SQL Basics

Introduction

Welcome to the SQL for Beginners course! At the end of this course, you will have the tools to fetch, manipulate, and present meaningful data from databases. This course aims to give you the basics of extracting knowledge from large amounts of data. SQL (which stands for "Structured Query Language") is used to manage data from a Relational Database Management System (RDBMS). This means you will learn how to manage data arranged in a tabular format.

Read More

What is a database

Databases are like large buckets that store data in an organized manner. A few examples of when we would like to create a database:

- A database for a university to save data about students, courses, and lecturers.
- A database for a car agency to track sales, car storage, and workers.
- A database for a hospital to save information about patient's history to provide good health care
- And many more

Inside a database there are tables, and each table has a name, column names, and rows. For example, this is a `workers` table:

	firstname	lastname	age	exp_years	gender
1	Ghully	Thuas	29	2.3	Female
2	Bostal	Shkolky	32	0.2	Male
3	Qaostu	Malop	21	4	Female

The `workers` table has 5 columns and 3 rows. We don't need any tool to know that we have 3 workers and it is easy to calculate the average age of all of them `(29 + 32 + 21) / 3` but what happens when we have a thousand rows or even a million rows?

For that, we have databases and the SQL language. database stores all of the tables and SQL extracts the data.

Database concepts

Let's define a few concepts

- rows are referred to as **records**
- columns are referred to as **fields**

The number of records in a database is unlimited, but the number of fields is limited to the number of fields that were created for the table. Field naming is important because we use it when writing SQL queries. Field names should be lowercase, without spaces, and singular because the field refers to a single record. For example:

- full_name instead of FullNames
- firstname instead of firstName
- salary instead of salaries

It is not allowed to name two different fields with the same name, as this can lead to confusion and difficulty in distinguishing between a table name and a field. To identify different records, it is best to use a unique identifier called a "key". Every table should have a key field, as it allows us to distinguish between two similar records. This key field is usually referred to as the "id".

Unique values

Let's assume we have the following table

sales

	country	city	amount
1	Poland	Warsaw	13
2	Germany	Berlin	24
3	Poland	Katowice	56

And we would like to know all of the countries that the product was sold.

If we use the normal query we know: `SELECT country from sales` it will return `Poland Germany Poland`. This is not what we are looking for because Poland is repeated twice.

To solve it we can use the `DISTINCT` keyword:

```
SELECT DISTINCT country FROM sales
```

Conditional statements part 1

Sometimes we would like to fetch records that meet a certain condition.

For example

- fetch all of the records that have the family name "Aothly"
- fetch all of the records that the amount is bigger than 5
- fetch all of the records with the country "Mexico"

To add condition we can use the `WHERE` keyword

sales

coin	amount
AGK	13
GOL	21
KLA	15
AGK	18

To fetch all of the records with the coin "AGK" we will write:

```
SELECT * FROM sales WHERE coin = "AGK"
```

To fetch all of the records with amount smaller than 20 we will write:

```
SELECT * FROM sales WHERE amount < 20
```

Conditional statements part 2

Creating a query with only one condition is not sufficient. Sometimes we would like to check something more complicated. For that SQL (and many other programming languages) have the `AND`, `OR`, and `NOT` keywords to increase our ability to fetch the right result we need.

The `AND` and `OR` keywords are used like this:

```
SELECT col1, col2
FROM table1
WHERE condition1 AND condition2 OR condition3 ...
```

We can stack as many conditions as we want together.

---

people

name	age	gender
Joas	13	male
Holwa	17	male
Nohlas	24	female
Polar	23	male
Loopa	18	female

The `AND` keyword means that **both** conditions must be true; if either of them is not, then the condition will not be met.

For example, if we will write

```
SELECT *
FROM people
```

```
WHERE gender = "female" AND age < 20
```

It means that we are looking for all records that the gender is "female" and the age is less than 20.

This will be the result:

name	age	gender
Loopa	18	female

The `OR` keyword means that we want one of the conditions will be true.

For example, if we take the same example from above and change the `AND` keyword to `OR`

```
SELECT *
FROM people
WHERE gender = "female" OR age < 20
```

It means that we are looking for all records that either the gender is female or the age is less than 20.

This will be the result:

name	age	gender
Joas	13	male
Holwa	17	male
Nohlas	24	female
Loopa	18	female

The `NOT` keywords mean that we don't want the condition to be met.

For example, if we write:

```
SELECT *
FROM people
WHERE NOT gender = "male"
```

This will be the result:

name	age	gender
Nohlas	24	female
Loopa	18	female

Conditional statements part 3

Conditions are booleans. Boolean is a data type with two possible values: `TRUE` or `FALSE`.

For example

- `10 > 100` - FALSE
- `10 > 5` - TRUE
- `10 > 5 AND 100 < 5` - FALSE

Boolean columns have only two values - either 1 or 0. `TRUE` indicates 1 and `FALSE` indicates 0

We can replace columns such as employed or unemployed to `1` or `0` to make it easier to filter data. To filter data using booleans we will use the `IS TRUE` or `IS NOT TRUE` keywords.

```
SELECT *
FROM table1
WHERE col1 IS NOT FALSE AND col2 IS TRUE
```