|  |  |  |
| --- | --- | --- |
| **CIS 422 DBMS** | **LAB 02** | **Points: 20** |
| **SQL Training and Exercises**  **ELIJAH CHONG TAY** | | |

**Filtering Data**

1. DISTINCT

To remove duplicate rows from a result set, you use the DISTINCT operator in the SELECT clause as follows:

**SELECT DISTINCT**

**column1, column2,...**

**FROM**

**table1;**

1. LIMIT

To limit the number of rows returned by a select statement, you use the LIMIT and OFFSET clauses. The following shows the syntax of LIMIT & OFFSET clauses:

**SELECT**

**column\_list**

**FROM**

**table1**

**ORDER BY column\_list**

**LIMIT row\_count OFFSET offset;**

* The **LIMIT row\_count** determines the number of rows (row\_count) returned by the query.
* The **OFFSET offset** clause skips the offset rows before beginning to return the rows.

**Note: When you use the LIMIT clause, it is important to use an ORDER BY clause to ensure the order of rows in the result set.**

1. Logical operators

A logical operator allows you to test for the truth of a condition and returns a value of true, false, or unknown.

The following table illustrates the SQL logical operators:

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| ALL | Return true if all comparisons are true |
| AND | Return true if both expressions are true |
| ANY | Return true if any one of the comparisons is true. |
| BETWEEN | Return true if the operand is within a range |
| EXISTS | Return true if a subquery contains any rows |
| IN | Return true if the operand is equal to one of the values in a list |
| LIKE | Return true if the operand matches a pattern |
| NOT | Reverse the result of any other Boolean operator |
| OR | Return true if either expression is true |
| SOME | Return true if some of the expressions are true |

* AND operator

The AND operator allows you to construct multiple conditions in the WHERE clause of an SQL statement such as SELECT, UPDATE, and DELETE:

**expression1 AND expression2**

* OR operator

Similar to the AND operator, the OR operator combines multiple conditions in an SQL statement’s WHERE clause:

**expression1 OR expression2**

* IS NULL

The IS NULL operator compares a value with a null value and returns true if the compared value is null; otherwise, it returns false. The syntax of IS NULL operator is as follows:

**expression IS NULL**

* BETWEEN Operator

The BETWEEN operator searches for values that are within a set of values, given the minimum value and maximum value. Note that the minimum and maximum values are included as part of the conditional set. The syntax of the BETWEEN operator is as follows:

**expression BETWEEN low AND high;**

* IN Operator

The IN operator compares a value to a list of specified values. The IN operator returns true if the compared value matches at least one value in the list; otherwise, it returns false. The following illustrates the syntax of the IN operator:

**expression IN (value1,value2,...)**

* LIKE Operator

The LIKE operator is one of the SQL logical operators. The LIKE operator returns true if a value matches a pattern or false otherwise. The syntax of the LIKE operator is as follows:

**expression LIKE pattern**

SQL provides two wildcards used in conjunction with the LIKE operator:

* + The percent sign (%) represents zero, one, or multiple characters.
  + The underscore sign (\_) represents a single character.
* ALL Operator

The ALL operator compares a value to all values in another value set. The ALL operator must be preceded by a comparison operator and followed by a subquery.

The following illustrates the syntax of the ALL operator:

**comparison\_operator ALL (subquery)**

* ANY Operator

The ANY operator compares a value to any value in a set according to the condition as shown below:

**comparison\_operator ANY(subquery)**

Similar to the ALL operator, the ANY operator must be preceded by a Comparison operator and followed by a subquery.

* EXISTS Operator

The EXISTS operator tests if a subquery contains any rows:

**EXISTS (subquery)**

If the subquery returns one or more rows, the result of the EXISTS is true; otherwise, the result is false.

* NOT Operator

NOT is used to negate the result of any Boolean expression. The following illustrates how to use the NOT operator:

**NOT [Boolean\_expression]**

**Lab Exercises**

***Please for each of the following question, provide the SQL statement and the screenshot for the result from the phpMyAdmin.***

1. Selects the unique values of salary column of the **employees** table and sorts them from high to low.

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC;

1. Use the LIMIT clause to return the first 5 rows in the **employees** table sorted by the first\_name column?

SELECT first\_name

FROM employees

ORDER BY first\_name LIMIT 5;

1. Use the LIMIT clause to get the top five employees with the highest salaries?

SELECT first\_name, employee\_id

FROM employees

ORDER BY salary DESC

LIMIT 5;

1. Use the LIMIT OFFSET clauses and subquery to get employees who have the 2nd highest salary in the company?

SELECT employee\_id, first\_name, last\_name, salary

FROM employees

WHERE salary = (

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 1 );

1. Find all employees whose salaries are greater than 5,000 and less than 7,000?

SELECT employee\_id, first\_name, last\_name, salary

FROM employees

WHERE salary > 5000 AND salary < 7000;

1. Find employees whose salary is either 7,000 or 8,000?

SELECT employee\_id, first\_name, last\_name, salary

FROM employees

WHERE salary = 7000 OR salary = 8000;

1. Find all employees who do not have a phone number?

SELECT employee\_id, first\_name, last\_name

FROM employees

WHERE phone\_number IS NULL;

1. Find all employees who have phone numbers?

SELECT employee\_id, first\_name, last\_name

FROM employees

WHERE phone\_number IS NOT NULL;

1. Find all employees whose salaries are between 9,000 and 12,000?

SELECT employee\_id, first\_name, last\_name

FROM employees

WHERE salary > 9000 AND salary < 12000

1. Find employees who have not joined the company from January 1, 1989 to December 31, 1999?

SELECT employee\_id, first\_name, last\_name, hire\_date

FROM employees

WHERE hire\_date NOT BETWEEN '1989-01-01' AND '1999-12-31';

1. Using IN operator, find employees who work in the Marketing and Sales departments?

SELECT employee\_id, first\_name, last\_name, department\_id

FROM employees

WHERE department\_id IN (2,8);

1. Find all employees whose first names end with er?

SELECT employee\_id, first\_name, last\_name, department\_id

FROM employees

WHERE first\_name LIKE '%er';

1. Find employees whose last names contain the word an?

SELECT employee\_id, first\_name, last\_name, department\_id

FROM employees

WHERE last\_name LIKE '%an%';

1. Find all employees whose first names start with the letter S but not start with Sh?

SELECT employee\_id, first\_name, last\_name, department\_id

FROM employees

WHERE first\_name LIKE 'S%' AND first\_name

1. Using ALL operator, find all employees whose salaries are greater than all salaries of employees in the department 8?

SELECT employee\_id, first\_name, last\_name, department\_id, salary

FROM employees

WHERE salary > ALL (

SELECT salary

FROM employees

WHERE department\_id = 8

)

1. Using ANY operator, find all employees whose salaries are greater than the average salary of every department?

SELECT employee\_id, first\_name, last\_name, department\_id, salary

FROM employees

WHERE salary > ANY (

SELECT AVG(salary)

FROM employees

GROUP BY department\_id );

1. Find all employees who have dependents using EXISTS operator?

SELECT employee\_id, first\_name, last\_name

FROM employees

WHERE EXISTS (

SELECT \*

FROM dependents

WHERE employee\_id = dependents.employee\_id);

1. Use NOT operator to find the employees who are not working in the departments 1, 2, or 3?

SELECT employee\_id, first\_name, last\_name, department\_id

FROM employees

WHERE NOT department\_id = 1 AND NOT department\_id = 2 AND NOT department\_id = 3;

1. Uses the NOT EXISTS operator to get the employees who do not have any dependents?

SELECT employee\_id, first\_name, last\_name

FROM employees

WHERE NOT EXISTS (

SELECT \*

FROM dependents

WHERE employees.employee\_id = dependents.employee\_id

)