Аналіз даних

Мова структурованих запитів

Анонс

Сьогодні майже кожному доводиться працювати з даними в тій чи іншій формі. Зазвичай це відбувається за допомогою електронних таблиць або баз даних, але якщо ви трохи навчитеся SQL, то зможете швидше знаходити відповіді на питання.

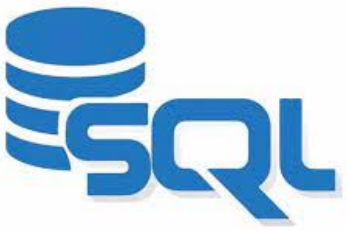
SQL — це спосіб швидко та легко працювати з даними.

Причини вивчити SQL:

- SQL і реляційні бази даних всюди;
- SQL легко вивчити;
- SQL легко працює з великими даними.

Мова структурованих запитів (SQL)

SQL — це мова програмування, яка використовується майже всіма реляційними базами даних для запитів, обробки та визначення даних, а також для забезпечення контролю доступу.



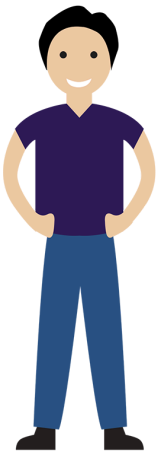
- SQL є мовою структурованих запитів.
- SQL дає змогу отримувати доступ до баз даних і маніпулювати ними.

SQL стала стандартом Американського національного інституту стандартів (ANSI) у 1986 році та Міжнародної організації зі стандартизації (ISO) у 1987 році.
SQL вперше було розроблено в IBM у 1970-х роках за участю Oracle як основного учасника, що призвело до впровадження стандарту SQL ANSI. SQL стимулювала багато розширень від таких компаній, як IBM, Oracle і Microsoft. Хоча SQL все ще широко використовується сьогодні, починають з'являтися інші мови запитів.

SQL може:

- створювати нові таблиці в базі даних;
- виконувати запити до бази даних;
- отримувати дані з бази даних;
- додавати записи в базу даних;
- оновлювати записи в базі даних;
- видаляти записи з бази даних;
- створювати нові бази даних;
- встановлювати дозволи для даних.

Історія однієї студії Дизайну



Герой нашої історії починав свою кар'єру в IT з замовлень на фрилансі, створював сайти й трохи займався дизайном. З часом замовлень ставало все більше, а вести декілька великих проєктів стало складно.
Він створив Студію Дизайну й почав набирати собі команду розробників. Крім систем керування проєктами, основну інформацію про співробітників вирішили зберігати в базі даних.
Завдання такої бази даних — вести облік людей у проєкті, контролювати виплати згідно з дедлайнами.

Завдання 1

Усі завдання цієї теми пов’язані між собою. Створіть теку на Google Диску для зберігання результатів.

Перш ніж заносити дані в базу, ми маємо визначитися, які нам потрібні поля та якого вони типу.

- phone — дані зі Списку контактів

Складіть короткий опис полів таблиці, вказавши тип даних та призначення.

Наприклад: fam string — прізвище.

Таблиця contact

```
Table "phone" {
  "code" int
  "num" int
  "fam" string
  "name" string
}
```

Синтаксис SQL

Таблиці бази даних

База даних найчастіше містить одну або декілька таблиць. Кожна таблиця ідентифікується назвою (наприклад, «Клієнти» або «Замовлення»). Таблиці містять записи (рядки) з даними.

Інструкції SQL

Більшість дій, які потрібно виконати з базою даних, виконуються за допомогою операторів SQL.

Розглянемо 4 інструкції, які допоможуть нам створити базу даних.

- CREATE DATABASE — створює нову базу даних.

  Приклад:

  CREATE DATABASE addrbook;

- CREATE TABLE — створює нову таблицю.

  Приклад:

  CREATE TABLE phone(num int, fam string);

- INSERT INTO — вносить нові дані в базу даних.

  Приклад:

  INSERT INTO phone values(10, 'Петренко');

- SELECT — витягує дані з бази даних.

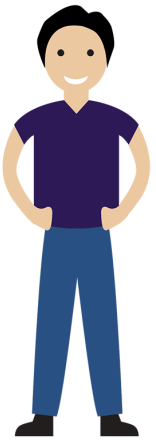
  Приклад:

  SELECT *
  FROM phone; (зірочка означає, що нам потрібні всі дані)

Тест 1

Максим, головний герой нашої історії, почав шукати талановитих розробників, дизайнерів та менеджерів проєктів, щоб об’єднати їх у спільному проєкті.

Під час формування команди Максим разом зі своїми колегами вирішив, що буде корисно зберігати основну інформацію про співробітників в базі даних. Ця база даних допоможе відстежувати учасників проєктів, контролювати виплати згідно з дедлайнами та взагалі спростить процес керування командою.



CREATE DATABASE — створює нову базу даних.

Для зберігання інформації про співробітників Максим вирішує створити таблицю "employees" у базі даних "design_studio".

```
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    position VARCHAR(100),
    salary DECIMAL(10, 2),
    hire_date DATE
);
```

За допомогою команди "INSERT INTO" Максим починає заповнювати таблицю інформацією про співробітників.

```
INSERT INTO employees VALUES (1, 'Максим', 'Директор', 6000.00, '2022-03-15');
INSERT INTO employees VALUES (2, 'Олена', 'Дизайнер', 4000.00, '2022-05-20');
INSERT INTO employees VALUES (3, 'Андрій', 'Розробник', 4500.00, '2022-04-10');
```

Зараз, завдяки запитам "SELECT", Максим може легко переглядати список співробітників та їхні дані. Це допомагає йому здійснювати контроль за виконанням проєктів, вчасними виплатами та забезпечує ефективне управління командою в студії дизайну.

.

Створення бази даних

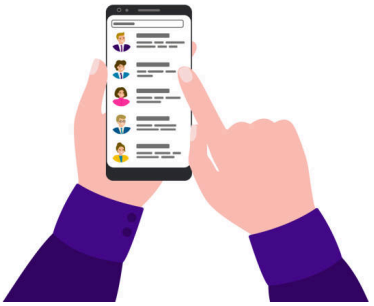
Можемо використовувати різні онлайн-сервіси.
Наприклад:

- https://www.jdoodle.com/;
- repl.it.

1. Почнемо створення з нової таблиці, визначаємо 2 поля: номер і прізвище:

```
CREATE TABLE phone(num int, fam string);
```

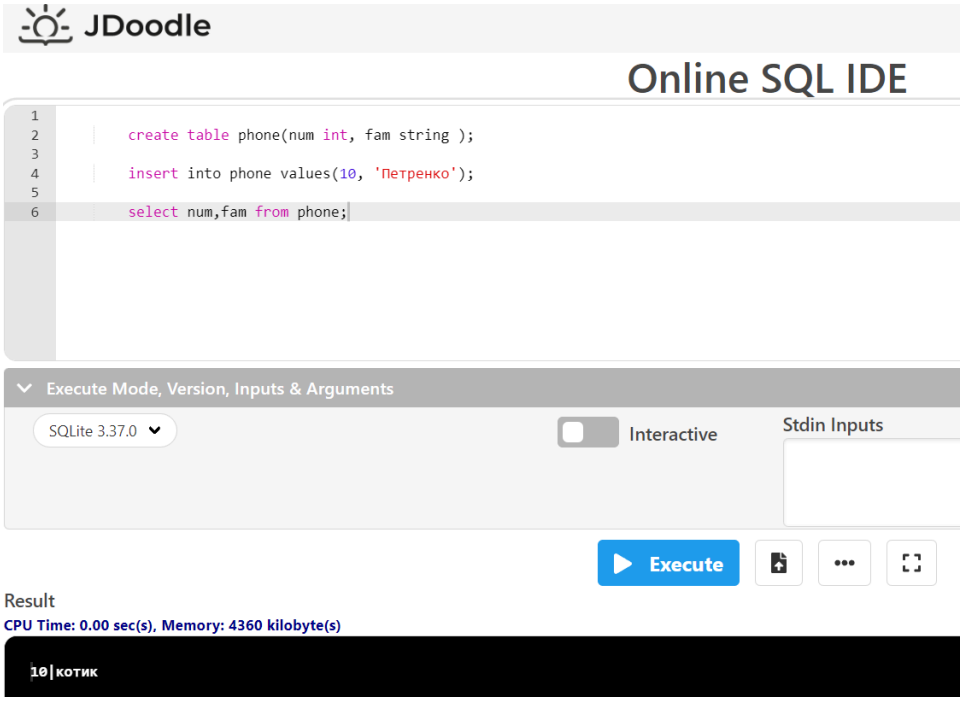
2. Додамо запис одного контакту:



```
INSERT INTO phone values(10, 'Петренко');
```

3. Виберемо поля для виведення:

```
SELECT num, fam FROM phone;
```



Завдання 2

За посиланням на сайті JDoodle.

1. Почнемо з того, що створимо нову таблицю **phone** для контактів:

```
CREATE TABLE phone(num int, fam string);
```

2. *І у нас є перший співробітник! До команди доєднується дизайнер Петренко Іван.*

```
INSERT INTO phone values(10, 'Петренко');
```

3. Щоб побачити, який вигляд має наша таблиця зараз, виберемо всі поля за допомогою інструкції select:

```
SELECT num, fam FROM phone;
```

4. *Виявилося, що нам важливо мати дані щодо адреси наших співробітників (будемо замовляти доставку піци?).*

Додайте до створеної таблиці ще одне поле: адреса.

5. *Не встигли ми занести перший запис, як до команди доєднуються ще двоє розробників: Петренко Сергій і Коваленко Тетяна.*

Додайте до створеної бази даних ще 2 записи для нових контактів.

```
CREATE table phone(num int, fam string , name string);
      INSERT INTO phone values(0505721022, 'Петренко','Іван');
      INSERT INTO phone values(0954561232, 'Петренко', 'Сергій');
      INSERT INTO phone values(067451235, 'Коваленко', 'Тетяна');
SELECT num,fam, name FROM phone;
```

Execute Mode, Version, In

Result

CPU Time: 0.00 sec(s), Memory: 42

```
505721022|Петренко|Іван
954561232|Петренко|Сергій
67451235|Коваленко|Тетяна
```

Фільтруємо і вибираємо записи за допомогою WHERE

WHERE використовується для фільтрації записів: вибору лише тих записів, які відповідають певній умові.

Синтаксис WHERE

SELECT поле1, поле2, ...
FROM таблиця
WHERE умова;

Operator
=
>
<
>=
<=
<>
BETWEEN
LIKE
IN

Приклад
Вибір всіх контактів з прізвищем Петренко:

```
SELECT *
FROM phone
WHERE fam='Петренко';
```

Оператори AND, OR і NOT

WHERE можна комбінувати з операторами AND, OR і NOT.

Оператори AND і OR використовуються для фільтрації записів на основі кількох умов.

Приклад
Вибір всіх контактів з прізвищем Петренко з іменем Іван:

```
SELECT *
FROM phone
WHERE fam='Петренко' AND name = 'Іван';
```

Завдання 3

1. *О! У нас новий проєкт. Будемо крім вебсайтів розробляти ще й мобільний застосунок. Але, здається, ми не дуже добре спроєктували нашу базу даних. У таблиці phone доведеться додати код проєкту і занести значення.*

100 — код веброзробки.
200 — код мобільної розробки.

2. *А в якому проєкті працює Петренко?*

Виберіть для виведення дані за прізвищем.

3. *Чекайте, нам потрібен лише Іван Петренко!*
   Побудуйте запит за 2 умовами.

```
1   CREATE table phone(num int, fam string , name string);
2     INSERT INTO phone values(0505721022, 'Петренко','Іван');
3     INSERT INTO phone values(0954561232, 'Петренко', 'Сергій');
4     INSERT INTO phone values(067451235, 'Коваленко', 'Тетяна');
5
6       SELECT * FROM phone
7           WHERE fam='Петренко' and name = 'Іван';
8
9
```

Execute Mode, Version, Inputs &

Execute

Result

CPU Time: 0.00 sec(s), Memory: 4316 kilo

```
505721022|Петренко|Іван
```

LIKE

Якщо потрібно знайти не точний збіг, а таке чи такі значення, що відповідають певному шаблону, — для пошуку використовується оператор LIKE.
У поєднанні з LIKE-оператором часто використовуються два символи підстановки:

- знак відсотка (%) означає нуль, один або декілька символів;
- знак підкреслення (_) означає один символ.

Синтаксис LIKE

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Приклади різних LIKE-операторів із символами узагальнення «%» і «_».

WHERE col LIKE 'a%'	Знаходить будь-які значення, що починаються з «a»
WHERE col LIKE '%a'	Знаходить будь-які значення, які закінчуються на «a»
WHERE col LIKE '%ко%'	Знаходить будь-які значення, які мають «ко» в будь-якій позиції
WHERE col LIKE '_p%'	Знаходить будь-які значення, які мають «p» на другій позиції
WHERE col LIKE 'a_%'	Знаходить будь-які значення, що починаються з «a» і містять принаймні 2 символи
WHERE col LIKE 'a%o'	находить будь-які значення, які починаються з «a» і закінчуються на «o»

Також можна комбінувати будь-яку кількість умов за допомогою операторів AND або OR.

Завдання 4

Шукаємо в базі записи за допомогою LIKE.

Приклад

```
WHERE fam LIKE 'К%';
```

```
CREATE table phone(num int, fam string , name string);
  INSERT INTO phone values(0505721022, 'Петренко','Іван');
  INSERT INTO phone values(0954561232, 'Петренко', 'Сергій');
  INSERT INTO phone values(067451235, 'Коваленко', 'Тетяна');

    SELECT * FROM phone
        WHERE fam LIKE 'К%';
```

Execute Mode, Version, Inp

Result

CPU Time: 0.00 sec(s), Memory: 4316

67451235|Коваленко|Тетяна

Додайте в таблицю дані про співробітників з такими значеннями, щоб можна було побудувати запити таких видів:

1. Знаходить будь-які значення, які закінчуються на «о».
2. Знаходить будь-які значення, які мають «ко» в будь-якій позиції.
3. Знаходить будь-які значення, які починаються з «А» і закінчуються на «в»
4. Знаходить записи всіх прізвищ, що починаються на «Кі».

Видалення та оновлення записів в базі даних

Видалення записів DELETE

DELETE — видаляє дані з бази даних.

Синтаксис DELETE

DELETE FROM таблиця WHERE умова;

Приклад

Видалимо з бази запис Петренка Івана.

```
DELETE FROM phone WHERE   fam='Петренко' AND name = 'Іван';

SELECT  num, fam, name FROM phone;
```

Оновлення бази даних UPDATE

UPDATE використовується для зміни наявних записів у таблиці.

Синтаксис UPDATE

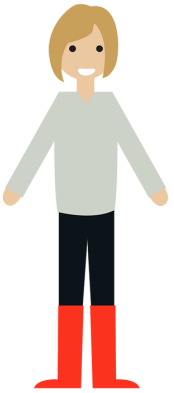
UPDATE таблиця
SET поле 1 = значення, поле 2 = значення, ...
WHERE умова;
Приклад

```
UPDATE phone
SET fam='Мельник', name='Дарина'
WHERE num=0505721022;
```

Завдання 5

1. Поки ми розбирали видалення й оновлення бази даних, сталися 2 події.

Петренко Сергій перейшов працювати в іншу компанію, припинив співпрацю з нами, тож його контакт більше не потрібен.



Видаліть контакт з бази.

2. *Петренко Іван також вирішив покинути проєкт, але на своє місце запропонував Мельник Дарину і передав їй свій корпоративний номер телефону.*

Зробіть заміну в базі даних.

Функції SQL

SQL має достатній набір функцій, які дають змогу провести аналіз даних, а також впорядкувати їх.
Основні функції

Функція	Опис	Приклад
MIN() і MAX()	Функція MIN() повертає найменше значення вибраного стовпця Функція MAX() повертає найбільше значення вибраного стовпця	SELECT MAX(num) FROM phone;
COUNT()	Функція COUNT()повертає кількість рядків, які відповідають заданому критерію	SELECT COUNT(num) FROM phone;
AVG()	Функція AVG() повертає середнє значення числового стовпця	SELECT AVG(num) FROM phone;
SUM()	Функція SUM() повертає загальну суму числового стовпця	SELECT SUM(num) FROM phone;

Сортування в базі даних

ORDER BY

`ORDER BY` використовується для сортування набору результатів у порядку зростання або спадання.

За замовчуванням ключове слово сортує записи в порядку зростання. Щоб відсортувати записи в порядку спадання, використовуйте `DESC`.

```
CREATE table phone(num int, fam string , name string);
      INSERT INTO phone values(0505721022, 'Петренко','Іван');
      INSERT INTO phone values(0954561232, 'Петренко', 'Сергій');
      INSERT INTO phone values(067451235, 'Коваленко', 'Тетяна');
SELECT num,fam, name FROM phone;
SELECT *  FROM phone
ORDER BY fam;
```

Завдання 6

У нас уже багато співробітників, для звіту потрібно вивести всі дані списку контактів у впорядкованому вигляді.

Відсортуйте таблицю за зростанням за полем «Прізвище».

Вибір даних з двох таблиць

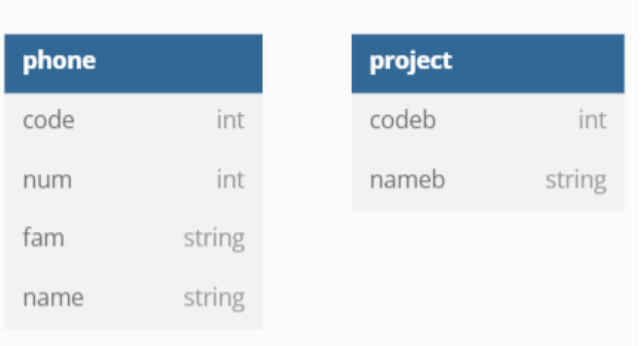
Створимо таблицю project, у якій будуть зберігатися код та назва проєкту.

```
CREATE table project(codeb int, nameb string );
    INSERT INTO project values(100, 'web');
    INSERT INTO project values(200, 'mobile');
    SELECT codeb, nameb FROM project;
```

У нас є таблиця контактів, в яку ми додамо поле **code.**

```
CREATE table phone(code int,num int,  fam string , name string);
   INSERT INTO phone values(100,0505721022, 'Петренко', 'Іван');
    INSERT INTO phone values(100,0954561232, 'Петренко', 'Сергій');
    INSERT INTO phone values(200,067451235, 'Коваленко', 'Тетяна');
```

Це поле нам потрібно для того, щоб зв'язати ці 2 таблиці й отримувати з таблиці моделей характеристики телефону.



Щоб встановити зв'язок між таблицями, використовуємо інструкцію INNER JOIN.

```
SELECT fam, name, num, nameb FROM phone
INNER JOIN
 brand
ON code=codeb
```

Щоб отримати інформацію за конкретним контактом, додамо умову:

```
SELECT fam, name, num, nameb FROM phone
INNER JOIN project
ON code=codeb
WHERE fam = 'Петренко'
```

Завдання 7

Студія набирає обертів, завдання ускладнюються!

Потрібна термінова інформація про те, в якому проєкті працює Петренко (здається, він повернувся у проєкт). Якщо у вас в базі його немає, додавайте й оновлюйте, всі Петренки з нами.
А ще просять для звіту інформацію про всіх співробітників — у яких проєктах вони зараз працюють.

```
1  CREATE table phone(code int, num int, fam string , name string);
2        INSERT INTO phone values(100,0505721022, 'Петренко','Іван');
3        INSERT INTO phone values(100,0954561232, 'Петренко', 'Сергій');
4        INSERT INTO phone values(200, 067451235, 'Коваленко', 'Тетяна');
5  SELECT num,fam, name FROM phone;
6
7  CREATE table project(codeb int, nameb string );
8      INSERT INTO project values(100, 'web');
9      INSERT INTO project values(200, 'mobile');
10     SELECT codeb, nameb FROM project;
11
12 SELECT fam, name, num, nameb FROM phone
13 INNER JOIN project
14 ON code=codeb
15 WHERE fam = 'Петренко'
16
17
18
```

Execute Mode, Version, Inputs

Result

CPU Time: 0.00 sec(s), Memory: 4424 ki

```
505721022|Петренко|Іван
954561232|Петренко|Сергій
67451235|Коваленко|Тетяна
100|web
200|mobile
Петренко|Іван|505721022|web
Петренко|Сергій|954561232|web
```

Вибір даних з багатьох таблиць

Таблиці поєднані між собою запитами, але у нас знову виникла потреба в змінах (так, проєктування бази даних без розуміння всього функціоналу на етапі проєктування забирає дуже багато сил і часу на етапі реалізації!).
З'явилася потреба зберігати дані щодо оплати роботи в кожному продукті, а також запис дедлайнів проєктів.
Додамо ще одну таблицю payment, в якій буде зберігатись оплата працівників і дедлайни за кожним із проєктів:

```
CREATE table payment(codes int, price double, deadline date );
INSERT INTO payment values(100, 16000.00, '2023-12-10');
INSERT INTO  payment values(200, 9500.00,'2024-01-01');
SELECT  codes, price, deadline FROM payment;
```

Здається, все готово.

Ще раз звернемо увагу на те, що спочатку ми мали спроєктувати схему, а вже потім писати код, але на початку роботи з базами так завжди й буває. Це досвід проєктування.
Залишається зв'язати нову таблицю з двома іншими:


```
CREATE table phone(code int, num int, fam string , name string);
      INSERT INTO phone values(100,0505721022, 'Петренко','Іван');
      INSERT INTO phone values(100,0954561232, 'Петренко', 'Сергій');
      INSERT INTO phone values(200, 067451235, 'Коваленко', 'Тетяна');

CREATE table project(codeb int, nameb string );
    INSERT INTO project values(100, 'web');
    INSERT INTO project values(200, 'mobile');

CREATE table payment(codes int, price double, deadline date );
INSERT INTO payment values(100, 16000.00, '2023-12-10');
INSERT INTO  payment values(200, 9500.00,'2024-01-01');
/*SELECT  codes, price, deadline FROM payment;*/

SELECT phone.fam, phone.name, phone.num, project.nameb,
payment.price FROM phone
INNER JOIN project
```

```
ON code=codeb
INNER JOIN payment
ON codeb=codes
```

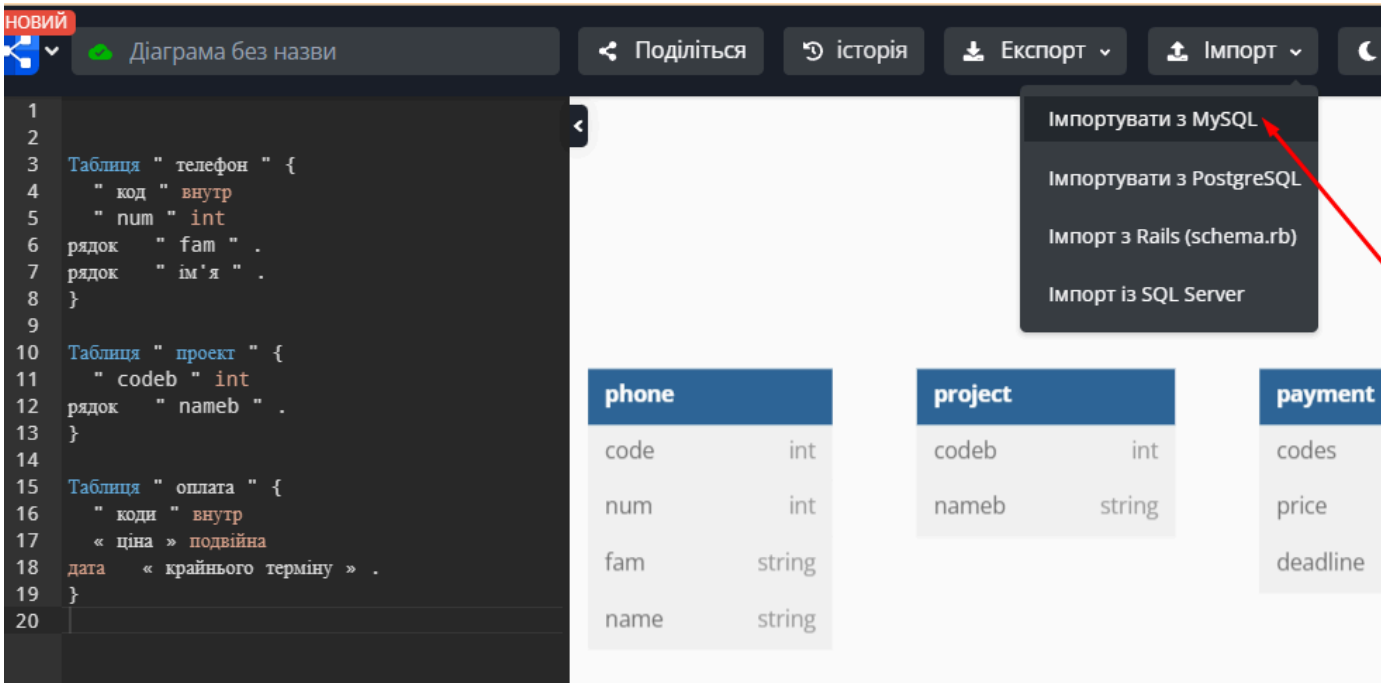
Щоб забезпечити вибір, достатньо сформулювати умову.
Наприклад, виберемо дані з усіх пов’язаних таблиць за одним прізвищем:

```
WHERE fam = 'Петренко'
```

Завдання 8

Наче все! Завершуємо роботу.

1. Зв’яжіть усі 3 таблиці в базі даних.
2. Додайте коментарі в код.
3. Додайте по 1 полю в кожну таблицю.
4. Додайте запит за номером мобільного телефону.
5. Скільки грошей отримає Коваленко, якщо завершить проєкт вчасно?
6. Побудуйте схему зв’язків за SQL кодом в онлайн-ресурсі diagram.io.



Підсумуємо

Надішліть документ з виконаним завданням.

- ☐ Які завдання ви змогли розв’язати без проблем?
- ☐ Де саме були основні проблеми при роботі з базою даних: на етапі проєктування чи на етапі програмування?
- ☐ Які інструкції SQL ви б внесли в ТОП інструкцій? Чому?
- ☐ На яке завдання пішло найбільше часу?
- ☐ Чого не вистачало, щоб зробити його швидше?

DESIGN STUDIO

Студія Дизайну дякує за хорошу роботу!