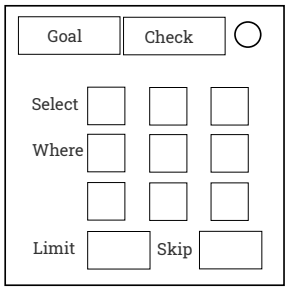
On the Subject of SQL - Basic



How fun is it to ride through a large table of numbers to find similarities...

This module is a SQL code simulator that you have to defuse by producing the proper SQL query. You have an input data in this document and the module allows you to see a goal that you must reproduce. Use the documentation to build a SQL query that will reproduce the expected solution from the input data.

The goal button

Press on the **Goal** button to show you what is expected by the module. The expectation is a matrix of numbers. The result can be from 1 to 4 rows by 2 to 3 columns. Here are a few example **but definitely not limited to:**

2	9	6
4	2	1
7	1	4
3	8	2

2	9	6
4	2	1

2	9
4	2
7	1

2	9	6

The check button

Pressing on the **Check** button tries to solve the module. When you press on **check,** the module compiles the SQL query and applies it on the input data. The result is compared to the goal matrix and you either get a strike or a solve depending if you found the proper SQL.

> **Note:** The crafted SQL doesn't need to be the one used to generate the goal matrix, as long as it generates the same result it's fine.

Solving the module

To solve the module, you have to find the numbers from the goal matrix in the input data. The rows are always from top to bottom, you'll see why later, but the columns can be in any order. Example:

Goal matrix

2	9	6
4	2	1

Example input data

A	B	C	D	E	F
9	0	1	6	7	3
6	8	2	4	5	9
2	7	5	3	6	1
1	3	4	9	5	2

The input data

The real input data that you must use as an input to solve the module is this one. You must find the values from the goal matrix inside this table and craft the proper SQL query to generate the same output.

A	B	C	D	E	F	G
9	3	0	2	0	4	6
7	8	6	1	2	9	8
2	0	5	4	7	3	1
4	7	1	5	3	0	9
6	2	7	9	8	6	5
1	9	3	7	6	5	4
8	5	4	3	1	8	2

Remember:

SQL results are not necessarily in reading order, it will be **from top to bottom** but you can **SELECT** any column you wish in any order.

On the subject of SQL

SQL is a programming language that describes how to manipulate sources of data and generate a difference output source. It is comprised of many different keywords, only a minimal list is used in this module.

How SQL is processed

SQL is processed always in the same way:

1. Each row is compared against the filters. If a row complies with the filter expression it is kept for further processing. (See **WHERE** clauses below for more information)

2. Each row that passes the filter expression is then processed against the selection expressions. (See **SELECT** clauses below for more information)

3. Finaly, each row of the end result is processed against the LIMIT clause; that is **SKIP** X amount of rows from the result and then **TAKE** only Y lines after that. (See **LIMIT** clauses below for more information)

SELECT X, Y, Z

When you **SELECT** something, you choose to take all the values that match this X, Y or Z expression and return it. The module offers 3 possible return expressions and they are always simple column references. You must click to select each column to return, you can select between A and G or disable the selection when it shows a "-". Here's an example of how it works:

A	B	C	D	E	F	G
9	3	0	2	0	4	6
7	8	6	1	2	9	8
2	0	5	4	7	3	1
4	7	1	5	3	0	9
6	2	7	9	8	6	5
1	9	3	7	6	5	4
8	5	4	3	1	8	2

SQL: SELECT D, A

Notice there are no 3rd expression. We used a "-".

D	A
2	9
1	7
4	2
5	4
9	6
7	1
3	8

WHERE X operation Y

WHERE introduces filters and all rows will be tested against the conditions you provide the module with. A row is kept in the output only if it complies with the filter condition. This SQL emulator makes it simple for you, you can only compare columns to fixed values such as: A < 2, C = 3, F >= 5. The possible operators are all simple mathematical operators that you already know (In most cases) such as:

- Equals or "="
- Not equals or "<>"
- Less than or "<"
- Less than or equal or "<="
- Greather than or ">"
- Greather than or equal or ">="

Here's a basic example of a single filter:

A	B	C	D	E	F	G
9	3	0	2	0	4	6
7	8	6	1	2	9	8
2	0	5	4	7	3	1
4	7	1	5	3	0	9
6	2	7	9	8	6	5
1	9	3	7	6	5	4
8	5	4	3	1	8	2

SQL: WHERE C >= 5

A	B	C	D	E	F	G
7	8	6	1	2	9	8
2	0	5	4	7	3	1
6	2	7	9	8	6	5

Note: Conditions can be applied to any column, not just the **SELECT**ed ones.

WHERE X AND/OR Y

If you think you need 2 conditions to achieve the result, click on the button below the **WHERE** label to introduce a 2nd filter. You can choose between **AND** and **OR** to connect the two filters. **AND** means both conditions must pass to keep the row, **OR** means at least one condition must pass to keep the row. Examples:

Example data	Condition	Result
A B C 9 3 0 7 8 6 2 0 5 4 7 1	WHERE A < 7 AND B >= 3	A B C ~~9 3 0~~ ~~7 8 6~~ ~~2 0 5~~ 4 7 1
A B C 9 3 0 7 8 6 2 0 5 4 7 1	WHERE B <> 8 OR C <= 2	A B C 9 3 0 ~~7 8 6~~ 2 0 5 4 7 1

LIMIT X, Y

LIMIT comes with one or two values. For simplicity, the SQL module splits them for you in two selectors:

- **LIMIT:** Indicates the number of rows to keep from the matching results after applying conditions. If you select "All", no rows are discarded.
- **SKIP:** Indicates the number of rows to skip after applying conditions. If you select "None", no rows are discarded.

Examples:

<table>
<tr><th>Example data</th><th>Condition</th><th>Result</th></tr>
<tr><td><table><tr><td>9</td><td>3</td><td>0</td></tr><tr><td>7</td><td>8</td><td>6</td></tr><tr><td>2</td><td>0</td><td>5</td></tr><tr><td>4</td><td>7</td><td>1</td></tr></table></td><td>LIMIT 2, 0

(Limit of 2; Skip none)</td><td><table><tr><td>9</td><td>3</td><td>0</td></tr><tr><td>7</td><td>8</td><td>6</td></tr></table></td></tr>
<tr><td><table><tr><td>9</td><td>3</td><td>0</td></tr><tr><td>7</td><td>8</td><td>6</td></tr><tr><td>2</td><td>0</td><td>5</td></tr><tr><td>4</td><td>7</td><td>1</td></tr></table></td><td>LIMIT 999, 1

(No limit; Skip 1)</td><td><table><tr><td>7</td><td>8</td><td>6</td></tr><tr><td>2</td><td>0</td><td>5</td></tr><tr><td>4</td><td>7</td><td>1</td></tr></table></td></tr>
<tr><td><table><tr><td>9</td><td>3</td><td>0</td></tr><tr><td>7</td><td>8</td><td>6</td></tr><tr><td>2</td><td>0</td><td>5</td></tr><tr><td>4</td><td>7</td><td>1</td></tr></table></td><td>LIMIT 1, 2

(Limit of 1; Skip 2)</td><td><table><tr><td>2</td><td>0</td><td>5</td></tr></table></td></tr>
</table>