SQL statements for one table

This section contains SQL statements, that are applied for one table. Consider the following situation: We have one table with name table_name. This table has n, n natural number, columns with names column_1, column_2, … , column_n. Every column has m, m natural number, values. The i-th column, $1 \leq i \leq n$, has the values v_{1, i}, v_{2, i}, … , v_{m, i}.

Selecting columns

```
/* Selecting the first column with every value in it. */
SELECT colum_1
FROM table_name;
```

```
/* Selecting the i-th column, 1 ≤ i ≤ n, with every value in it. */
SELECT colum_i
FROM table_name;
```

```
/* Selecting the last column with every value in it. */
SELECT colum_n
FROM table_name;
```

```
/* Selecting the first and the second column with every value in it. */
SELECT colum_1,  colum_2
FROM table_name;
```

```
/* Selecting the i_{1}-th and i_{2}-th column, 1 ≤ i_{1}, i_{2} ≤ n and i_{1} ≠ i_{2},  with every value in it. */
SELECT colum_i_{1},  colum_i_{2}
FROM table_name;
```


Document written by Artur Sanin

```
/* Selecting the i_{1}-th, i_{2}-th, …. , i_{q}-th column, 1 ≤ i_{1}, i_{2} , …. , i_{q} ≤ n
and  i_{1} ≠ i_{2} ≠ ….  ≠ i_{q}, q natural number with q ≤ n. */
SELECT column_i_{1}, column_i_{2}, …. , column_i_{q}
FROM table_name;


/* Selecting every column, with every value in it. */
SELECT column_1, column_2, …. , column_n
FROM table_name;
        or
/* Selecting every column, with every value in it. */
SELECT *
FROM table_name;
```

Ordering results

To sort your query results, use the ORDER BY keyword. Note that the order is ascending by default. If the column, by which you want the results ordered, is numerical, then you get the lowest number first ascending to the highest number. If the column, by which you want the results ordered, is a string, then you get an alphabetical order.

Note if ordered by several columns, then your results will be ordered by the first specified column first, if the same value accurse multiple times, then those will be ordered by the next column and so on.

```
/* Selecting column_i, 1 ≤ i ≤ n, from table_name and ordering the fields by column_i in ascending order. */
SELECT  column_i
FROM table_name
ORDER BY column_i ASC;
```

or

```
/* Selecting column_i, 1 ≤ i ≤ n, from table_name and ordering the fields by column_i in ascending order. */
SELECT  column_i
FROM table_name
ORDER BY column_i;
```

```
/* Selecting column_i, 1 ≤ i ≤ n, from table_name and ordering the fields by column_i in descending order. */
SELECT  column_i
FROM table_name
ORDER BY column_i DESC;
```

```
/* Selecting column_i, 1 ≤ i ≤ n, from table_name and ordering the fields by column_j, 1 ≤ j ≤ n and i ≠ j,  in ascending or descending order. */
SELECT  column_i
FROM table_name
ORDER BY column_j ASC/DESC;
```

```
/* Selecting column_i, 1 ≤ i ≤ n, from table_name and ordering the fields by column_j_{1}
ascending or descending order first and column_j_{2} ascending or descending order second,
1 ≤ j_{1}, j_{2} ≤ n and j_{1} ≠ j_{2}. */
SELECT  column_i
FROM table_name
ORDER BY column_j_{1} ASC/DESC, column_j_{2} ASC/DESC;


/* Selecting column_i, 1 ≤ i ≤ n, from table_name and ordering the fields by column_j_{2}
ascending or descending order first and column_j_{1} ascending or descending order second,
1 ≤ j_{1}, j_{2} ≤ n and j_{1} ≠ j_{2}. */
SELECT  column_i
FROM table_name
ORDER BY column_j_{2} ASC/DESC, column_j_{1} ASC/DESC;


/* Selecting column_i, 1 ≤ i ≤ n, from table_name and ordering the fields by column_j_{1}
ascending or descending order first, column_j_{2} ascending or descending order second, ….
, column_j_{p} ascending or descending order last, 1 ≤ j_{1}, j_{2}, …., j_{p} ≤ n and j_{1}
≠ j_{2} ≠ …. ≠ j_{p}, p natural number and p ≤ n. */
SELECT  column_i
FROM table_name
ORDER BY column_j_{1} ASC/DESC, column_j_{2} ASC/DESC, …., column_j_{p}
ASC/DESC;
```

Note that one of the columns column_j_{1}, column_j_{2}, …. , column_j_{p} can be column_i.

```
/* Selecting column_i_{1} and column_i_{2}, 1 ≤ i_{1}, i_{2} ≤ n and i_{1} ≠  i_{2}, from
table_name and ordering by column_i_{1} in ascending or descending order. */
SELECT  column_i_{1}, column_i_{2}
FROM table_name
ORDER BY column_i_{1} ASC/DESC;


/* Selecting column_i_{1} and column_i_{2}, 1 ≤ i_{1}, i_{2} ≤ n and i_{1} ≠  i_{2}, from
table_name and ordering by column_i_{2} in ascending or descending order. */
SELECT  column_i_{1}, column_i_{2}
FROM table_name
ORDER BY column_i_{2} ASC/DESC;
```

```
/* Selecting column_i_{1} and column_i_{2}, 1 ≤ i_{1}, i_{2} ≤ n and i_{1} ≠ i_{2}, from
table_name and ordering by column_i_{1} ascending or descending order first and
column_i_{2} ascending or descending order second. */
SELECT  column_i_{1}, column_i_{2}
FROM table_name
ORDER BY column_i_{1} ASC/DESC, column_i_{2} ASC/DESC;


/* Selecting column_i_{1} and column_i_{2}, 1 ≤ i_{1}, i_{2} ≤ n and i_{1} ≠ i_{2}, from
table_name and ordering by column_i_{2} ascending or descending order first and
column_i_{1} ascending or descending order second. */
SELECT  column_i_{1}, column_i_{2}
FROM table_name
ORDER BY column_i_{2} ASC/DESC, column_i_{1} ASC/DESC;

/* Selecting column_i_{1} and column_i_{2}, 1 ≤ i_{1}, i_{2} ≤ n and i_{1} ≠ i_{2}, from
table_name and ordering by column_j, 1 ≤ j ≤ n and i_{1} ≠ j ≠ i_{2}, in ascending or
descending order. */
SELECT  column_i_{1}, column_i_{2}
FROM table_name
ORDER BY column_j ASC/DESC;


/* Selecting column_i_{1} and column_i_{2}, 1 ≤ i_{1}, i_{2} ≤ n and i_{1} ≠ i_{2}, from
table_name and ordering by column_j_{1} ascending or descending first, column_j_{2}
ascending or descending second, …. , column_j_{p} ascending or descending last, 1 ≤ j_{1},
j_{2}, …., j_{p} ≤ n and j_{1} ≠ j_{2} ≠ …. ≠ j_{p}, p natural number and p ≤ n. */
SELECT  column_i_{1}, column_i_{2}
FROM table_name
ORDER BY column_j_{1} ASC/DESC, column_j_{2} ASC/DESC, …., column_j_{p}
ASC/DESC;
```

Note that the columns column_j_{1}, column_j_{2}, …. , column_j_{p} can be column_i_{1} or column_i_{2}.

```
/* Selecting columns column_i_{1}, column_i_{2}, … , column_i_{q}, 1 ≤ i_{1}, i_{2}, … ,
i_{q} ≤ n and i_{1} ≠ i_{2} ≠ … ≠ i_{q}, q natural number and q ≤ n, in table_name ordered
by column_j_{1} ascending or descending first, column_j_{2} ascending or descending
```

```
second, … column_j_{p} ascending or descending last, 1 ≤ j_{1},  j_{2}, … , j_{p} ≤ n and
j_{1} ≠ j_{2} ≠ …  ≠ j_{p}, p natural number and p ≤ n. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{q}
FROM table_name
ORDER BY column_j_{1} ASC/DESC, column_j_{2} ASC/DESC, … , column_j_{p}
ASC/DESC;
```

Note that the column_j_{1}, column_j_{2}, …. , column_j_{p} can be one of the columns column_i_{1}, column_i_{2}, …. , column_i_{q}.

```
/* Selecting all records from table_name ordered by column_j, 1 ≤ j ≤ n, in ascending order.
*/
SELECT *
FROM table_name
ORDER BY column_j ASC;
```

or

```
/* Selecting all records in table_name ordered by column_j, 1 ≤ j ≤ n, in ascending order. */
SELECT *
FROM table_name
ORDER BY column_j;
```

```
/* Selecting all records in table_name ordered by column_j, 1 ≤ j ≤ n, in descending order. */
SELECT *
FROM table_name
ORDER BY column_j DESC;
```

```
/* Selecting all records from table_name and ordering by column_j_{1} ascending or
descending order first and column_j_{2} ascending or descending order second.  */
SELECT *
FROM table_name
ORDER BY column_j_{1} ASC/DESC, column_j_{2} ASC/DESC;
```

```
/* Selecting all records from table_name and ordering by column_j_{2} ascending or
descending order first and column_j_{1} ascending or descending order second. */
SELECT *
FROM table_name
```

```
ORDER BY column_j_{2} ASC/DESC, column_j_{1} ASC/DESC;
```

```
/* Selecting all records in table_name ordered by column_j_{1} ascending or descending order first, column_j_{2} ascending or descending order second, … column_j_{p} ascending or descending order last, 1 ≤ j_{1}, j_{2}, …., j_{p} ≤ n and j_{1} ≠ j_{2} ≠ …. ≠ j_{p}, p natural number and p ≤ n */
SELECT *
FROM table_name
ORDER BY column_j_{1} ASC/DESC, column_j_{2} ASC/DESC, … , column_j_{p} ASC/DESC;
```

Filtering records

This section contains general formulated statements for filtering records and fields from a table. We use the WEHRE keyword with a specified condition, that the records should meet. Conditions can be combined with the AND, OR, NOT keywords.

```
/* Selecting only records in table_name where column_i, 1 ≤ i ≤ n, has value v_{j, i}, 1 ≤ j ≤ m.*/
SELECT *
FROM table_name
WHERE column_i = v_{j, i};
```

```
/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, or column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2}, i_{2}}, 1 ≤ j_{2} ≤ m. */
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) OR (column_i_{2} = v_{j_{2}, i_{2}});
```

```
/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, and column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2}, i_{2}}, 1 ≤ j_{2} ≤ m. */
```

```
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) AND (column_i_{2} = v_{j_{2}, i_{2}});


/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value
v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, or column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2},
i_{2}}, 1 ≤ j_{2} ≤ m or … or column_i_{p}, 1 ≤ i_{p} ≤ n, has value v_{j_{p}, i_{p}}, 1 ≤
j_{p} ≤ m, */
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) OR (column_i_{2} = v_{j_{2}, i_{2}}) OR …
OR (column_i_{p} = v_{j_{p}, i_{p}});


/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value
v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, and column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2},
i_{2}}, 1 ≤ j_{2} ≤ m and … and column_i_{p}, 1 ≤ i_{p} ≤ n, has value v_{j_{p}, i_{p}},
1 ≤ j_{p} ≤ m, */
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) AND (column_i_{2} = v_{j_{2}, i_{2}}) AND
… AND (column_i_{p} = v_{j_{p}, i_{p}});


/* Selecting all records in table_name. 1 ≤ i ≤ n is arbitrary. */
SELECT *
FROM table_name
WHERE (column_i = v_{1, i}) OR (column_i = v_{2, i}) OR … OR (column_i = v_{m, i});


/* General syntax for filtering records. */
SELECT *
FROM table_name
WHERE condition;
        or
/* General syntax for filtering records, j some natural number AND/OR = AND or OR. */
SELECT *
FROM table_name
WHERE condition_1 AND/OR condition_2 AND/OR … AND/OR condition_j;
```

```
/* Filtering all records, that do not satisfy the condition. */
SELECT *
FROM table_name
WHERE NOT condition ;


/* General syntax for filtering fields form specific columns. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{j}
FROM table_name
WHERE condition;
        or
/* General syntax for filtering fields, p some natural number AND/OR = AND or OR. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{j}
FROM table_name
WHERE condition_1 AND/OR condition_2 AND/OR … AND/OR condition_p;


/* Filtering fields, that do not satisfy a condition. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{j}
FROM table_name
WHERE NOT condition;
```

Databases

```
/* Creating a database with name my_database. */
CREATE DATABASE my_database;


/* Droping an existing database with name my_database. */
DROP DATABASE my_database;


/* Showing all currently existing databases. */
SHOW DATABASES;
```

Tables

```
/* Creating a table with name my_table, that has one column column_1 with datatyp type_1.
*/
CREATE TABLE my_table (
        column_1 type_1
);


/* Creating a table with name my_table, that has one columns column_1 with datatyp type_1
and column_2 with datatyp type_2. */
CREATE TABLE my_table (
        column_1 type_1,
        column_2 type_2
);


/* Creating a table with name my_table, that has n columns column_1 with datatype type_1,
column_2 with datatype type_2, …. and column_n with datatyp type_n. */
CREATE TABLE my_table (
        column_1 type_1,
        column_2 type_2,
        .
        .
        .
```

```
        column_n type_n
);


/* Copying an existing table with name my_table. */
CREATE TABLE copy_my_table AS
        SELECT *
        FROM  my_table;



/* Copying only records of my_table, that satisfy an condition. */
CREATE TABLE copy_my_table AS
        SELECT *
        FROM  my_table
        WHERE condition;


/* Copying the columns column_i_{1}, column_i_{2}, … , column_i_{p}, 1 ≤ i_{1}, i_{2},
… , i_{p} ≤ n and p natural number, of an existing table with name my_table. */
CREATE TABLE copy_my_table AS
        SELECT column_i_{1}, column_i_{2}, … , column_i_{p}
        FROM  my_table;


/* Copying the columns column_i_{1}, column_i_{2}, … , column_i_{p}, 1 ≤ i_{1}, i_{2},
… , i_{p} ≤ n and p natural number, of an existing table with name my_table that meet a
condition. */
CREATE TABLE copy_my_table AS
        SELECT column_i_{1}, column_i_{2}, … , column_i_{p}
        FROM  my_table
        WHERE  condition;


/* Inserting one value in a table with only one column. */
INSERT INTO my_table (column_1)
VALUES
(v_{1, 1});


/* Inserting m values in a table with only one column. */
```

```
INSERT INTO my_table (column_1)
VALUES
(v_{1, 1}),
(v_{2, 1}),
(v_{3, 1}),
.
.
.
(v_{m, 1});
/* Inserting one record in a table with only two columns. */
INSERT INTO my_table (column_1, column_2)
VALUES
(v_{1, 1}, v_{1, 2});


/* Inserting m records in a table with two columns. */
INSERT INTO my_table (column_1, column_2)
VALUES
(v_{1, 1}, v_{1, 2}),
(v_{2, 1}, v_{2, 2}),
(v_{3, 1}, v_{3, 2}),
.
.
.
(v_{m, 1}, v_{m, 2});


/* Inserting one record in a table with n columns. */
INSERT INTO my_table (column_1, column_2, column_3, …. , column_n)
VALUES
(v_{1, 1}, v_{1, 2}, v_{1, 3}, …. , v_{1, n});
                or
/* Inserting one record in a table with n columns. */
INSERT INTO my_table
VALUES
(v_{1, 1}, v_{1, 2}, v_{1, 3}, …. , v_{1, n});


/* Inserting m records in a table with n columns. */
INSERT INTO my_table (column_1, column_2, column_3, …. , column_n)
VALUES
```

(v_{1, 1}, v_{1, 2}, v_{1, 3}, …. , v_{1, n}),
(v_{2, 1}, v_{2, 2}, v_{2, 3}, …. , v_{2, n}),
(v_{3, 1}, v_{3, 2}, v_{3, 3}, …. , v_{3, n}),
.
.
.
(v_{m, 1}, v_{m, 2}, v_{m, 3}, …. , v_{m, n});

or

```
/* Inserting m records in a table with n columns. */
INSERT INTO my_table
VALUES
(v_{1, 1}, v_{1, 2}, v_{1, 3}, …. , v_{1, n}),
(v_{2, 1}, v_{2, 2}, v_{2, 3}, …. , v_{2, n}),
(v_{3, 1}, v_{3, 2}, v_{3, 3}, …. , v_{3, n}),
.
.
.
(v_{m, 1}, v_{m, 2}, v_{m, 3}, …. , v_{m, n});
```

```
/* Inserting one record in a table with n columns, but only insert values in the columns
column_i_{1}, column_i_{2}, … , column_i_{p}, 1 ≤ i_{1}, i_{2}, … , i_{p} ≤ n and p
natural number. */
INSERT INTO my_table (column_i_{1}, column_i_{2}, column_i_{3}, … , column_i_{p})
VALUES
(v_{1, i_{1}}, v_{1, i_{2}}, v_{1, i_{3}}, …. , v_{1, i_{p}});
```

Note that every column, that no value was inserted into, will have null asigned to it.

```
/* Inserting m records in a table with n columns, but only insert values in the columns
column_i_{1}, column_i_{2}, … , column_i_{p}, 1 ≤ i_{1}, i_{2}, … , i_{p} ≤ n and p
natural number. */
INSERT INTO my_table (column_i_{1}, column_i_{2}, column_i_{3}, … , column_i_{p})
VALUES
(v_{1, i_{1}}, v_{1, i_{2}}, v_{1, i_{3}}, …. , v_{1, i_{p}}),
(v_{2, i_{1}}, v_{2, i_{2}}, v_{2, i_{3}}, …. , v_{2, i_{p}}),
(v_{3, i_{1}}, v_{3, i_{2}}, v_{3, i_{3}}, …. , v_{3, i_{p}}),
.
.
```

.
(v_{m, i_{1}}, v_{m, i_{2}}, v_{m, i_{3}}, …. , v_{m, i_{p}});

```
/* Updating all records in a table with n columns, that satisfy a condition. */
UPDATE my_table
SET column_1 = new_v_{1, 1}, column_2 = new_v_{1, 2}, …. , column_n = new_v_{1, n}
WHERE  condition;

/* If you don't have a WHERE in the statement, all records will be updated to the value
(new_v_{1, 1}, new_v_{1, 2}, …. , new_v_{1, n}). */
UPDATE my_table
SET column_1 = new_v_{1, 1}, column_2 = new_v_{1, 2}, …. , column_n = new_v_{1, n}

/* Deleting all records in table my_table, that satisfy a certain condition. */
DELETE FROM my_table
WHERE condition;

/* Deleting all records in table my_table, but not the table. */
DELETE FROM my_table;

/* Adding a new column with name new_column of datatype type in a existing table. */
ALTER TABLE my_table
ADD new_column type;

/* Droping an existing column with name ex_column in a existing table. */
ALTER TABLE my_table
DROPE COLUMN ex_column;

/* Renaming an existing column with name ex_column to the name new_column_name in a
existing table. */
ALTER TABLE my_table
RENAME COLUMN ex_column TO new_column;
```

```
/* Deleting an existing table, with name my_table. */
DROP TABLE my_table;


/* Deleting only the data in the table my_table, but not the table. */
TRUNCATE TABLE my_table;
```