Merit-Quality-Excellence

IBA Institute of Emerging Technologies Khairpur

DATABASE MANAGEMENT SYSTEM

LAB No: 04

Instructor: Marina Gul

Objective of Lab No. 4:
After performing lab 4, students will be able to:

- Aggregate Functions
- GROUP BY Statement
- HAVING clause
- Regular Expressions

Aggregate Functions

An aggregate function allows you to perform a calculation on a set of values to return a single scalar value. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

WHERE clause comes before GROUP BY clause.

HAVING clause can come after GROUP BY clause.

Alias names cannot be used in GROUP BY clause.

GROUP BY columns are not necessarily be in the SELECT clause

GROUP BY clause can be applied on more than one columns

The following are the most commonly used SQL aggregate functions:

- AVG – calculates the average of a set of values.
- COUNT– counts rows in a specified table or view.
- MIN – gets the minimum value in a set of values.
- MAX – gets the maximum value in a set of values.
- SUM – calculates the sum of values.

Notice that all aggregate functions above ignore NULL values except for the COUNT function.

SQL aggregate functions syntax: To call an aggregate function, you use the following syntax:

```
aggregate_function (DISTINCT | ALL expression)
```

First, specify an aggregate function that you want to use e.g., MIN, MAX, AVG, SUM or COUNT.

Second, put DISTINCT or ALL modifier followed by an expression inside parentheses. If you explicitly use the DISTINCT modifier, the aggregate function ignores duplicate values and only consider the unique values. If you use the ALL modifier, the aggregate function uses all values for calculation or evaluation. The ALL modifier is used by default if you do not specify any modifier explicitly.

GROUP BY Statement

The GROUP BY statement group rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

DISTINCT vs GROUP BY

Distinct is used to find unique/distinct records where as a group by is used to group a selected set of rows into summary rows by one or more columns or an expression.

The group by can also be used to find distinct values as shown in below query.

```
SELECT DEPARTMENT_ID
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID;
```

OR

```
SELECT DISTINCT DEPARTMENT_ID
FROM EMPLOYEES ;
```

HAVING clause

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
```

```
ORDER BY column_name(s);
```

Example:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

The above SQL statement lists the number of customers in each country, sorted high to low (Only include countries with more than 5 customers):

Regular Expressions

Regular Expressions help search data matching complex criteria. We looked at wildcards in the previous labs. If you have worked with wildcards before, you may be asking why learn regular expressions when you can get similar results using the wildcards. Because, compared to wildcards, regular expressions allow us to search data matching even more complex criterion.

Syntax:

```
SELECT columns
FROM table
WHERE columnName
REGEXP 'pattern';
```

Example:

```
SELECT *
FROM employees
WHERE FIRST_NAME
REGEXP 'al';
```

The above query searches for all the employees' first name that have the word "al" in them. It does not matter whether the "al" is at the beginning, middle or end of the first name. As long as it is contained in the name then it will be considered.

Let's suppose that we want to search employees whose first name start with a, b, c or d , followed by any number of other characters. We can use a regular expression together with the metacharacters to achieve our desired results.

```
SELECT *
FROM employees
WHERE FIRST_NAME
REGEXP '^[a, b, c, d]';
```

Char	Description	Example
*	The asterisk (*) metacharacter is used to match zero (0) or more instances of the strings preceding it	SELECT * FROM movies WHERE title REGEXP 'da*'; will give all movies containing characters "da" .For Example, Da Vinci Code , Daddy's Little Girls.

?	The question(?) metacharacter is used to match zero (0) or one instances of the strings preceding it.	SELECT * FROM `categories` WHERE `category_name` REGEXP 'com?'; will give all the categories containing string com .For Example, comedy , romantic comedy .
.	The dot (.) metacharacter is used to match any single character in exception of a new line.	SELECT * FROM movies WHERE `year_released` REGEXP '200.'; will give all the movies released in the years starting with characters "200" followed by any single character .For Example, 2005,2007,2008 etc.
^	The caret (^) is used to start the match at beginning.	SELECT * FROM `movies` WHERE `title` REGEXP '^[cd]'; gives all the movies with the title starting with any of the characters in "cd" .For Example, Code Name Black, Daddy's Little Girls and Da Vinci Code.
$	The ($) is used to match at the end.	SELECT * FROM `movies` WHERE `title` REGEXP 'n$'; gives all the movies with the title that ends with n letter.
[abc]	The charlist [abc] is used to match any of the enclosed characters.	SELECT * FROM `movies` WHERE `title` REGEXP '[vwxyz]'; will give all the movies containing any single character in "vwxyz" .For Example, X-Men, Da Vinci Code, etc.
[^abc]	The charlist [^abc] is used to match any characters excluding the ones enclosed.	SELECT * FROM `movies` WHERE `title` REGEXP '^[^vwxyz]'; will give all the movies containing characters other than the ones in "vwxyz".
[A-Z]	The [A-Z] is used to match any upper case letter.	SELECT * FROM `members` WHERE `postal_address` REGEXP '[A-Z]'; will give all the members that have postal address containing any character from A to Z. .For Example, Janet Jones with membership number 1.
[a-z]	The [a-z] is used to match any lower case letter	SELECT * FROM `members` WHERE `postal_address` REGEXP '[a-z]'; will give all the members that have postal addresses containing any character from a to z. .For Example, Janet Jones with membership number 1.
[0-9]	The [0-9] is used to match any digit from 0 through to 9.	SELECT * FROM `members` WHERE `contact_number` REGEXP '[0-9]' will give all the members have submitted contact numbers containing characters "[0-9]" .For Example, Robert Phil.

https://dev.mysql.com/doc/refman/5.6/en/regexp.html

Lab Tasks

1. Write a query to lists the number of employees in each department.
2. Write a query to display the department id where at least 5 employees should be in each department.
3. Write a query to display all columns of those employees who has first name is unique.
4. Write a SQL query to get name of students containing exactly four characters.
5. Write a query to display the list of employee names that have letters 'LA' in their names.
6. Write a query to display names of those employees whose first name starts with 'A' and ends with 'N'.
7. Write a query to display first names of all employees that end with alphabet 'N'.
8. Write a query to display FIRST_NAME, LASTNAME of all employees whose first name starts with letter 'A'.
9. Write a query to display the number of employees with the same job.
10. Display the manager number and the salary of the lowest paid employee of that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is 2000. Sort the output is descending order of the salary.
11. Display the total number of employees who have no commission.
12. Write a query to display FIRST_NAME, LASTNAME of all employees whose first name small 't'.