Constraints

Constraints are rules that we apply to our data when we're creating a table in a database. They help us make sure that the data we put into our tables is accurate and reliable.

Constraint	Description
PRIMARY KEY	used to uniquely identify a row
FOREIGN KEY	references a row in another table
UNIQUE	values cannot match any existing value in the same table
NOT NULL	values cannot be null, they always have to contain data
CHECK	validates a condition for new values being entered into the database
DEFAULT	sets a default value if no value is passed

Join

This is used to join two tables together, creating a new one. There are three main types of join.
`INNER JOIN`- Joins both tables, only with the matching data.
`LEFT/RIGHT JOIN` - Joins matching data and the rest of the data from one of the tables (left or right).
`FULL OUTER JOIN` - Joins the two tables completely.
After the join statement you should state the columns that are the same on both tables with the `ON` keyword.

Example:

```
SELECT table_name.column_name1, …
FROM table_name
INNER JOIN another_table
ON table_name.column_name1=another_table.another_column;
```

Group by

The GROUP BY keyword groups rows that have a common value.

```
SELECT column FROM table GROUP BY comlumn_name;
```

Functions

A function has an action to perform, a parameter between parenthesis, and then it returns a value.
Although we use it in combination with group by, the functions **must be used after the select statement.**

```
SELECT column FUNC(column_name) FROM table GROUP BY comlumn_name;
```

A list of the most used functions

FUNCTION	RETURNS
COUNT(column_name)	counts the amount of entries in a group
MAX(column_name)	FindS the maximum value for a certain column within this group.
MIN(column_name)	Finds the minimum value for a certain column within this group
SUM(column_name)	Calculates the sum of all values for a certain column within this group.
AVG(column_name)	Calculates the average of all values for a certain column within this group.

Alias

Used to rename columns or tables within a statement
Columns: use AS between name and alas
Tables: write the alias right after the name

```
SELECT AVG(column_name) AS average_nameç
FROM table_name_strangely_long_and absurd shortname;
```

The example above is just to show the syntax, as aliases are not used again at the statement. You can use aliases even before declaration them, as long as they are in the same statement. In the example below “s” is the alias for the “student” table, and “average_score” the alias for “AVG(score)”

```
SELECT s.name, AVG(score) as average_score
FROM student_record
JOIN student s ON student_id = s.id
GROUP BY student_id
HAVING average_score < 55;
```

Having

Used after **group by** to filter on a condition.
IT’s not the same as **where.**The WHERE clause is used to filter rows based on a specific condition, whereas the HAVING clause is used to filter the results of an aggregation based on a specific condition.

```
SELECT column_1, column_2, aggregate_function(column3)
FROM table_name
GROUP BY coumn_1
HAVING search_condition;
```