SQL

Languages

> `SQL:` Structured Query Language
> `DML:` Data Manipulation Language `(INSERT/SELECT/UPDATE/DELETE)`
> `DDL:` Data Definition Language `(create/delete tables or columns)`
> `DCL:` Data Control Language `(tables access permissions)`

DML

Data Manipulation Language

Structure

```
SELECT [DISCTINCT] {expressions} [AS nickname]
 FROM [tables] [AS nickname]
    [WHERE condition]
    [GROUP BY {attributes}]
    [HAVING condition]
    [ORDER BY {attributes} [ASC/DESC]];
           or {column_nbr} [ASC/DESC]];
```

SELECT

`Selection in a data base.`

```
SELECT id, lastname
 FROM customer;
```

lucky777

> The result will be a list with all customer IDs and names.

```
SELECT *
 FROM my_table;
```

> The result will be a list with all lines of the table.*my_table*

DISTINCT

`No duplicates.`

```
SELECT DISTINCT city AS city
 FROM customer;
```

> The result will be a list with all the cities of the customers, but without duplicates.
> So even if more than one customer lives in `Brussels`, we will see it only once in our resulting list.
> The column `customer_city` of the resulting list will be named `city`.

WHERE

`Condition.`

```
SELECT lastname, city, code
 FROM customer
    WHERE city == 'Brussels' AND code != 1070;
```

> The result will be a list with the name, city and code of all customers who live in `Brussels`, but **not** in the town with the postal code `1000`.

IN

```
SELECT lastname, city
 FROM customer
    WHERE city in ('Brussels', 'Liege', 'Antwerp');
```

> The result will be a list with the name and the city of all customers who live in `Brussels`, `Liege` or `Antwerp`.

BETWEEN

```
SELECT lastname, age
 FROM customer
```

```
    WHERE age BETWEEN 18 AND 25;
```

> The result will be a list with the name of the customers from 18 to 25 years old.

LIKE

```
SELECT lastname, cat
 FROM customer
    WHERE lastname LIKE '%x%' AND cat LIKE 'B_';
    /*
    Equivalence in Linux:
        %x%   =>   *x*
        B_    =>    B?
```

> The result will be a list with the name and the category of all customers with a e in their name and with a category of two letters, starting with a B.
>
> > Examples for lastname:
> >
> > - somethingxelse *ok*
> > - xs *ok*
> > - testxxx *ok*
> > - x *ok*
> > - blbl *not ok!*
>
> > Examples for cat:
> >
> > - BA *ok*
> > - B2 *ok*
> > - BAA *not ok!*
> > - 2A *not ok!*

```
SELECT product_name AS Product, 0.21*price AS Taxe
 FROM product;
    /*
    price = price of the product
    taxe = 21% of the price
    */
```

> The result will be a list with the name and the Taxe of all products.

null

```
SELECT *
 FROM customer
    WHERE city IS null AND code IS NOT null;
      /*
```

```
        city == null -> NOT GOOD
        code != null -> NOT GOOD
        */
```

> `null` can't be compared to anything, not even with itself.
> We need to use the word `IS` instead of `=`.

Agregates

COUNT(...)

`Line counting.`

```
SELECT COUNT(lastname)
 FROM customer;
```

> The result will be a list with the `count` of lines where lastname is `non-null`.

```
SELECT COUNT(DISTINCT city)
 FROM customer;
```

> The result will be a list with the `count` of lines where city is `non-null`, but without counting twice the same city.

SUM(...)

`Sum of values.`

```
SELECT SUM(price)
 FROM product;
```

> The result will be a line with the `sum` of the price of all products in the table.
> If we have 3 products at 5€, the result will be 15.

AVG(...)

`Average of values.`

```
SELECT AVG(price)
 FROM product;
```

> The result will be a line with the `average price` of all products in the table.
> If we have 1 product at 10€, 1 at 15€ and one at 20€, the result will be 15.

MAX(...)

`Maximum value.`

```
SELECT MAX(price)
 FROM product;
```

> The result will be a line with the `maximum price` of all products in the table.
> If we have 1 product at 10€, 1 at 15€ and one at 20€, the result will be 20.

MIN(...)

`Minimum value.`

```
SELECT MIN(price)
 FROM product;
```

> The result will be a line with the `minimum price` of all products in the table.
> If we have 1 product at 10€, 1 at 15€ and one at 20€, the result will be 10.

DDL

Data Definition Language

CREATE

```
CREATE TABLE test (
    tId char(7) NOT NULL CONSTRAINT idPK PRIMARY KEY,
    tName varchar(14) NOT NULL,
    tAge int DEFAULT 18 NOT NULL,
    tWeight decimal(4, 1) NULL,
    CONSTRAINT tAgeC CHECK(tAge > 16 AND tAge < 140),
    CONSTRAINT tWeightC CHECK(tWeight BETWEEN 20 AND 300)
    );
```

> - **`char(7)`**
>   7 characters (no more no less).
> - **`varchar(14)`**
>   Maximum 14 characters.
> - **`decimal(4, 1)`**
>   Decimal number with 4 digits and only one after coma.
>   *(From `0.0` to `999.9` in this case)*
>   This has `NULL` parameter, so `tWeight` is **optional**.
> - **`DEFAULT`**
>   This will set the default value on creation of an attribute.

DROP (delete table)

```
DROP TABLE test;
```

> - `DROP TABLE`
>   This request will delete the table.