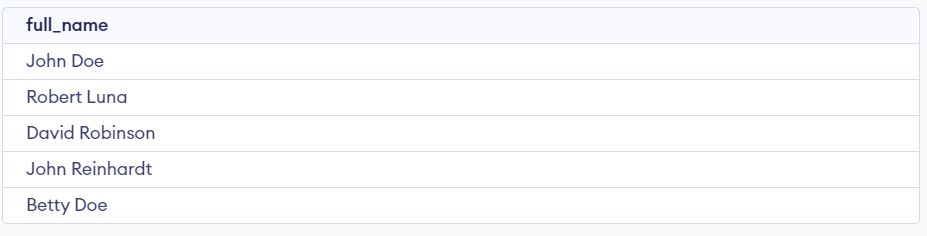
FROM Customers;



**3. Об'єднати first\_name, порожній пробіл та last\_name та у єдиний стовпець з іменем full\_name у результуючому наборі**

SELECT CONTACT(first\_name, ` `, last\_name) AS full\_name

FROM Customers;



**LIKE та NOT LIKE**

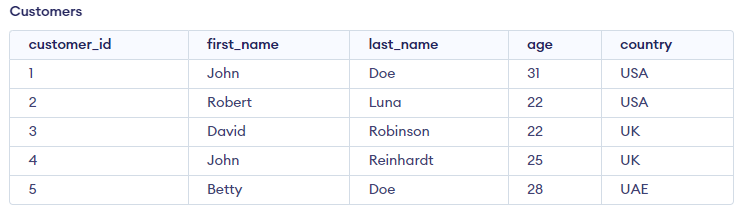
Примітка: Хоча у цьому прикладі оператор **LIKE** поводиться подібно до оператора **=**, це не одне й те саме. Оператор **=** використовується для перевірки рівності, тоді як оператор LIKE використовується лише для порівняння рядкових шаблонів.

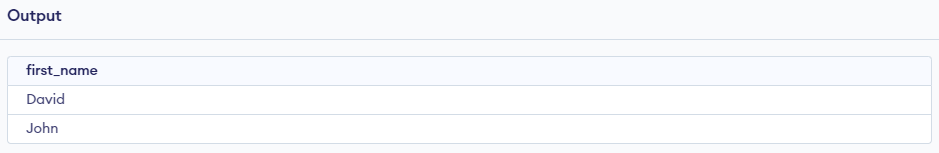
**1. Обирати клієнтів, які проживають у UK**

SELECT \*

FROM Customers

WHERE country LIKE 'UK';



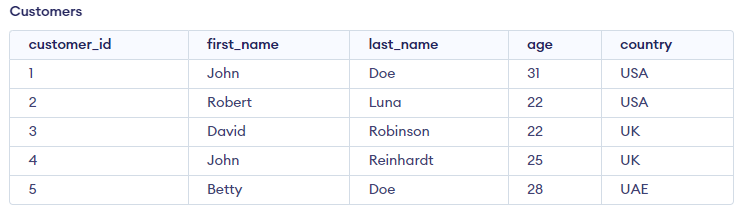


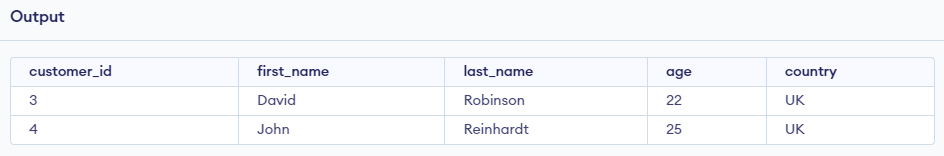
**2. Вибрати клієнтів, чиє ім'я починається з R, за яким слідує нуль або більше символів**

SELECT \*

FROM Customers

WHERE last\_name LIKE 'R%';



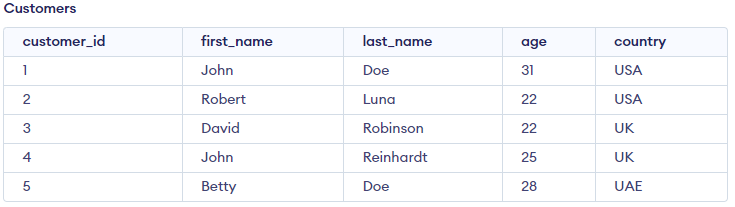


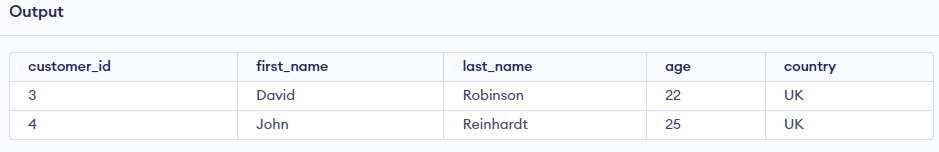
**3. Вибрати клієнтів, назви країн яких починаються з U, за яким слідує один символ**

SELECT \*

FROM Customers

WHERE country LIKE 'U\_';



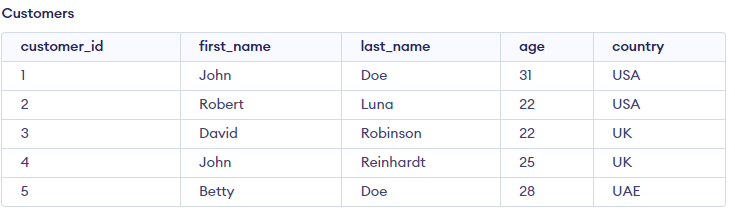


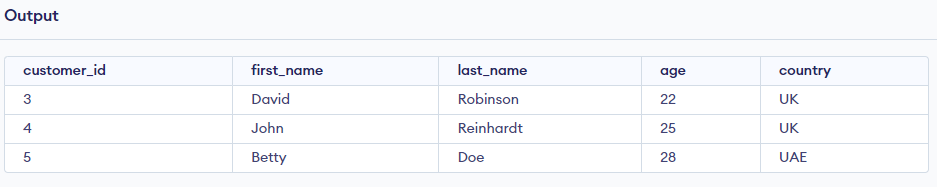
**4. Обрати клієнтів, які не живуть у США**

SELECT \*

FROM Customers

WHERE country NOT LIKE 'USA';



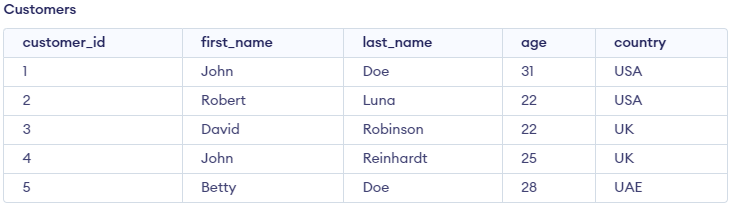


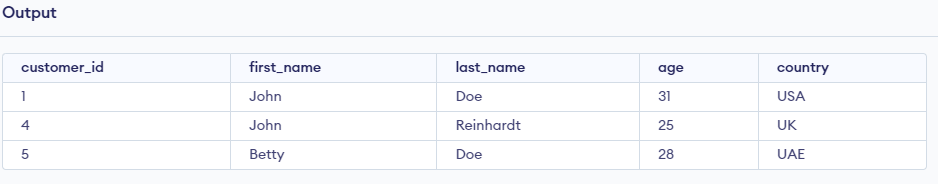
**5. Вибрати клієнтів, чиє прізвище починається на R і закінчується на t, або клієнтів, чиє прізвище закінчується на e**

SELECT \*

FROM Customers

WHERE last\_name LIKE 'R%t' OR last\_name LIKE '%e';





**6. Вибрати клієнтів, країна яких починається з UK або UA і далі йде довільна кількість символів**

SELECT \*

FROM Customers

WHERE country LIKE 'U[KA]%';

**7. Вибрати рядки, в яких прізвища клієнтів не починаються на D або R**

SELECT \*

FROM Customers

WHERE last\_name LIKE '[!DR]%';

**Wildcard** (символи)

Wildcard (спецсимволи) в різних базах даних:

* **SQL Server:**

% - нуль або більше символів

\_ - один символ

[] - один символ у дужках

^ - будь-який символ поза дужками

- - один символ в межах діапазону

* **PostgreSQL and MySQL:**

% - нуль або більше символів

\_ - один символ

* **Oracle:**

% - нуль або більше символів

\_ - один символ

[] - один символ у дужках

{} - екранований символ