Lab 02: Perform Arithmetic Operations & Querying Database Tables

Objective(s):

1. Arithmetic operators
2. WHERE clause
3. Operators in the WHERE clause
4. SQL Operator Precedence

1: Arithmetic operators

Arithmetic operators can perform arithmetical operations on numeric operands involved. Arithmetic operators are addition (+), subtraction (-), multiplication (*) and division (/). The + and - operators can also be used in date arithmetic.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo

Syntax:

```
SELECT expression <arithmetic operator> expression
FROM table_name
WHERE condition;
```

2: WHERE clause

The next thing we want to do is to start limiting, or filtering, the data we fetch from the database.

By adding a WHERE clause to the SELECT statement, we add one (or more) conditions that must be met by the selected data. This will limit the number of rows that answer the query and are fetched. In many cases, this is where most of the "action" of a query takes place.

In other words:

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Note: The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.! (will learn in upcoming labs)

The following SQL statement selects all the employees from the FIRST_NAME "Ellen", in the "employees" table:

Example:

```
SELECT *
FROM employees
WHERE FIRST_NAME = 'Ellen';
```

If you want to get the opposite, the employees other than Ellen then query will be:

```
SELECT *
FROM employees
WHERE FIRST_NAME <> 'Ellen';
```

Also you can use "**!=**" at the place of "**<>**".

Text Field & Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes.

Syntax:

```
SELECT *
FROM employees
WHERE EMPLOYEE_ID = 103;
```

3: Operators in the WHERE clause

Operator	Type	Description

=	Comparison	Equal
>	Comparison	Greater than
<	Comparison	Less than
>=	Comparison	Greater than or equal
<=	Comparison	Less than or equal
<>	Comparison	Not equal. Note: In some versions of SQL this operator may be written as !=
AND	Logical	
OR	Logical	
NOT	Logical	
BETWEEN	Logical	Between a certain range
LIKE	Logical	Search for a pattern
IN	Logical	To specify multiple possible values for a column

AND, OR & NOT operator:

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

BETWEEN operator:

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

LIKE operator:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

"%" ➔ The percent sign represents zero, one, or multiple characters

"_" ➔The underscore represents a single character

The percent sign and the underscore can also be used in combinations!

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name LIKE pattern;
```

Here are some examples showing different LIKE operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

Example:

```
SELECT *
FROM employees
WHERE FIRST_NAME LIKE 'El%';
```

IN Operator:

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IN (value1, value2, value3, ...);
```

OR

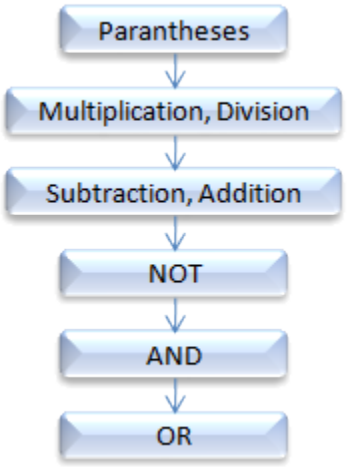
```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

3: SQL Operator Precedence

Operator precedence describes the order in which operations are performed when an expression is evaluated.

Operations with a higher precedence are performed before those with a lower precedence.

Parentheses has the highest precedence and **OR** has the lowest.



Lab Task(s):

Exercise

1. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose SALARY is less than $3000.

2. Write a query to display FIRST_NAME, LASTNAME of all employees whose first name starts with letter 'A'.
3. Write a query to display FIRST_NAME, JOB_ID, DEPARTMENT_ID of employees who are either PU_CLERK or belongs to MANAGER_ID = 114.
4. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose salaries lies in the range of $1500 to $3000;
5. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose commission is empty.
6. Write a query to display first names of all employees that end with alphabet 'N'.
7. Write a query to display FIRST_NAME, JOB_ID, DEPARTMENT_ID of employees who are not PU_CLERK.
8. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of those employees who do not have salaries of $3300, $3200, $2200.
9. Write a query to display names of those employees whose first name starts with 'A' and ends with 'N'.
10. Write a query to display the list of employee names that have letters 'LA' in their names.
11. Write a query to display the EMPLOYEE_ID, FIRST_NAME, and SALARY of employees. In that, the highest paid employee should display first and lowest paid should display last.
12. Write a query to display FIRST_NAME of employees that have "a" in the second position.
13. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose salaries do not lies in the range of $1500 to $3000;
14. Write a query to display FIRST_NAME, LAST_NAME and DEPARTMENT_ID of all employees in departments 30 or 100 in ascending order.
15. Write a query to display FIRST_NAME, LAST_NAME and SALARY for all employees whose salary is not in the range $10,000 through $15,000 and are in department 30 or 100.
16. Write a query to display FIRST_NAME, LAST_NAME and HIRE_DATE for all employees who were hired in 1987.
17. Write a query to display the LAST_NAME of employees whose LAST_NAME have exactly 6 characters.
18. Write a query to display FIRST_NAME, SALARY and PF (15% of salary) of all employees.
19. Write a query to display FIRST_NAME, SALARY and commission amount (% of salary) of all employees.
20. Write a query to display FIRST_NAME, SALARY and NET_SALARY after 500 deduction from salary of all employees;

END