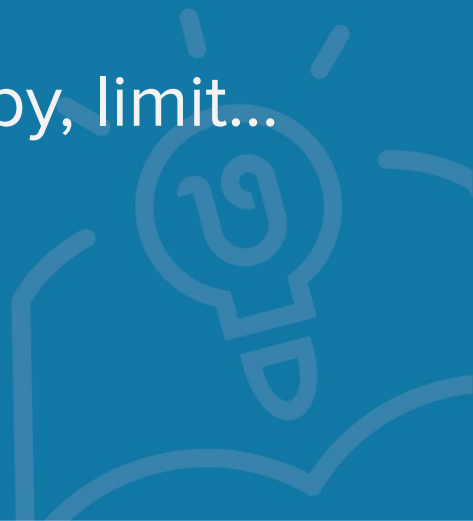
Data querying

select, where, in, between, like, order by, limit...



Preparation

- Download SQL file from this link
- This file contains test data
- Open command line
- Import data with this command:

```
mysql -u root -p vivify_blog < [path to the downloaded file]
```

- You will need to enter mysql password (it's "vivify")
- After this, you will have test data in "**vivify_blog**" database

The SELECT statement

```
SELECT column1, column2, ... FROM table_name;
```

- The SELECT statement is used to select data from a database
- The data returned is stored in a result table, called the result-set

The SELECT statement (examples)

```sql
SELECT id, title FROM posts;
```

```sql
SELECT id, title, content, category, created_at FROM posts;
```

```sql
SELECT * FROM posts;
```

The SELECT DISTINCT statement

```
SELECT DISTINCT column1, column2, ... FROM table_name;
```

- The SELECT DISTINCT statement is used to return only distinct (different) values

- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values

The SELECT DISTINCT statement

```
SELECT DISTINCT category FROM posts;
```

```
SELECT DISTINCT category, rating FROM posts;
```

The WHERE clause

```
SELECT column1, column2, ... FROM table_name WHERE condition;
```

- The WHERE clause is used to filter records.

- The WHERE clause is used to extract only those records that fulfill a specified condition.

Operators in the WHERE clause

operator	description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

The WHERE clause (examples)

```
SELECT category, rating, title FROM posts WHERE category = 'fashion';

SELECT category, rating, title FROM posts WHERE category <> 'sports';

SELECT category, rating, title, created_at FROM posts WHERE rating > 8;

SELECT category, rating, title, created_at FROM posts WHERE rating < 5;

SELECT * FROM posts WHERE rating >= 5;

SELECT * FROM posts WHERE rating <= 3;
```

The BETWEEN operator

```
SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;
```

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates

- The BETWEEN operator is inclusive: begin and end values are included

The IN operator

```
SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2, ...);

SELECT column_name(s) FROM table_name WHERE column_name IN (SELECT STATEMENT);
```

- The IN operator allows you to specify multiple values in a WHERE clause

The LIKE operator

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

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column

- There are two wildcards used in conjunction with the LIKE operator:
  - % - The percent sign represents zero, one, or multiple characters
  - _ - The underscore represents a single character

The LIKE operator wildcards

wildcard	description
‘eur%’	finds any values that start with ‘eur’
‘%ia’	finds any values that end with ‘ia’
‘%or%’	finds any values that have ‘or’ in any position
‘_r%’	finds any values that have ‘r’ in the second position
‘a_%_%’	finds any values that start with ‘a’ and have at least 3 characters
‘a%r’	finds any values that start with ‘a’ and end with ‘r’

The BETWEEN, IN and LIKE operators (examples)

```
SELECT title, category, rating, teaser FROM posts WHERE rating BETWEEN 3 AND 7;

SELECT title, created_at FROM posts WHERE created_at BETWEEN '2015-01-01' AND
'2016-01-01';

SELECT title, category, teaser FROM posts WHERE category LIKE '%s';

SELECT title, category, teaser FROM posts WHERE title LIKE 'Object%';

SELECT title, category, teaser FROM posts WHERE category LIKE 'f _ _ d';

SELECT * FROM posts WHERE category IN ('sports', 'health');

SELECT * FROM posts WHERE category IN (SELECT category FROM posts WHERE rating = 3);
```

The AND, OR and NOT operators

```
SELECT column1, column2, ... FROM table_name WHERE condition1 AND condition2 AND
condition3 ...;
```

```
SELECT column1, column2, ... FROM table_name WHERE condition1 OR condition2 OR
condition3 ...;
```

```
SELECT column1, column2, ... FROM table_name WHERE NOT condition;
```

The AND, OR and NOT operators

operator	description
AND	The AND operator displays a record if all the conditions are true
OR	The OR operator displays a record if at least one condition is true
NOT	The NOT operator displays a record if condition(s) is not true

The AND, OR and NOT operators (examples)

```
SELECT title, category, rating FROM posts WHERE category = 'fashion' AND rating < 5;

SELECT title, category, rating FROM posts WHERE rating BETWEEN 6 AND 10 AND category
IN ('food', 'health', 'fashion');

SELECT title, category, rating FROM posts WHERE category IN ('sports', 'politics') OR
title LIKE '%implementation' OR rating <= 3;

SELECT * FROM posts WHERE NOT rating = 5;

SELECT * FROM posts WHERE rating NOT IN (1, 3, 5, 7, 9);
```

The ORDER BY keyword

```
SELECT column1, column2, ... FROM table_name ORDER BY column1, column2, ... ASC|DESC;
```

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

The ORDER BY keyword (examples)

```
SELECT title, category, rating FROM posts ORDER BY rating;

SELECT * FROM posts ORDER BY category DESC;

SELECT rating, category, title FROM posts ORDER BY rating ASC, category DESC;

SELECT id, category, rating, title FROM posts WHERE rating > 7 ORDER BY title DESC;
```

The LIMIT keyword

```
SELECT column1, column2, ... FROM table_name LIMIT number;
```

- The LIMIT clause is used to specify the number of records to return

- The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance

The LIMIT keyword (examples)

```
SELECT title, category, rating FROM posts LIMIT 20;

SELECT * FROM posts WHERE rating > 8 LIMIT 10;

SELECT category, rating, title FROM posts WHERE rating > 8 ORDER BY rating DESC LIMIT
20;
```

SQL functions (MIN, MAX, SUM, AVG, COUNT...)

```sql
SELECT MIN(column_name) FROM table_name WHERE condition;

SELECT MAX(column_name) FROM table_name WHERE condition;

SELECT SUM(column_name) FROM table_name WHERE condition;

SELECT AVG(column_name) FROM table_name WHERE condition;

SELECT COUNT(column_name) FROM table_name WHERE condition;
```

SQL functions (MIN, MAX, SUM, AVG, COUNT...)

function	description
MIN()	returns the smallest value of the selected column
MAX()	returns the largest value of the selected column
SUM()	returns the total sum of a numeric column
AVG()	returns the average value of a numeric column
COUNT()	returns the number of rows that matches a specified criteria

SQL functions (examples)

```
SELECT MAX(rating) FROM posts;

SELECT MIN(rating) FROM posts;

SELECT AVG(rating) FROM posts WHERE category = 'sports';

SELECT SUM(rating) FROM posts WHERE title LIKE '%implementation';

SELECT COUNT(id) FROM posts WHERE category IN ('fashion', 'food');
```

SQL aliases

```
SELECT column_name AS alias_name FROM table_name AS alias_name;
```

- SQL aliases are used to give a table, or a column in a table, a temporary name

- Aliases are often used to make column names more readable. An alias only exists for the duration of the query

SQL aliases (examples)

```sql
SELECT title AS blog_title, rating AS grade, category FROM posts;

SELECT title AS naslov, rating AS ocena FROM posts;

SELECT title AS t, rating AS r, category AS c FROM posts;
```

The GROUP BY Statement

```
SELECT column_name(s) FROM table_name WHERE condition

GROUP BY column_name(s);
```

- The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

The SQL GROUP BY Statement (example)

```
SELECT COUNT(*), category FROM posts GROUP BY category;
```

- The number of posts by category

```
SELECT SUM(rating), category FROM posts GROUP BY category;
```

- Rating sum by category

```
SELECT AVG(rating), category FROM posts GROUP BY category;
```

- Average rating by category

The HAVING Clause

```
SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s)

HAVING condition;
```

- The SQL HAVING clause is used in combination with the GROUP BY clause to restrict the groups of returned rows to only those whose condition is TRUE.

The HAVING Clause (examples)

```
SELECT COUNT(*) as count, category FROM posts GROUP BY category

HAVING count > 15;
```

- This will return category only for those categories that have more than 15 posts