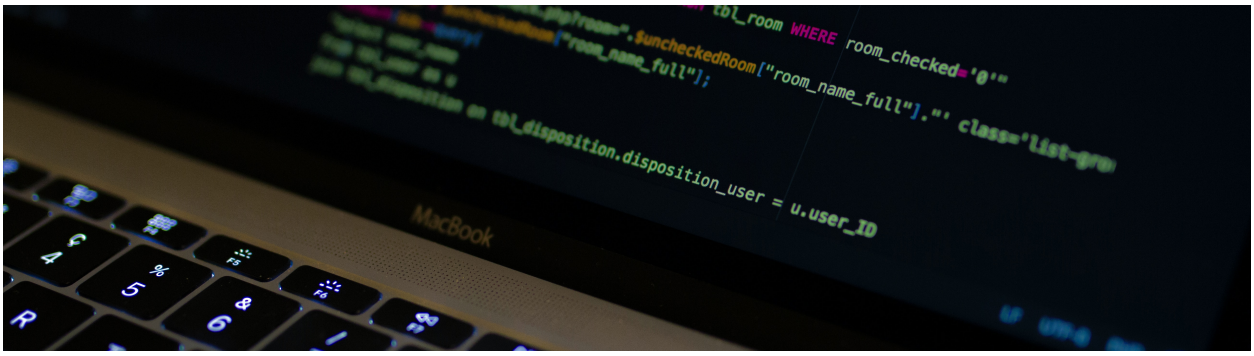


SQL

What is a database?

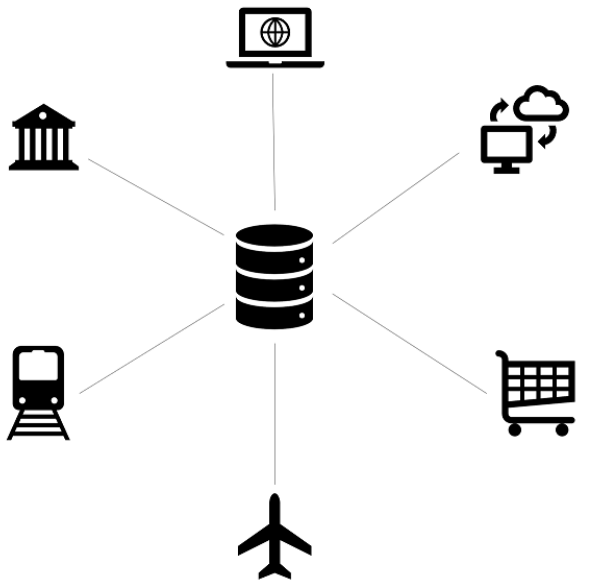
In simple terms, a database is a collection of data stored in a computer system. Here are other definitions of a database: According to **Wikipedia**:A database is an organized collection of data, generally stored and accessed electronically from a computer system.According to **Oracle**:A database is an organized collection of structured information, or data, typically stored electronically in a computer system.People use data and databases with or without awareness in their daily life activities.

How are databases used in the real-world?

Databases are almost everywhere. Your bank, your grocery store, an app on your cellphone, websites all use databases to keep track of your data. When you access a website, the website starts to collect your data (e.g. accessing date and time, your location, your browser info) and store it in its database.

Let's take another example. When you order a product on a commercial website, your order is stored in a database. You withdrew money from your bank account. Your bank stores this transaction in the database. Social media platforms such as Facebook, Instagram, Twitter use databases to store data like members, their friends, member activities, messages, advertisements, etc.

(Note: In diagrams, databases are represented as a cylinder shape.)



Category	Usage
Banking&Finance	Customer information, accounts, transactions
Education	Student information, course registrations, grades

Category	Usage
Telecommunication	Internet&phone usage, subscriber information
Human resources	Employees, managers, salaries, hire&termination dates
Websites	Products, visitors, website traffic statistics
Transportation	Passenger, reservation and schedule information

A database is typically controlled by a **database management system (DBMS)**

. Data and DBMS along with the applications that are associated with them are called a database system, often shortened to just database.

> Q: What is a Database?A: A database is an organized collection of data, generally stored and accessed electronically from a computer system. In simple terms, a database is a collection of data stored in a computer system. When you order a product on a commercial website, your order is stored in a database. You withdrew money from your bank account. Your bank stores this transaction in the database. Social media platforms such as Facebook, Instagram, Twitter use databases to store data like members, their friends, member activities, messages, advertisements, etc.

What is in a database?

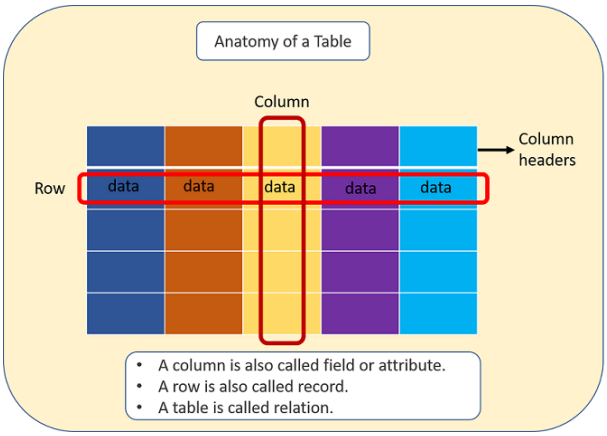
The information inside the database is grouped into tables. A table in a database is called a *database table*. Tables are the basic unit of data storage in databases. We talked about the definition of the database in the previous lesson. We used the term **structured data.** Structured data here means table. A table consists of columns and rows. You may think of it as an Excel or Google spreadsheet. Although there are similarities between the database table and Excel/Google spreadsheet, they are different things.

Anatomy of a Table

A table is made up of columns and rows. A column is a piece of data stored by the table. A row is a single set of columns that describe the attributes of a single thing. Columns should have a unique name. Columns and rows together make up a table.

Tips:

- In database world;
- A column is also called a field or attribute,
- A row is also called a record or a tuple,
- A table is also called a relation.



A database can consist of one or more tables. In most cases, more than one table. Each table has a unique name, such as employees, departments, or customers, etc.

Example

Let's take the case of a company database. Suppose that it has two tables. One is employees, and other is departments. Take a closer look at the employees table below. Here is the breakdown of the table.

- Table's name is *employees.*
- The table has seven columns (aka, fields or attributes)
- The table has ten rows (aka, records or tuples)
- Table's column header names are: emp_id, first_name, last_name, salary, job_title, gender, hire_date
- Inside the employees' table, there is data about each employee in the company

emp_id	first_name	last_name	salary	job_title	gender	hire_date
26650	Elvis	Ritter	86000	Sales Manager	Male	11/24/2017
70950	Rodney	Weaver	87000	Project Manager	Male	12/20/2018
97927	Billie	Lanning	67000	Web Developer	Female	6/25/2018
67323	Lisa	Wiener	75000	Business Analyst	Female	8/9/2018
17679	Robert	Gilmore	110000	Operations Director	Male	9/4/2018
76589	Jason	Christian	99000	Project Manager	Male	1/21/2019
51821	Linda	Foster	95000	Data Scientist	Female	4/29/2019
71329	Gayle	Meyer	77000	HR Manager	Female	6/28/2019
49714	Hugo	Forester	55000	IT Support Specialist	Male	11/22/2019
30840	David	Barrow	85000	Data Scientist	Male	12/2/2019

Of course, a real-world company would have many more employees. We use a small table to illustrate concepts.

> Q: What is a table, column and row?A: A table is an organized collection of data stored in the form of columns and rows. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

Type of Databases

Typically, there are two main database storage types:

- **Relational Database - SQL**
- **Non-Relational Database - NoSQL**

A *relational database* is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the *key*. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

Actually, the term "relational database" was invented by E. F. Codd at IBM in 1970. Codd introduced the term in his research paper "A Relational Model of Data for Large Shared Data Banks". In this paper and later papers, he defined what he meant by "relational". One well-known definition of what constitutes a relational database system is composed of Codd's 12 rules. However, no commercial implementations of the relational model conform to all of Codd's rules, so the term has gradually come to describe a broader class of database systems, which at a minimum:

- Present the data to the user as relations (a presentation in tabular form, i.e. as a *collection* of tables with each table consisting of a set of rows and columns);
- Provide relational operators to manipulate the data in tabular form.

A software system used to maintain relational databases is called a *Relational Database Management System (RDBMS).* Here are some examples of RDBMS:

- Amazon Aurora
- Amazon RDS
- Microsoft SQL Server
- Oracle Database
- MySQL
- IBM DB2
- Maria DB
- PostgreSQL
- SQLite

SQL (stands for Structured Query Language) is accepted as the standard Relational Database Management System (RDBMS) language. So we usually prefer to call Relational Database as SQL and Non-Relational database as NoSQL.

Tips:

- SQLite is a relational database management system contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. (Wikipedia)

Structured Query Language (SQL)

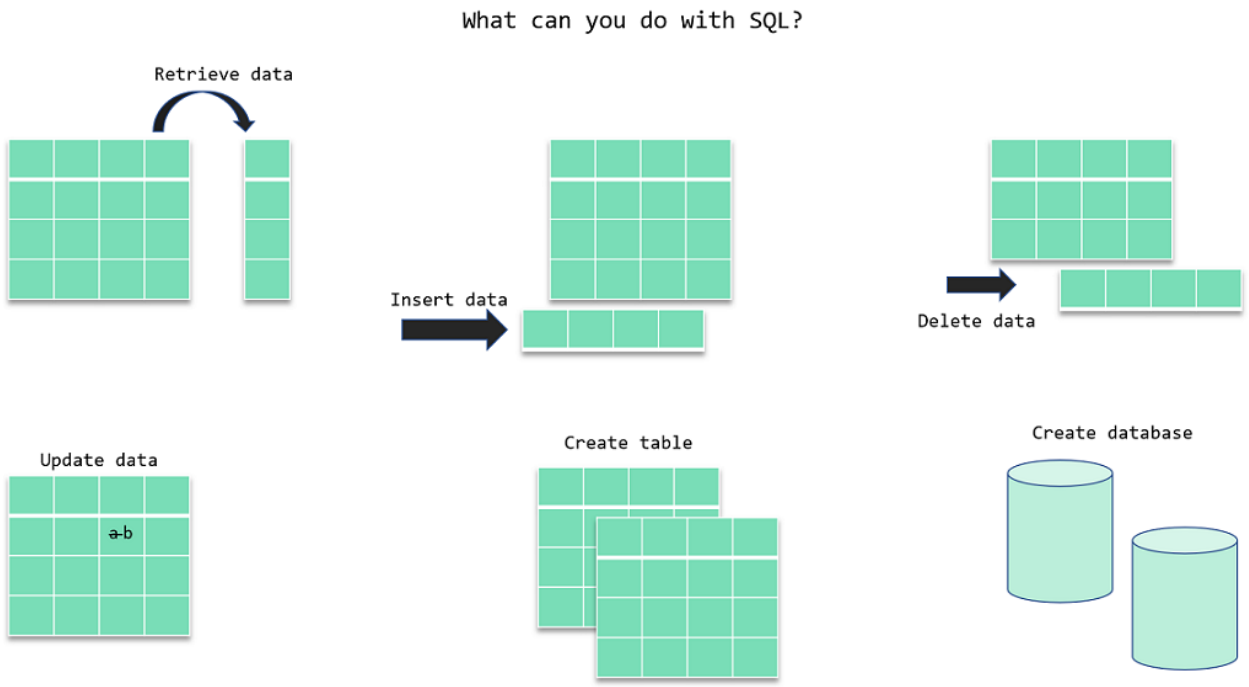
SQL stands for Structured Query Language and used to communicate with relational databases. SQL is a declarative language, not a procedural language. You write a single SQL declaration and hand it to the DBMS. The DBMS then executes internal code, which is hidden from us.

- Declarative paradigm is where you say what you want without having to say how to do it. With procedural paradigm (used in JAVA, C), you have to specify exact steps to get the result. SQL is declarative than procedural since the queries don't specify steps to produce the result.
- SQL in it's purest form is not a programming language, but a query language. Because, it needs to be able to perform loops and control structures. However, with some extensions, SQL can have looping and control structures but they exist outside or rather as an appendage to the original SQL spec. In this manner, some argue that SQL is a programming language.

Most commercial database systems employ the SQL language. With SQL, you can access or manipulate data stored in the database. There are different types of access. These are:

- Retrieval of data from the database
- Insertion of new data into the database
- Updating the data in the database
- Deletion of data from the database

Besides, you can create new databases and tables using SQL.



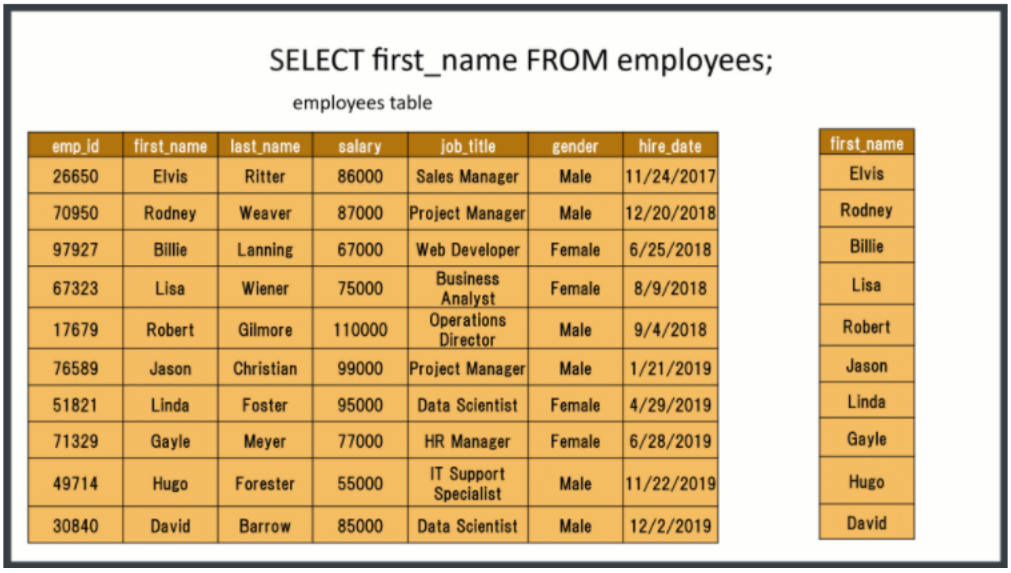
A **query** is a statement asking for the retrieval of information from the database.

> Q: What is SQL?
>
> A: SQL stands for Structured Query Language and used to communicate with a database. With SQL, you can access or manipulate data stored in the database.

SQL language structure is easy to understand. It looks like plain English. Here is an example SQL statement:

```
SELECT first_name FROM employees;
```

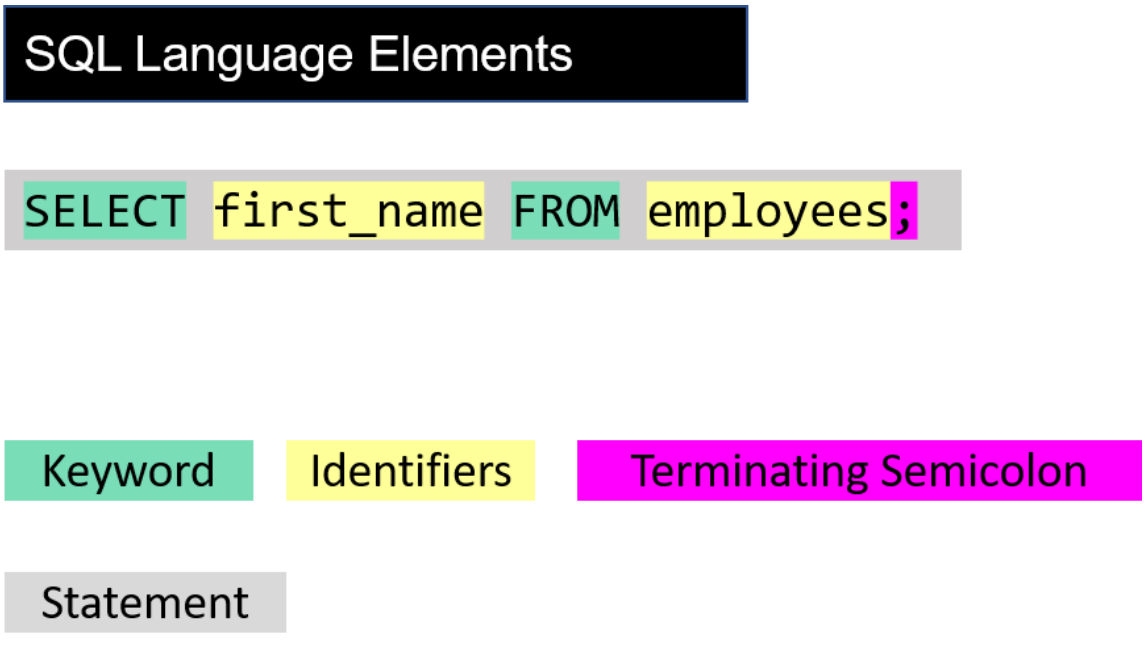
If we apply this SQL statement to our *employees* table, we get a single column that is *first_name*.



emp_id	first_name	last_name	salary	job_title	gender	hire_date
26650	Elvis	Ritter	86000	Sales Manager	Male	11/24/2017
70950	Rodney	Weaver	87000	Project Manager	Male	12/20/2018
97927	Billie	Lanning	67000	Web Developer	Female	6/25/2018
67323	Lisa	Wiener	75000	Business Analyst	Female	8/9/2018
17679	Robert	Gilmore	110000	Operations Director	Male	9/4/2018
76589	Jason	Christian	99000	Project Manager	Male	1/21/2019
51821	Linda	Foster	95000	Data Scientist	Female	4/29/2019
71329	Gayle	Meyer	77000	HR Manager	Female	6/28/2019
49714	Hugo	Forester	55000	IT Support Specialist	Male	11/22/2019
30840	David	Barrow	85000	Data Scientist	Male	12/2/2019

first_name
Elvis
Rodney
Billie
Lisa
Robert
Jason
Linda
Gayle
Hugo
David

The whole command `SELECT first_name FROM employees;`
is called **statement.**
Let's breakdown of this statement. Herein SELECT, FROM words are keywords. They are special commands for SQL. first_name, employees are identifiers. SQL statements end with a semicolon (;). SQL Language Elements is also called SQL Syntax.



SELECT Statement

- The `SELECT` statement is used to select data from a database.
- You can retrieve rows from the columns of the table by using this statement.

- `SELECT` statement is used with `FROM` keyword.
- The syntax of the `SELECT` statement can be seen below.

```
SELECT column_name(s) FROM table_name;
```

Basic Syntax

- SQL statements start with a keyword like `SELECT`, `INSERT`, `UPDATE`, `DELETE`, etc. and all the statements end with a semicolon (;).
- The semicolon at the end indicates that the statement is completed and ready to be executed.
- SQL is also case insensitive, which means you can use both `SELECT` and `select` in your query. They mean the same thing for SQL.
- Writing SQL commands in the upper-case is the most common and preferred style. But, you can write the same query in both ways
- White spaces and empty lines are ignored in SQL... Boşluk bırakırsak sonuç değişmiyor

Selecting Multiple Columns

You use commas between column names for multiple columns

You use * to retrieve all the columns

```
SELECT column1, column2 FROM table1;
SELECT * FROM table1;
```

DISTINCT Clause

Columns in the tables may often contain some duplicate values, but you may only need the distinct values as a result. Here comes the `SELECT` statement with the `DISTINCT` clause.

ⓘ **Info:** We have learned what the statement and keyword are. Here is another term which we introduced to you in this lesson: Clause. We want to define each three terms to help you gain better understanding. We will use query

```
SELECT first_name, last_name, gender FROM employees;
```

as an example to explain the concepts.

- **Keyword:** These are the individual elements which are predefined. In the example these are SELECT and FROM seperately.
- **Clause:** It's a part of a SQL statement. In our example, these ere `SELECT first name, last name, gender` and `FROM employees`.
- **Statement:** The complete query is a statement. A statement may consist of two or more clauses.

The `SELECT DISTINCT` s used to return only distinct (different/unique) values to eliminate duplicate rows in a result set. Here is the syntax of the `DISTINCT` clause:

When all the rows in that column have unique values or in other words if there are no duplicated rows in a column, `SELECT` and `SELECT DISTINCT` gives the same result.

Q: What are some common clauses used with SELECT query in SQL?

A: WHERE clause, ORDER BY clause, GROUP BY clause and HAVING clause

By using this query, you explain to SQL that you only want to see the unique/distinct data from the column/columns in the given table.

WHERE & LIMIT Clauses

The WHERE clause is used to filter records.

- It allows you to define a specific search condition for the result set returned by a query.

- So, the result set only consists of the records that fulfill the predefined condition(s).

The WHERE clause is mostly used with the SELECT statement. In addition to the SELECT statement, it may also be used with some other statements like DELETE and UPDATE

It's used in a query after the FROM clause as in the below example.

```
SELECT column_name(s) FROM table_name WHERE condition(s);
```

By using this query, you explain to SQL that you only want to get the data that pass the defined condition(s) as a result set.

WHERE Clause - Operators

You can use the following operators in the WHERE clause.

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. This operator may be written as != in some versions of SQL
BETWEEN	Test if a value is between a certain range of values
LIKE	Determine if a character string matches a predefined pattern
IN	Test whether or a value matches any value in a list

student	lesson	grade
Student1	Mathematics	95
Student2	Literature	60
Student3	Mathematics	45
Student4	Chemistry	85
Student5	Physics	70
Student6	Physics	75
Student7	Mathematics	75

Assume that;

You have a table named student_table as above.

If we want to select only the records of which grade is higher than 70 in the result set, then we should write a query like this.

```
SELECT * FROM student_table WHERE grade > 70
After the execution of the query, you will get a result set like below.
output:
student     lesson       grade
----------  -----------  ----------
Student1    Mathematics  95
```

```
Student4    Chemistry    85
Student6    Physics      75
Student7    Mathematics  75
```

• You want to see only the records of which lesson is Mathematics in the result set.

```
SELECT * FROM student_table WHERE lesson = "Mathematics";
student     lesson       grade
----------  -----------  ----------
Student1    Mathematics  95
Student3    Mathematics  45
Student7    Mathematics  75
```

LIMIT Clause

In this lesson, you will focus on the `LIMIT` clause in SQL. The `LIMIT` clause is used to filter records. It constrains the number of rows returned by a query. Assume that your query returns one thousand rows. But you only want to see the first 10 rows in the result set. In such cases, we use `LIMIT` clause to obtain the desired output. Here is the syntax of the `LIMIT` clause.

```
SELECT column_name(s) FROM table_name LIMIT number_rows;
```

Let's select all the columns of the student_table and return the first 3 rows.

```
SELECT * FROM student_table LIMIT 3;

Output:
student     lesson       grade
----------  -----------  ----------
Student1    Mathematics  95
Student2    Literature   65
Student3    Mathematics  45
```

We can also combine `LIMIT` with `WHERE` . `LIMIT` clause is placed after the `WHERE` clause. Let's select the students whose grade is higher than 70 and let our query return the first 2 rows.

```
SELECT * FROM student_table WHERE grade > 70 LIMIT 2;

Output:
student     lesson       grade
----------  -----------  ----------
Student1    Mathematics  95
Student4    Chemistry    85

SELECT * FROM student_table WHERE lesson = "Mathematics" LIMIT 1;
Output:
student     lesson       grade
----------  -----------  ----------
Student1    Mathematics  95
```

Order By Clause

SELECT statement returns records in an unspecified order. In case you want to retrieve data in alphabetical or numeric order, we use `ORDER BY` keyword.

The `ORDER BY` keyword sorts the result-set in descending or ascending order.

By default `ORDER BY` keyword sorts the records in ascending order. Use the keyword `DESC` to sort the records in descending order. You can also use `ASC` to sort the data in ascending order. You have to use either of them.

Here is the syntax of `ORDER BY:` Herein "|" symbol means "use either `ASC` or `DESC` ". If you don't use any of them, the default value is ASC (ascending order).

```
SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC|DESC;
```

emp_id	first_name	last_name	salary	job_title	gender	hire_date
26650	Elvis	Ritter	86000	Sales Manager	Male	11/24/2017
70950	Rodney	Weaver	87000	Project Manager	Male	12/20/2018
97927	Billie	Lanning	67000	Web Developer	Female	6/25/2018
67323	Lisa	Wiener	75000	Business Analyst	Female	8/9/2018
17679	Robert	Gilmore	110000	Operations Director	Male	9/4/2018
76589	Jason	Christian	99000	Project Manager	Male	1/21/2019
51821	Linda	Foster	95000	Data Scientist	Female	4/29/2019
71329	Gayle	Meyer	77000	HR Manager	Female	6/28/2019
49714	Hugo	Forester	55000	IT Support Specialist	Male	11/22/2019
30840	David	Barrow	85000	Data Scientist	Male	12/2/2019

Here is our employees table. I want to sort the *first_name* column in alphabetical order (A-Z). This is the appropriate query:

```
SELECT * FROM employees ORDER BY first_name ASC;

emp_id      first_name  last_name   salary      job_title      gender      hire_date
----------  ----------  ----------  ----------  -------------  ----------  ----------
97927       Billie      Lanning     67000       Web Developer  Female      6/25/2018
30840       David       Barrow      85000       Data Scientis  Male        12/2/2019
26650       Elvis       Ritter      86000       Sales Manager  Male        11/24/2017
71329       Gayle       Meyer       77000       HR Manager     Female      6/28/2019
49714       Hugo        Forester    55000       IT Support Sp  Male        11/22/2019
76589       Jason       Christian   99000       Project Manag  Male        1/21/2019
51821       Linda       Foster      95000       Data Scientis  Female      4/29/2019
67323       Lisa        Wiener      75000       Business Anal  Female      8/9/2018
17679       Robert      Gilmore     110000      Operations Di  Male        9/4/2018
70950       Rodney      Weaver      87000       Project Manag  Male        12/20/2018

SELECT * FROM employees ORDER BY first_name DESC;
emp_id      first_name  last_name   salary      job_title        gender      hire_date
----------  ----------  ----------  ----------  ---------------  ----------  ----------
70950       Rodney      Weaver      87000       Project Manager  Male        12/20/2018
17679       Robert      Gilmore     110000      Operations Dire  Male        9/4/2018
67323       Lisa        Wiener      75000       Business Analys  Female      8/9/2018
51821       Linda       Foster      95000       Data Scientist   Female      4/29/2019
76589       Jason       Christian   99000       Project Manager  Male        1/21/2019
49714       Hugo        Forester    55000       IT Support Spec  Male        11/22/2019
71329       Gayle       Meyer       77000       HR Manager       Female      6/28/2019
26650       Elvis       Ritter      86000       Sales Manager    Male        11/24/2017
30840       David       Barrow      85000       Data Scientist   Male        12/2/2019
97927       Billie      Lanning     67000       Web Developer    Female      6/25/2018
```

After executing the query, we get the result table below. Our table is now sorted by the first names in ascending order. Not only the *first_name* column is sorted, but also other columns are affected by the sort accordingly.

💡 **Tip:** When you sort the data, the original table's order is not affected. Remember from the previous lessons that a query returns a result table. Thus, we sort the result table, not the original one.

Until now, we've sorted the column containing textual data. It's time to sort numerical data in our table. You may wonder whose salary is the highest. Let's write the query. This time we don't want to retrieve all columns instead we want first name, last name and salary.

```
SELECT first_name, last_name, salary FROM employees ORDER BY salary DESC;
first_name  last_name   salary
----------  ----------  ----------
Robert      Gilmore     110000
Jason       Christian   99000
Linda       Foster      95000
Rodney      Weaver      87000
Elvis       Ritter      86000
David       Barrow      85000
Gayle       Meyer       77000
Lisa        Wiener      75000
```

```
Billie      Lanning     67000
Hugo        Forester    55000
```

Sorting By Multiple Columns

We are now able to sort by one column using the **ORDER BY** keyword. In some cases, we may need to sort our data by two columns or more. To do this, separate the columns by a comma. Here is the **syntax:**

```
SELECT column_name(s) FROM table_name ORDER BY column1 ASC|DESC, column2 ASC|DESC, columnN ASC|DESC;
```

```
SELECT * FROM employees ORDER BY gender DESC, first_name ASC;

emp_id      first_name  last_name   salary      job_title       gender      hire_date
----------  ----------  ----------  ----------  --------------  ----------  ----------
30840       David       Barrow      85000       Data Scientist  Male        12/2/2019
26650       Elvis       Ritter      86000       Sales Manager   Male        11/24/2017
49714       Hugo        Forester    55000       IT Support Spe  Male        11/22/2019
76589       Jason       Christian   99000       Project Manage  Male        1/21/2019
17679       Robert      Gilmore     110000      Operations Dir  Male        9/4/2018
70950       Rodney      Weaver      87000       Project Manage  Male        12/20/2018
97927       Billie      Lanning     67000       Web Developer   Female      6/25/2018
71329       Gayle       Meyer       77000       HR Manager      Female      6/28/2019
51821       Linda       Foster      95000       Data Scientist  Female      4/29/2019
67323       Lisa        Wiener      75000       Business Analy  Female      8/9/2018
```

ORDER BY Clause with WHERE Clause

In this part, we will use ORDER BY with the WHERE clause.

This is the syntax:

```
SELECT column_name(s) FROM table_name WHERE condition ORDER BY column_name(s)s ASC|DESC;
```

ORDER BY clause is placed after the WHERE clause.

Technically, any SQL statement can be written on a single line. However, it will become difficult to read when you start to write long queries. The solution in such cases is to organize the code, not just *horizontally*, but also *vertically*. This is called **beautifying.** Let's rewrite the syntax above.

```
SELECT column_name(s)
FROM table_name
WHERE condition
ORDER BY column_name(s)s ASC|DESC;
```

emp_id	first_name	last_name	salary	job_title	gender	hire_date
26650	Elvis	Ritter	86000	Sales Manager	Male	11/24/2017
70950	Rodney	Weaver	87000	Project Manager	Male	12/20/2018
97927	Billie	Lanning	67000	Web Developer	Female	6/25/2018
67323	Lisa	Wiener	75000	Business Analyst	Female	8/9/2018
17679	Robert	Gilmore	110000	Operations Director	Male	9/4/2018
76589	Jason	Christian	99000	Project Manager	Male	1/21/2019
51821	Linda	Foster	95000	Data Scientist	Female	4/29/2019
71329	Gayle	Meyer	77000	HR Manager	Female	6/28/2019
49714	Hugo	Forester	55000	IT Support Specialist	Male	11/22/2019
30840	David	Barrow	85000	Data Scientist	Male	12/2/2019

Assume that we try to find the employees whose salary is higher than $80,000. Next, we will sort it by first_name in descending order.

Here is the query:

```
SELECT *
FROM employees
WHERE salary > 80000
ORDER BY first_name DESC;

emp_id      first_name  last_name   salary      job_title        gender      hire_date
----------  ----------  ----------  ----------  ---------------  ----------  ----------
70950       Rodney      Weaver      87000       Project Manager  Male        2018-12-20
17679       Robert      Gilmore     110000      Operations Dire  Male        2018-09-04
51821       Linda       Foster      95000       Data Scientist   Female      2019-04-29
76589       Jason       Christian   99000       Project Manager  Male        2019-01-21
26650       Elvis       Ritter      86000       Sales Manager    Male        2017-11-24
30840       David       Barrow      85000       Data Scientist   Male        2019-12-02
```

We first returned the employees whose salary is higher than $80,000. Next, we sorted this by the first names in descending order.

AND, OR & NOT Operators

In SQL, **AND, OR & NOT** keywords are called operators. In particular, they are called logical operators. Their purposes are filtering the data based on conditions.

The `WHERE` clause can be combined with AND, OR & NOT operators. Let's start with the `AND` operator.

AND Operator

The `AND` operator is used with the `WHERE` clause and combines multiple expressions. It returns only those records where both conditions (in `WHERE` clause) evaluate to `True`. The syntax has the following form in the `WHERE` condition:

```
WHERE left_conditon AND right_condition
```

Now, display the employees whose title is a data scientist and gender is male. You would be asked for the same thing as "show me the male data scientists in the company. " They are both the same.

emp_id	first_name	last_name	salary	job_title	gender	hire_date
26650	Elvis	Ritter	86000	Sales Manager	Male	11/24/2017
70950	Rodney	Weaver	87000	Project Manager	Male	12/20/2018
97927	Billie	Lanning	67000	Web Developer	Female	6/25/2018
67323	Lisa	Wiener	75000	Business Analyst	Female	8/9/2018
17679	Robert	Gilmore	110000	Operations Director	Male	9/4/2018
76589	Jason	Christian	99000	Project Manager	Male	1/21/2019
51821	Linda	Foster	95000	Data Scientist	Female	4/29/2019
71329	Gayle	Meyer	77000	HR Manager	Female	6/28/2019
49714	Hugo	Forester	55000	IT Support Specialist	Male	11/22/2019
30840	David	Barrow	85000	Data Scientist	Male	12/2/2019

There are two conditions here. One is the title of the employee is a data scientist, other is the gender of his/her should be male. The correct search condition in where clause is `job_title = 'data scientist' AND gender = 'Male'.`

Let's write the query.

query :

```
SELECT * FROM employees WHERE job_title = "Data Scientist" AND gender = "Male";

emp_id      first_name  last_name   salary      job_title       gender      hire_date
----------  ----------  ----------  ----------  --------------  ----------  ----------
30840       David       Barrow      85000       Data Scientist  Male        2019-12-02
```

OR Operator

The `OR` operator is used with the `WHERE` clause and combines multiple expressions. It displays the record where either one of conditions (in WHERE clause) evaluates to `True`. The syntax has the following form in the `WHERE` condition.

```
WHERE first_condition OR second_condition
```

Display the employees whose title is a data scientist or gender is male.

```
SELECT *
FROM employees
WHERE job_title = 'Data Scientist' OR gender = 'Male';

emp_id      first_name  last_name   salary      job_title            gender      hire_date
----------  ----------  ----------  ----------  -------------------  ----------  ----------
17679       Robert      Gilmore     110000      Operations Director  Male        2018-09-04
26650       Elvis       Ritter      86000       Sales Manager        Male        2017-11-24
30840       David       Barrow      85000       Data Scientist       Male        2019-12-02
49714       Hugo        Forester    55000       IT Support Speciali  Male        2019-11-22
51821       Linda       Foster      95000       Data Scientist       Female      2019-04-29
70950       Rodney      Weaver      87000       Project Manager      Male        2018-12-20
76589       Jason       Christian   99000       Project Manager      Male        2019-01-21
```

💡 **Tip:** Don't get confused with ANDs and ORs!

- When you want **ALL** of your conditions to be true, use **AND**
- When you want **ANY** of your conditions to be true, use **OR**

NOT Operator

The `NOT` operator is used to negate a condition in the `WHERE` clause. `NOT` is placed right after `WHERE` keyword. You can use it with AND & OR operators. Here is the syntax of `NOT` operator.

```
WHERE NOT first_condition
```

Display the male employees.

```
SELECT * FROM employees WHERE NOT gender = 'Female';

emp_id      first_name  last_name   salary      job_title            gender      hire_date
----------  ----------  ----------  ----------  -------------------  ----------  ----------
17679       Robert      Gilmore     110000      Operations Director  Male        2018-09-04
26650       Elvis       Ritter      86000       Sales Manager        Male        2017-11-24
30840       David       Barrow      85000       Data Scientist       Male        2019-12-02
49714       Hugo        Forester    55000       IT Support Speciali  Male        2019-11-22
70950       Rodney      Weaver      87000       Project Manager      Male        2018-12-20
76589       Jason       Christian   99000       Project Manager      Male        2019-01-21
```

BETWEEN OPERATOR

The `BETWEEN` operator is used for comparison in `WHERE` clauses. It's a comparison operator. You can use it to test if a value is in a range of values. If the value is in the specified range, the query returns all records fallen within that range.

The following displays the syntax of the `BETWEEN` operator:

```
WHERE test_expression BETWEEN low_expression AND high_expression
```

☝ **Important:** The **BETWEEN** operator is inclusive. To specify an exclusive range, use the greater than (>) and less than operators (<).

If we need to find the names of the employees with salary amounts between $80,000 and $90,000, we can use the `BETWEEN` comparison operator to write:

```
SELECT *
FROM employees
WHERE salary BETWEEN 80000 AND 90000;

emp_id      first_name  last_name   salary      job_title      gender      hire_date
----------  ----------  ----------  ----------  -------------  ----------  ----------
26650       Elvis       Ritter      86000       Sales Manager  Male        2017-11-24
30840       David       Barrow      85000       Data Scientis  Male        2019-12-02
70950       Rodney      Weaver      87000       Project Manag  Male        2018-12-20

SELECT *
FROM employees
WHERE salary >= 80000 AND salary <= 90000;

AYNI SONUCU VERİR
```

We can use `NOT BETWEEN` to negate the result of the `BETWEEN` operator. The following is the syntax:

For instance, you need to find the employees whose salary is not between $80,000 and $90,000. Here is the query:

```
SELECT *
FROM employees
WHERE salary NOT BETWEEN 80000 AND 90000;
emp_id      first_name  last_name   salary      job_title            gender      hire_date
----------  ----------  ----------  ----------  -------------------  ----------  ----------
17679       Robert      Gilmore     110000      Operations Director  Male        2018-09-04
49714       Hugo        Forester    55000       IT Support Speciali  Male        2019-11-22
51821       Linda       Foster      95000       Data Scientist       Female      2019-04-29
67323       Lisa        Wiener      75000       Business Analyst     Female      2018-08-09
71329       Gayle       Meyer       77000       HR Manager           Female      2019-06-28
76589       Jason       Christian   99000       Project Manager      Male        2019-01-21
97927       Billie      Lanning     67000       Web Developer        Female      2018-06-25

There are seven employees whose salary is not between $80,000 and $90,000.
WE COULD ALSO WRITE
SELECT *
FROM employees
WHERE salary < 80000 OR salary > 90000;
```

BETWEEN with Date Example

It's also possible to use the `BETWEEN` operator with dates.

Assume that we try to find employees who have joined the company from June 1, 2018 to March 31, 2019. We also want to sort by hire date in ascending order.

employees table

	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

☝ **Important:**
Please enclose your date values with single quote (') and use YYYY-MM-DD date format in your query.

```
SELECT * FROM employees
WHERE hire_date BETWEEN '2018-06-01' AND '2019-03-31'
ORDER BY hire_date;

emp_id      first_name  last_name   salary      job_title      gender      hire_date
----------  ----------  ----------  ----------  -------------  ----------  ----------
97927       Billie      Lanning     67000       Web Developer  Female      2018-06-25
67323       Lisa        Wiener      75000       Business Anal  Female      2018-08-09
17679       Robert      Gilmore     110000      Operations Di  Male        2018-09-04
70950       Rodney      Weaver      87000       Project Manag  Male        2018-12-20
76589       Jason       Christian   99000       Project Manag  Male        2019-01-21

There are five employees who have joined the company from June 1, 2018 to March 31, 2019.
```

```
SELECT *
FROM employees
WHERE salary NOT BETWEEN 75000 AND 95000 ORDER BY salary DESC;

emp_id      first_name  last_name   salary      job_title            gender      hire_date
----------  ----------  ----------  ----------  -------------------  ----------  ----------
17679       Robert      Gilmore     110000      Operations Director  Male        2018-09-04
76589       Jason       Christian   99000       Project Manager      Male        2019-01-21
97927       Billie      Lanning     67000       Web Developer        Female      2019-06-25
49714       Hugo        Forester    55000       IT Support Speciali  Male        2019-11-22
```

IN Operator

The `IN` operator is used to determine whether a value matches any value in a list. We use `IN` operator with `WHERE` clause. Following is the syntax of the `IN` operator:

```
WHERE column_name IN (value_list)
```

Suppose that you are building a team in your company. The team is comprised of Data Scientist and Business Analyst. You need to search the employee table to find the right candidates for your team. You may come up with this query:

```
SELECT *
FROM employees
WHERE job_title = 'Data Scientist' OR job_title = 'Business Analyst';
```

However, there is a better operator in case you try to match a value in a specified list. We may rewrite the query as follows:

```
SELECT *
FROM employees
WHERE job_title IN ('Data Scientist', 'Business Analyst');
```

The query retrieves the employees whose job title is Data Scientist or Business Analyst. Herein the value list is ('Data Scientist', 'Business Analyst'). When any value in the job title column matches one of the values in the list, the related row is returned

An Extended Value List

Suppose that you have decided to add teammates to your existing team. In addition to data scientist and business analyst, you are in need of a project manager and web developer. You modified the old query in which you used the `OR` operator.

Here is the new query:

```
SELECT *
FROM employees
WHERE job_title = 'Data Scientist'
OR
job_title = 'Business Analyst'
OR
```

```
job_title = 'Project Manager'
OR
job_title = 'Web Developer';
```

Then you thought that it's a better idea to use `IN` operator and revised the query using `IN` operator as below:

```
SELECT *
FROM employees
WHERE job_title IN ('Data Scientist', 'Business Analyst', 'Project Manager', 'Web Developer');
```

💡 **Tip:** If you have a query in which you use many `OR` operators, consider using the `IN` operator instead. This will make your query more readable.

NOT IN Operator

In this part, we are going to add the keyword `NOT` to our `IN` operator. You're building a team again. This time you decided to select the right candidates in a different way. You don't want to include Operations Director, HR Manager, and Sales Manager in the team.

employees table

	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

If we know which values we don't want to include in a list, we can use NOT keyword with IN . NOT gives you the opposite results, anything that doesn't match the list. Use NOT

just before IN operator.

Let's write the query:

```
SELECT *
FROM employees
WHERE job_title
NOT IN ('Operations Director', 'HR Manager', 'Sales Manager');

emp_id      first_name  last_name   salary      job_title             gender      hire_date
----------  ----------  ----------  ----------  --------------------  ----------  ----------
30840       David       Barrow      85000       Data Scientist        Male        2019-12-02
49714       Hugo        Forester    55000       IT Support Specialist  Male        2019-11-22
51821       Linda       Foster      95000       Data Scientist        Female      2019-04-29
67323       Lisa        Wiener      75000       Business Analyst      Female      2018-08-09
70950       Rodney      Weaver      87000       Project Manager       Male        2018-12-20
76589       Jason       Christian   99000       Project Manager       Male        2019-01-21
97927       Billie      Lanning     67000       Web Developer         Female      2018-06-25

There are seven employees whose job title is not in the Operations Director, HR Manager, Sales Manager list. Alright, we completed the IN o
```

```
SELECT *
FROM student_info
WHERE field IN ('Data Analysis','Data Science','DevOps')

student_id  first_name  last_name   gender      state       county      field       start_date
----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------
110028      Michael     Crawford    M           Virginia    Albemarle   DevOps      2019-07-19
110080      Amelia      Anderson    F           West Virgi  Webster     Data Analy  2019-04-25
```

```
110091      Richard     Morgan      M           Virginia    Prince Wil  DevOps      2019-06-28
110095      Hugo        Wallace     M           Virginia    Accomack    Data Analy  2019-06-16
120011      Oliver      Taylor      M           West Virgi  Hancock     Data Analy  2019-04-14
120033      Lucas       Parker      M           West Virgi  Wayne       Data Scien  2019-05-19
120087      Bill        Tucker      M           Virginia    Halifax     Data Analy  2019-06-16
130646      Jack        Rogers      M           Virginia    Prince Wil  Data Scien  2019-05-19
140711      Gabriel     Young       M           Virginia    Loudoun     Data Scien  2019-05-19
140741      Harry       Taylor      M           West Virgi  Kanawha     Data Analy  2019-04-09
140766      David       Collins     M           West Virgi  Monongalia  Data Analy  2019-06-14
140778      Chloe       Young       F           West Virgi  Logan       Data Scien  2019-05-19
150288      Joseph      Bennett     M           Virginia    Shenandoah  DevOps      2019-06-14
160001      Daniel      Thomas      M           West Virgi  Kanawha     Data Scien  2019-05-19
160011      Matilda     Green       F           West Virgi  Harrison    DevOps      2019-06-14
170587      Jason       Sanders     M           Virginia    Montgomery  Data Scien  2019-05-19
```

```
SELECT *
FROM student_info
WHERE field IN ('Data Analysis','Data Science','DevOps') AND state = 'Virginia';

student_id  first_name  last_name   gender      state       county      field       start_date
----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------
110028      Michael     Crawford    M           Virginia    Albemarle   DevOps      2019-07-19
110091      Richard     Morgan      M           Virginia    Prince Wil  DevOps      2019-06-28
110095      Hugo        Wallace     M           Virginia    Accomack    Data Analy  2019-06-16
120087      Bill        Tucker      M           Virginia    Halifax     Data Analy  2019-06-16
130646      Jack        Rogers      M           Virginia    Prince Wil  Data Scien  2019-05-19
140711      Gabriel     Young       M           Virginia    Loudoun     Data Scien  2019-05-19
150288      Joseph      Bennett     M           Virginia    Shenandoah  DevOps      2019-06-14
170587      Jason       Sanders     M           Virginia    Montgomery  Data Scien  2019-05-19
```