**DQL (Data Query Language)**

DQL statements are used to query the data contained within schema objects. The DQL Command's purpose is to obtain some schema relation based on the query passed to it. DQL can be defined as follows: It is a SQL statement component that allows retrieving data from the database and imposing order on it. It contains the SELECT statement. This command allows you to extract data from the database in order to perform operations on it. When a SELECT statement is executed against a table or tables, the result is compiled into a new temporary table, which is then displayed or received by the programme, i.e. a front-end.

List of DQL Commands:

SELECT: This command is used to retrieve data from a database.

**1. SELECT**

The SELECT statement is used to retrieve information from a database. The information returned is saved in a result table known as the result set.

Syntax

a. To select all columns

SELECT \* FROM *table*;

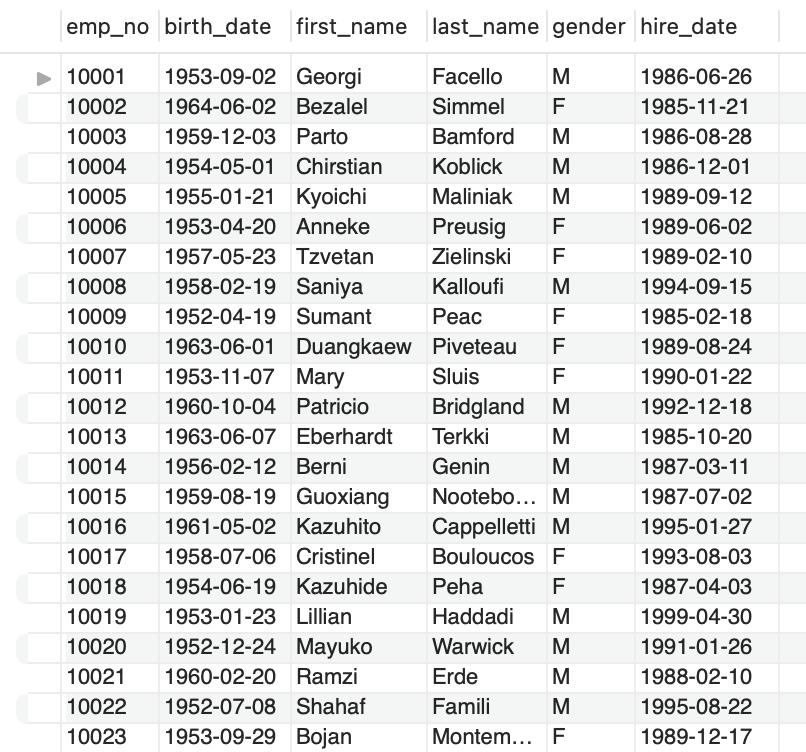
b.To select specific columns

SELECT *column1*, *column2, ...*

FROM *table*;

#### Example

SELECT \* FROM employees



SELECT first\_name FROM employees;



**2. SELECT DISTINCT Statement**

Only distinct values are returned by the SELECT DISTINCT statement. A column within a table frequently contains many duplicate values, and you may only want to list the different values.

Syntax

SELECT DISTINCT column1, column2,...

FROM table;

Example

SELECT DISTINCT gender FROM employees



**3. SQL WHERE Clause**

To filter records, use the WHERE clause. It is used to extract only records that meet a predefined condition.

Syntax

SELECT column1, column2, ...

FROM table

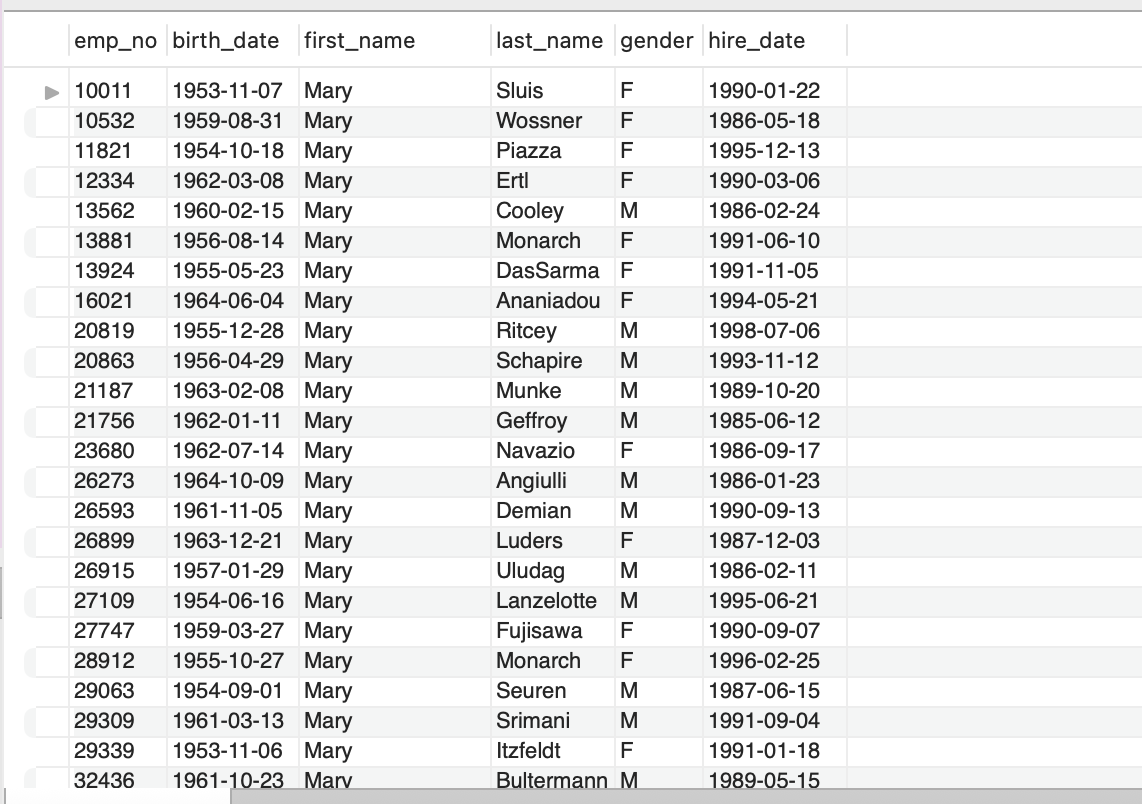
WHERE condition;

Example

SELECT \*

FROM employees

WHERE first\_name = 'Mary'



**4. AND, OR and NOT Operators**

The WHERE clause can be used in conjunction with the AND, OR, and NOT operators. To filter records based on more than one condition, use the AND and OR operators. If all of the conditions separated by AND are TRUE, the AND operator displays a record.

If any of the conditions separated by OR is TRUE, the OR operator displays a record.

If the condition(s) is NOT TRUE, the NOT operator displays a record.

AND Syntax

SELECT *column1*, *column2, ...*

FROM *table*

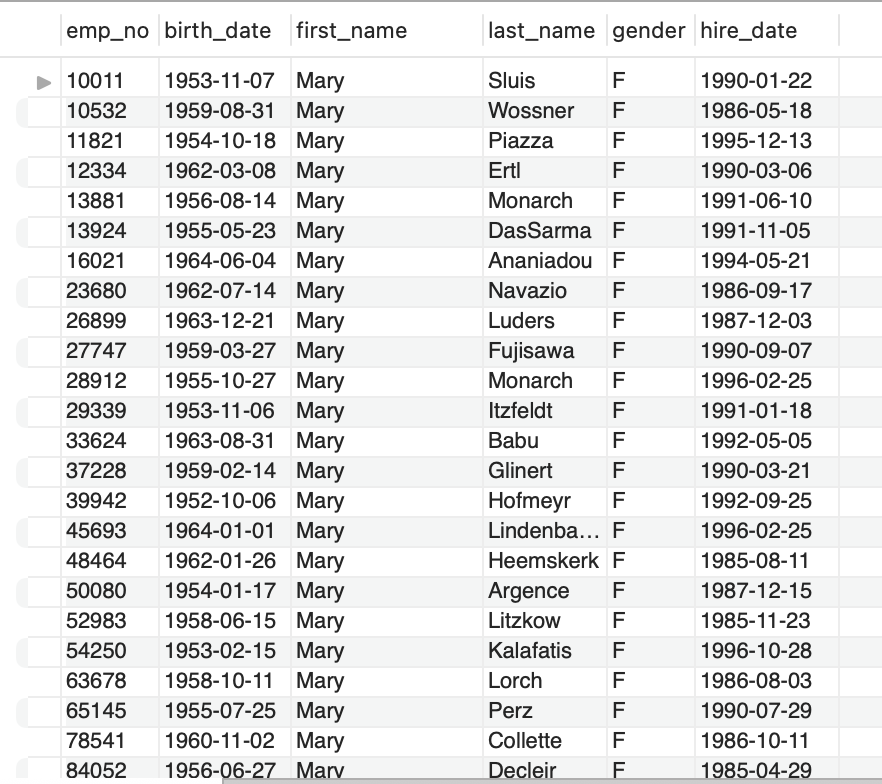
WHERE *condition1* AND *condition2* AND *condition3 ...*;

Example

SELECT \*

FROM employees

WHERE first\_name = 'Mary' and gender = 'F'



OR Syntax

SELECT column1, column2, ...

FROM table

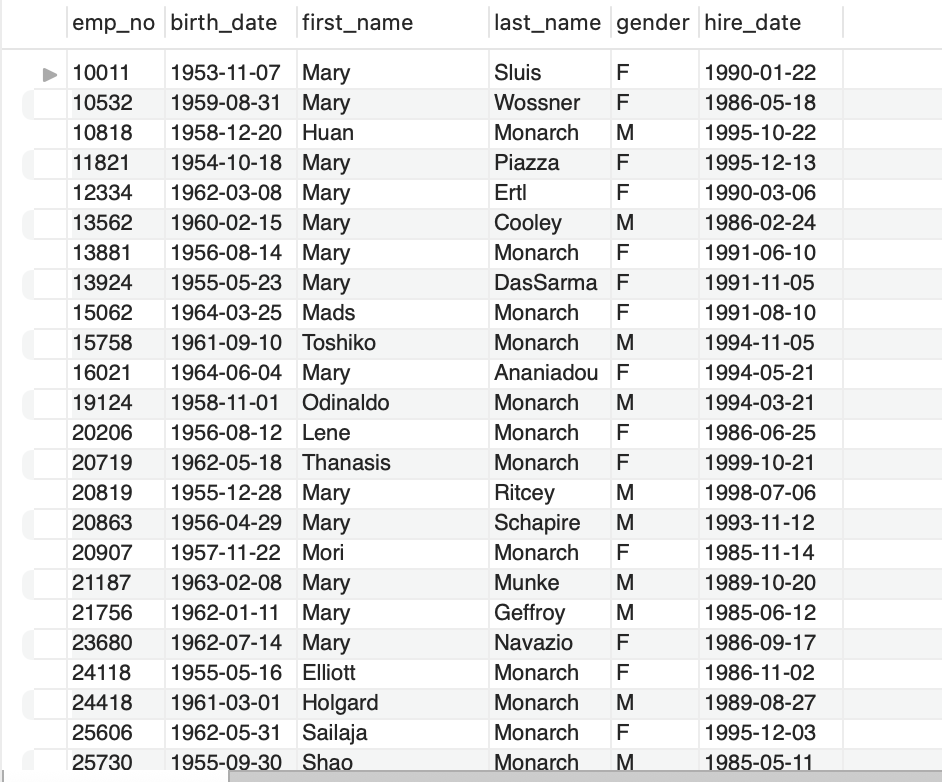
WHERE condition1 OR condition2 OR condition3 ...;

Example

SELECT \*

FROM employees

WHERE first\_name = 'Mary' or last\_name = 'Monarch'



NOT syntax

SELECT column1, column2, ...

FROM table

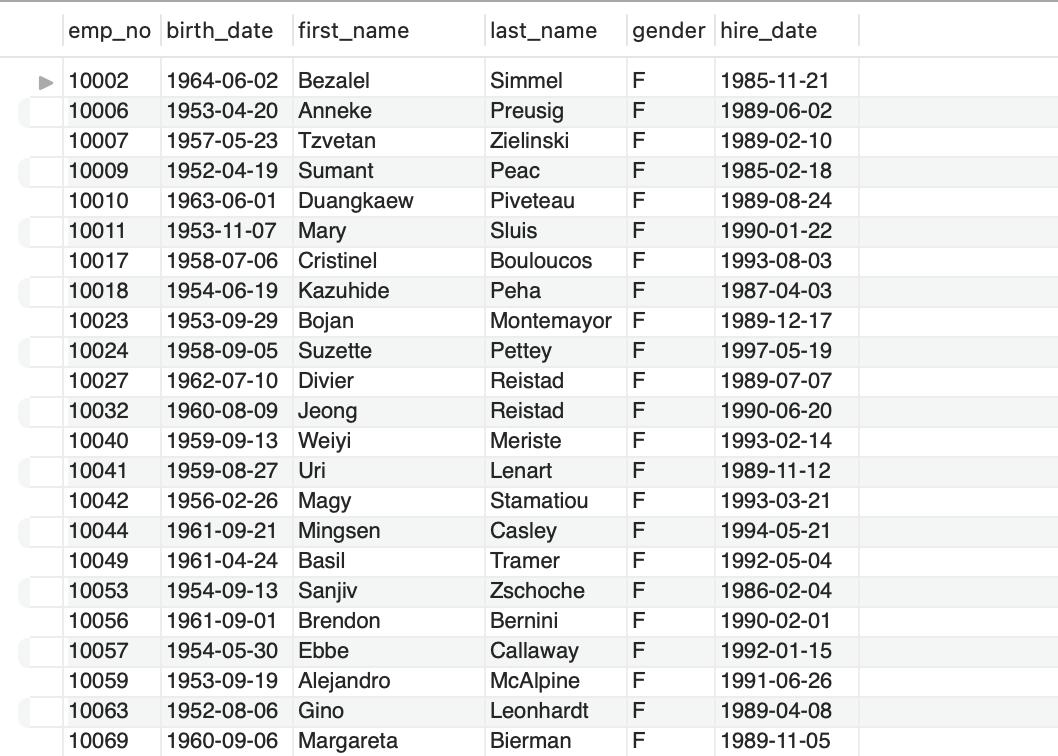
WHERE NOT condition;

Example

SELECT \*

FROM employees

WHERE NOT gender = 'M'



**5. IN OPERATOR**

To match values in a list, the IN operator is used together with the WHERE clause.

Syntax

SELECT *column(s)*

FROM *table*

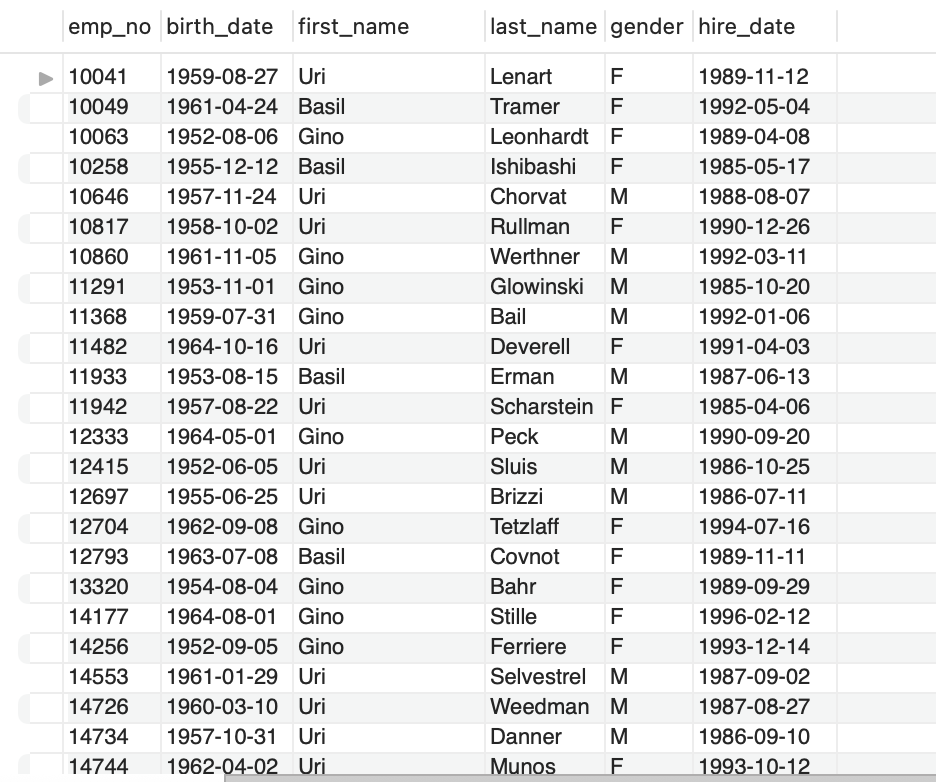
WHERE *column* IN (*value1*, *value2*, ...);

Example

SELECT \*

FROM employees

WHERE first\_name IN ("Uri", "Basil", "Gino");



**6. BETWEEN OPERATOR**

The BETWEEN operator selects values from a specified range. Numbers, text, and dates can all be used as values. The BETWEEN operator includes both begin and end values.

Syntax

SELECT *column(s)*

FROM *table*

WHERE *column* BETWEEN *value1* AND *value2;*

Example

SELECT \*

FROM employees

WHERE emp\_no BETWEEN 10001 AND 10005



**7. LIKE Operator**

In a WHERE clause, the LIKE operator is used to look for a specific pattern in a column.The LIKE operator is frequently used in conjunction with two wildcards:

The percent sign (%) denotes zero, one, or more characters.

The underscore symbol (\_) denotes a single character.

Syntax

SELECT *column1, column2, ...*

FROM *table*

WHERE *columnN* LIKE *pattern*;

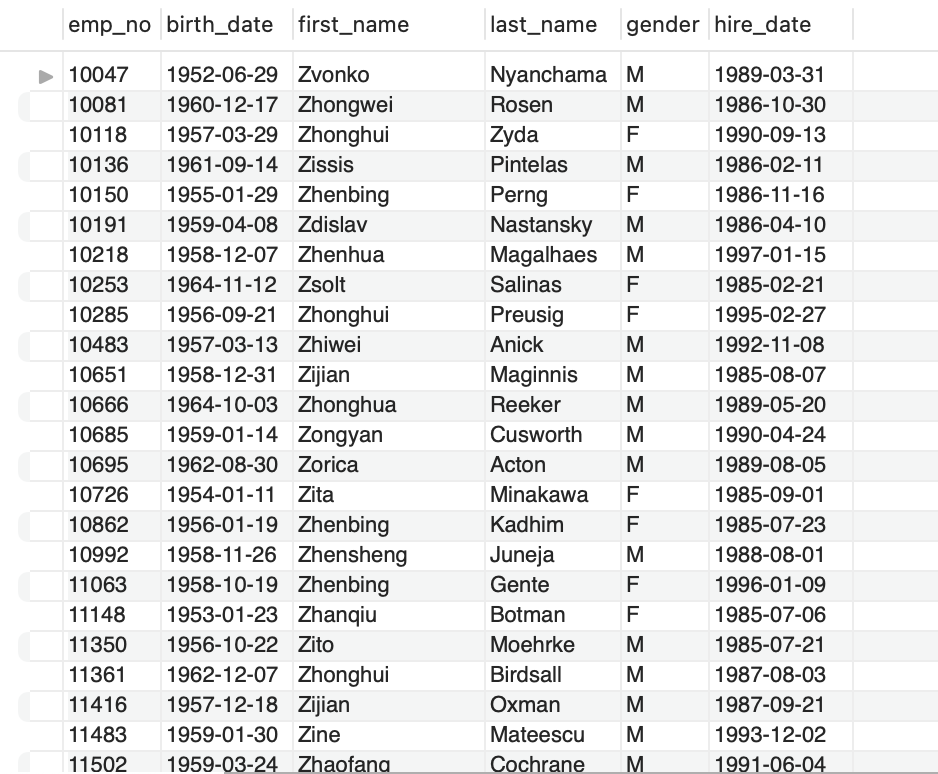
Example

Using the percent sign (%)

SELECT \* FROM employees

WHERE first\_name LIKE 'z%';

Finds any values that start with "z".



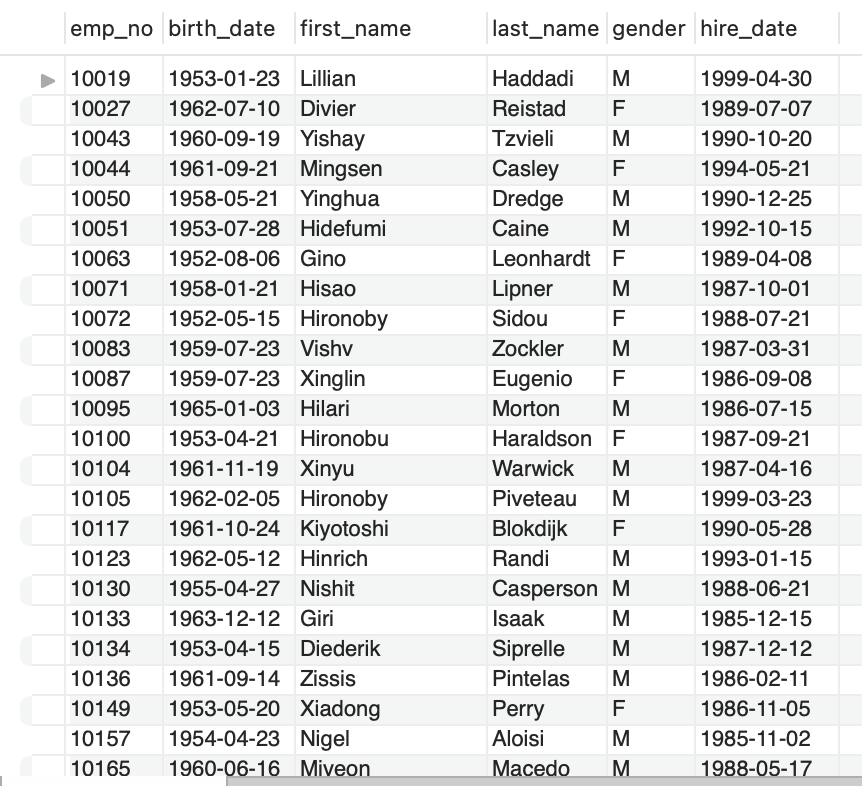
Example

Using the percent sign (\_)

SELECT \* FROM employees

WHERE first\_name LIKE '\_i%';

Finds any values that have "i" in the second position.



**8. ORDER BY CLAUSE**

The ORDER BY clause in SQL is used to sort the result set in ascending or descending order. By default, the ORDER BY keyword sorts the records in ascending order. Use the DESC keyword to sort the records in descending order.

Syntax

SELECT column1, column2, ...

FROM table

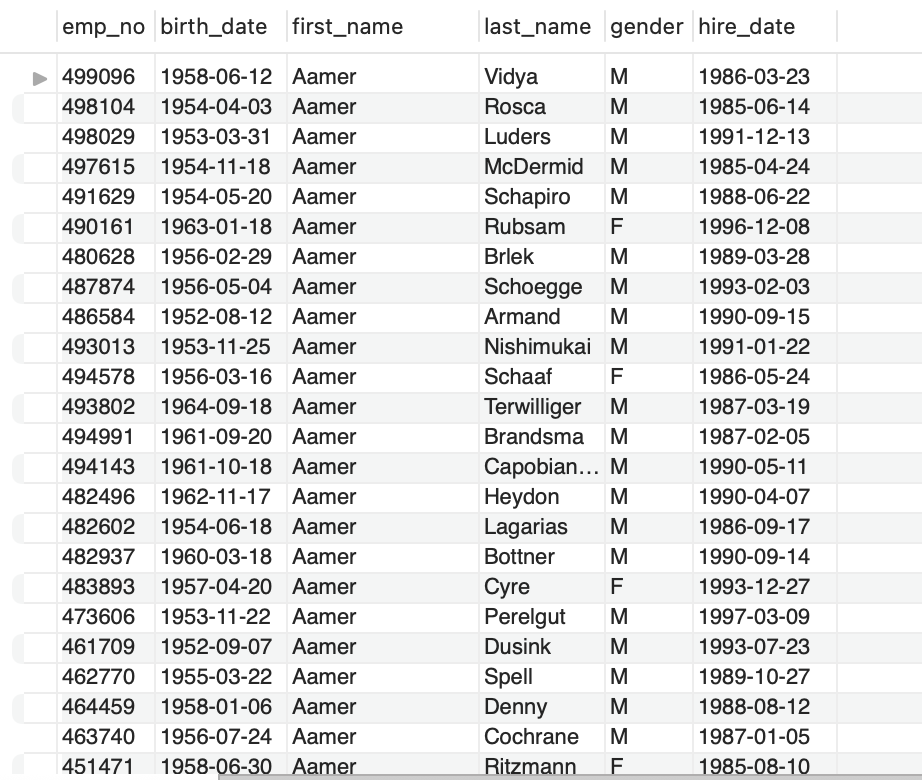
ORDER BY column1, column2, ... ASC|DESC;

Example

SELECT \* FROM employees

ORDER BY first\_name;

Arranging the table according to first\_name in ascending order.



SELECT \* FROM employees

ORDER BY first\_name DESC;

Arranging the table according to first\_name in descending order.



**9. LIMIT CLAUSE**

The LIMIT clause specifies the number of records to return. On large tables with thousands of records, the LIMIT clause comes in handy. A large number of records returned can have an impact on performance.

Syntax

SELECT column(s)

FROM table

WHERE condition

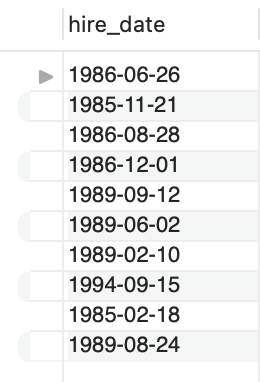
LIMIT number;

Example

SELECT hire\_date

FROM employees

LIMIT 10;



**10. MIN() and MAX() FUNCTION**

The MAX() function returns a column's maximum value.

The MIN() function returns the column's minimum value.

MIN() Syntax

SELECT MIN(*column*)

FROM *table*

WHERE *condition*;

Example

SELECT MIN(birth\_date)

FROM employees

WHERE last\_name = 'Swan';



MIN() Syntax

SELECT MAX(*column*)

FROM *table*

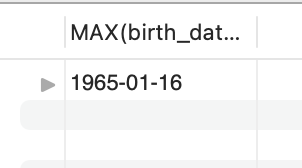
WHERE *condition*;

Example

SELECT MAX(birth\_date)

FROM employees

WHERE last\_name = 'Swan';



**11. COUNT() FUNCTION**

COUNT() returns the number of rows in the result.

COUNT() Syntax

SELECT COUNT(*column*)

FROM *table*

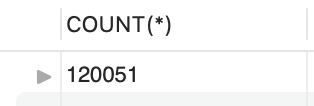
WHERE *condition*;

Example

SELECT COUNT(\*)

FROM employees

WHERE gender = 'F'



**12. AVG() FUNCTION**

AVG() returns the average value of a numerical column.

Syntax

SELECT AVG(*column\_name*)

FROM *table\_name*

WHERE *condition*;

Example

SELECT AVG(salary) FROM salaries;



**13. SUM() FUNCTION**

SUM() returns the total sum of a numerical column.

Syntax

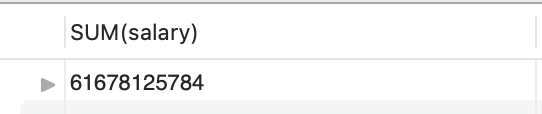
SELECT SUM(*column*)

FROM *table*

WHERE *condition*;

Example

SELECT SUM(salary) FROM salaries;



**14. GROUP BY CLAUSE**

To group rows by one or more columns, use the GROUP BY clause.

Syntax

SELECT *column(s)*

FROM *table*

WHERE *condition*

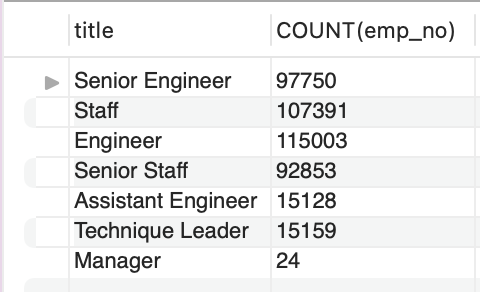
GROUP BY *column(s)*

Example

SELECT title,COUNT(emp\_no)

FROM titles

GROUP BY title;



**15. HAVING CLAUSE**

Instaed of the WHERE clause HAVING clause is used with aggregate functions. And it is used always after the GROUP BY Clause.

Syntax

SELECT *column*

FROM *table*

WHERE *condition*

GROUP BY *column*

HAVING *condition;*

Example

SELECT title,COUNT(\*)

FROM titles

GROUP BY title

HAVING COUNT(\*) > 50000

