|  |  |  |
| --- | --- | --- |
| **CIS 422 DBMS** | **LAB 03** | **Points: 20** |
| **SQL Training and Exercises**  **ELIJAH CHONG TAY** | | |

**Aggregate Functions and Grouping Data**

**GROUP BY:**

Grouping is one of the most important tasks that we have to deal with while working with data. To group rows into groups, you use the GROUP BY clause. The GROUP BY clause is an optional clause of the SELECT statement that combines rows into groups based on matching values in specified columns. One row is returned for each group. It’s worth to note that we often use the GROUP BY in conjunction with an aggregate function to calculate a measure that provides the information for each group. To sort the groups, you add the ORDER BY clause after the GROUP BY clause.

**HAVING:**

The HAVING clause is often used with the GROUP BY clause in the SELECT statement. If you use a HAVING clause without a GROUP BY clause, the HAVING clause behaves like the WHERE clause.

The WHERE clause applies the condition to individual rows before the rows are summarized into groups by the GROUP BY clause. However, the HAVING clause applies the condition to the groups after the rows are grouped into groups. Therefore, it is important to note that the HAVING clause is applied after whereas the WHERE clause is applied before the GROUP BY clause.

**Aggregate Functions:**

You will learn about the SQL aggregate functions including AVG(), COUNT(), MIN(), MAX(), and SUM(). An SQL aggregate function calculates on a set of values and returns a single value. For example, the average function (AVG) takes a list of values and returns the average.

Because an aggregate function operates on a set of values, it is often used with the GROUP BY clause of the SELECT statement. The GROUP BY clause divides the result set into groups of values and the aggregate function returns a single value for each group.

You can use aggregate functions as expressions only in the following:

* The select list of a SELECT statement, either a subquery or an outer query.
* A HAVING clause

1. AVG

The SQL AVG function is an aggregate function that calculates the average value of a set. The following illustrates the syntax of the SQL AVG function:

AVG([ALL|DISTINCT] expression)

If we use the ALL keyword, the AVG function takes all values in the calculation. By default, the AVG function uses ALL whether we specify it or not. If we specify the DISTINCT keyword explicitly, the AVG function will take the unique values only in the calculation.

1. COUNT

The SQL COUNT function is an aggregate function that returns the number of rows returned by a query. The following illustrates the syntax of the SQL COUNT function:

COUNT([ALL | DISTINCT] expression);

The result of the COUNT function depends on the argument that you pass to it.

* By default, the COUNT function uses the ALL keyword whether you specify it or not. The ALL keyword means that all items in the group are considered including the duplicate values.
* If you specify the DISTINCT keyword explicitly, only unique non-null values are considered.

Another form of the COUNT function that accepts an asterisk (\*) as the argument is as follows:

COUNT(\*)

* The COUNT(\*) function returns the number of rows in a table including the rows that contain the NULL values.

1. SUM

The SQL SUM function is an aggregate function that returns the sum of all or distinct values. We can apply the SUM function to the numeric column only. The following illustrates the syntax of the SUM function.

SUM([ALL|DISTINCT] expression)

* The ALL operator allows you to apply the aggregate to all values. The SUM function uses the ALL operator by default.
* To calculate the sum of unique values, you use the DISTINCT.

1. MAX

SQL provides the MAX function that allows you to find the maximum value in a set of values. The following illustrates the syntax of the MAX function:

MAX(expression)

* The MAX function ignores NULL values.
* Unlike the SUM, COUNT, and AVG functions, the DISTINCT option is not applicable to the MAX function.

1. MIN

The SQL MIN function returns the minimum value in a set of values. The following demonstrates the syntax of the MIN function:

MIN(expression)

* Like the MAX function, the MIN function also ignores NULL values, and the DISTINCT option is not applicable to the MIN function.

**Subquery**

1. Plain Subquery

In plain subquery, the outer query makes use of the result returned from the subquery. The outer query depends on the subquery for its value. However, the subquery does not depend on the outer query.

1. Correlated Subquery

A correlated subquery is a subquery that uses the values from the outer query. Also, a correlated subquery may be evaluated once for each row selected by the outer query. Because of this, a query that uses a correlated subquery may be slow.

**Lab Exercises**

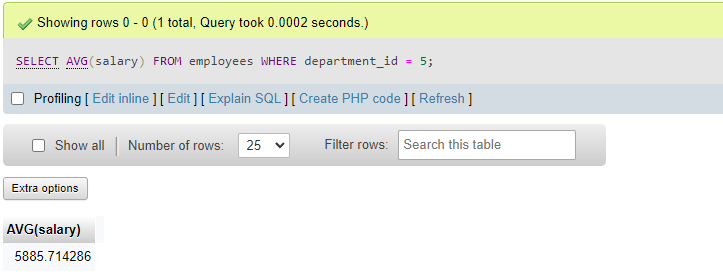
***For each of the following question, provide the SQL statement and the screenshot for the result from the phpMyAdmin.***

1. Calculate the average salary of employees in the department id 5?

SELECT AVG (salary)

FROM employees

WHERE department\_id = 5;



1. Returns department id, department name and the average salary of employees of each department and sort the result in descending order?

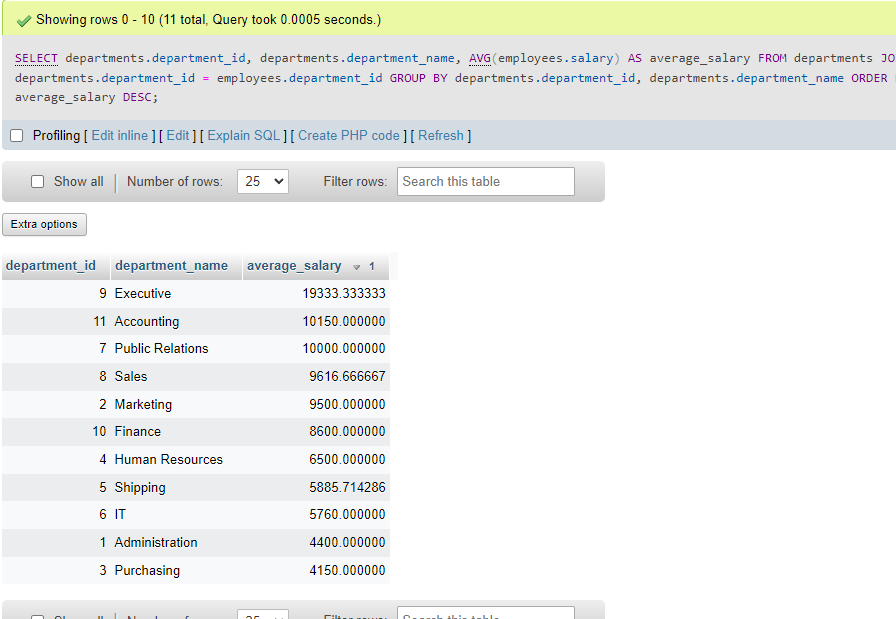
SELECT departments.department\_id, departments.department\_name, AVG(employees.salary) AS average\_salary

FROM departments

JOIN employees ON departments.department\_id = employees.department\_id

GROUP BY departments.department\_id, departments.department\_name

ORDER BY average\_salary DESC;



1. Find the department id, department name and number of employees for each department in which the number of employees is greater than 5 sorted by the number of employees in descending order?

SELECT departments.department\_id, departments.department\_name,

COUNT(employees.employee\_id) AS num\_employees

FROM employees

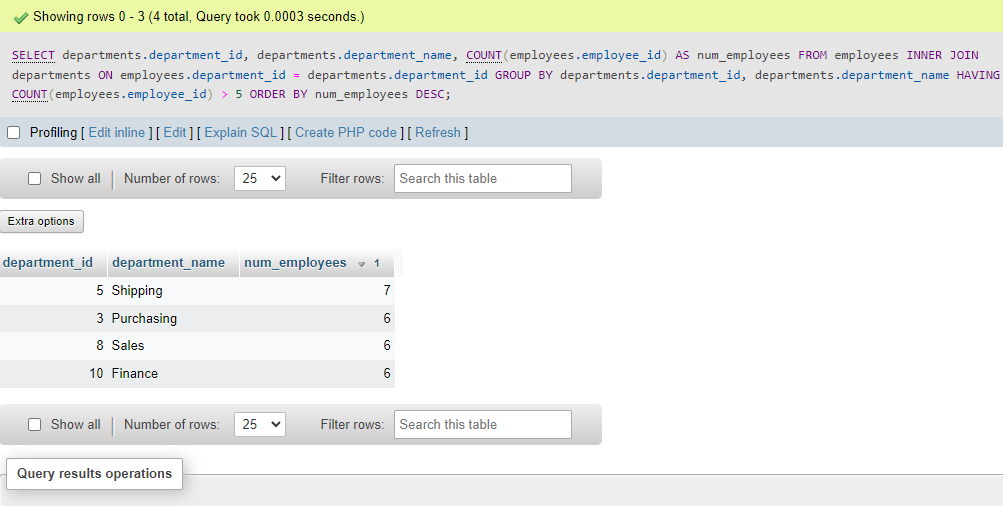
INNER JOIN departments

ON employees.department\_id = departments.department\_id

GROUP BY departments.department\_id, departments.department\_name

HAVING COUNT(employees.employee\_id) > 5

ORDER BY num\_employees DESC



1. Find department id, department name and the total salaries for each department in which the total salaries is greater than 3000 sorted in descending order?

Inner join to combine employees and departments

Having clause to filter departments with total\_salaries > 3000

SELECT departments.department\_id, departments.department\_name, SUM(employees.salary) AS total\_salaries

FROM employees

INNER JOIN departments

ON employees.department\_id = departments.department\_id

GROUP BY departments.department\_id, departments.department\_name

HAVING SUM(employees.salary) > 3000

ORDER BY total\_salaries DESC;

Graphical user interface, application

Description automatically generated

1. Find the employee(s) id, first name, last name, and salary who have the highest salary? Note: try to use the subquery in this question.

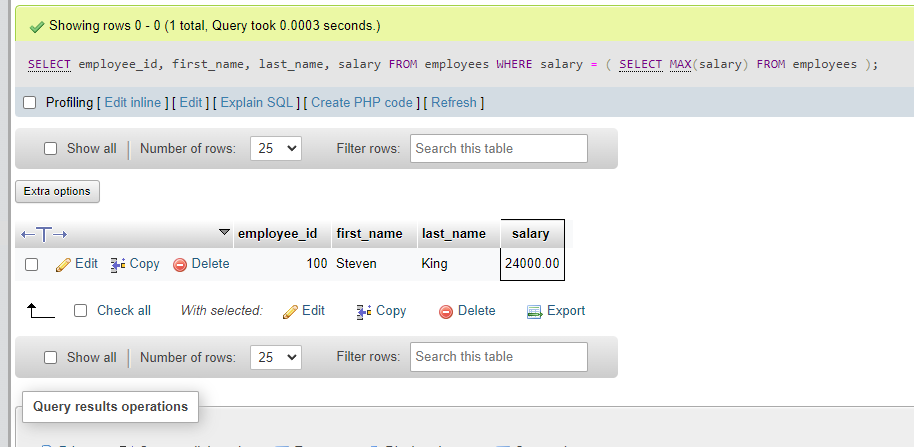
SELECT employee\_id, first\_name, last\_name, salary

FROM employees

WHERE salary = (

SELECT MAX(salary)

FROM employees)



1. Find department id, department name for each department that has employee whose highest salary is greater than 12000?

SELECT departments.department\_id, departments.department\_name

FROM departments

INNER JOIN employees

ON departments.department\_id = employees.department\_id

GROUP BY departments.department\_id, departments.department\_name

HAVING MAX(employees.salary) > 12000;

1. Retrieve the employee who has the lowest salary in each department, then sorts these departments by the salary in ascending order?

SELECT departments.department\_id, departments.department\_name, employees.employee\_id, employees.first\_name, employees.last\_name, employees.salary

FROM departments

INNER JOIN employees ON departments.department\_id = employees.department\_id

WHERE employees.salary = (

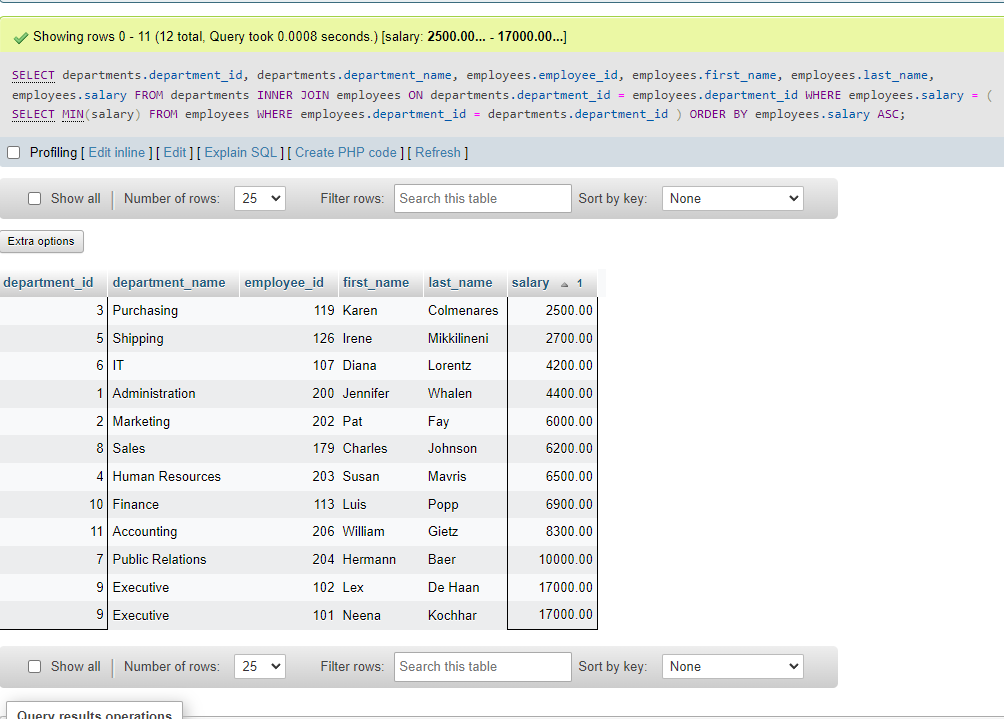
SELECT MIN(salary)

FROM employees

WHERE employees.department\_id = departments.department\_id

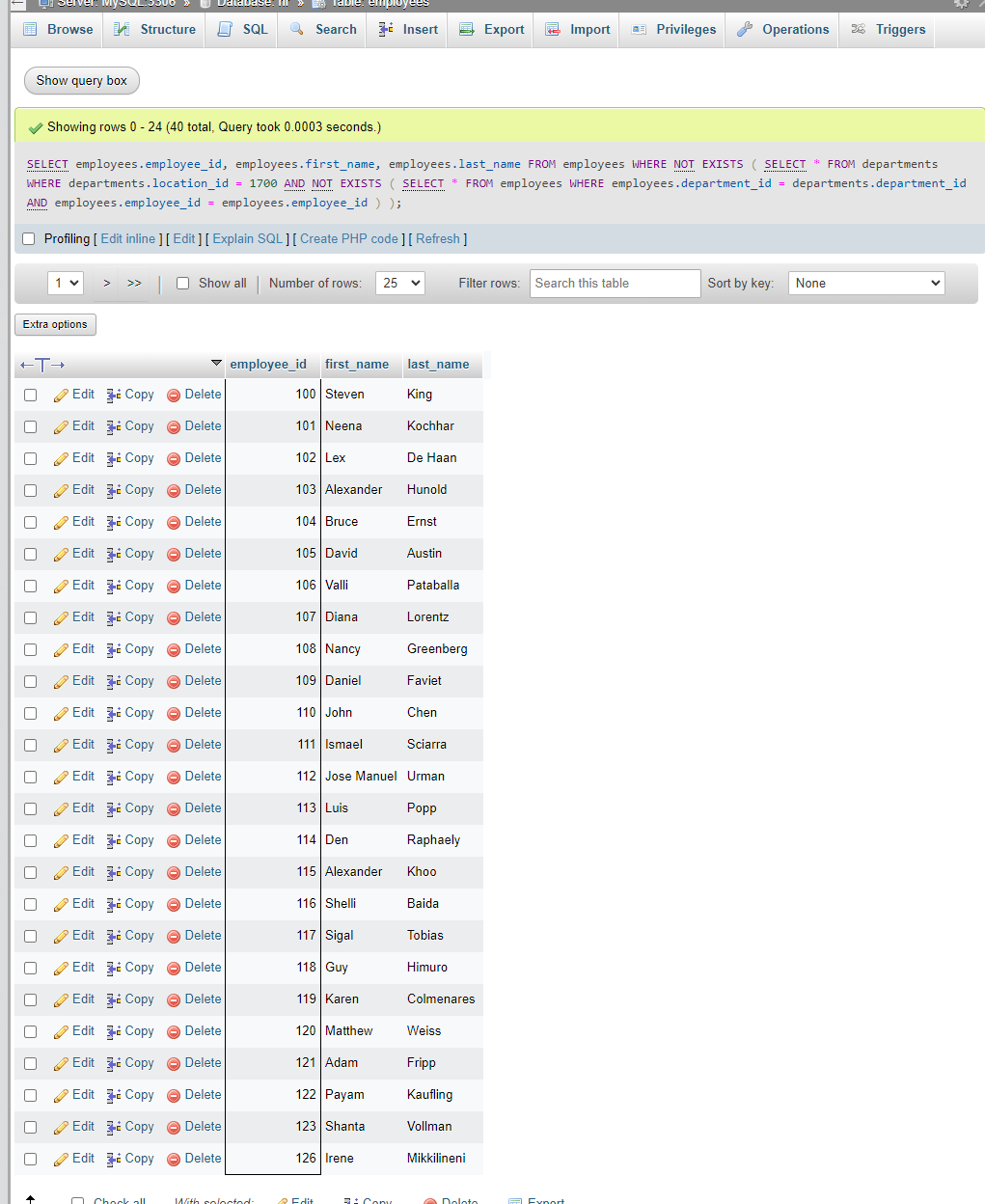
)

ORDER BY employees.salary ASC;



1. Find the Id, first name and second name for all employees who are working for all departments located at the location whose id is 1700.

SELECT employees.employee\_id, employees.first\_name, employees.last\_name FROM employees WHERE NOT EXISTS ( SELECT \* FROM departments WHERE departments.location\_id = 1700 AND NOT EXISTS ( SELECT \* FROM employees WHERE employees.department\_id = departments.department\_id AND employees.employee\_id = employees.employee\_id ) );



1. Find the Id, first name and second name for all employees who have the highest salary.

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

FROM employees

WHERE salary = (

SELECT MAX(salary)

FROM employees );

Graphical user interface, text, application

Description automatically generated

1. Find the Id, first name, second name and salary for all employees whose salaries are greater than the lowest salary of every department.

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

FROM employees

WHERE salary > (

SELECT MIN(salary)

FROM employees

WHERE department\_id = employees.department\_id );

Graphical user interface

Description automatically generated with medium confidence

1. Calculate the average of average salary of all departments.

SELECT AVG(avg\_salary) as avg\_of\_avgs

FROM (

SELECT AVG(salary) as avg\_salary

FROM employees

GROUP BY department\_id

) AS department\_salaries;

Graphical user interface, text, application, email

Description automatically generated

1. Find the Id, first name, second name, salary, average salary, and the difference between the salary of each employee and the average salary for all employees

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

(SELECT AVG (salary) FROM employees) AS salary\_avg,

salary - (SELECT AVG (salary) FROM employees) AS salary\_difference FROM employees;

Graphical user interface, application

Description automatically generated

1. Find the Id, first name, second name, salary and department id of all employees whose salary is higher than the average salary of the employees in their departments.

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

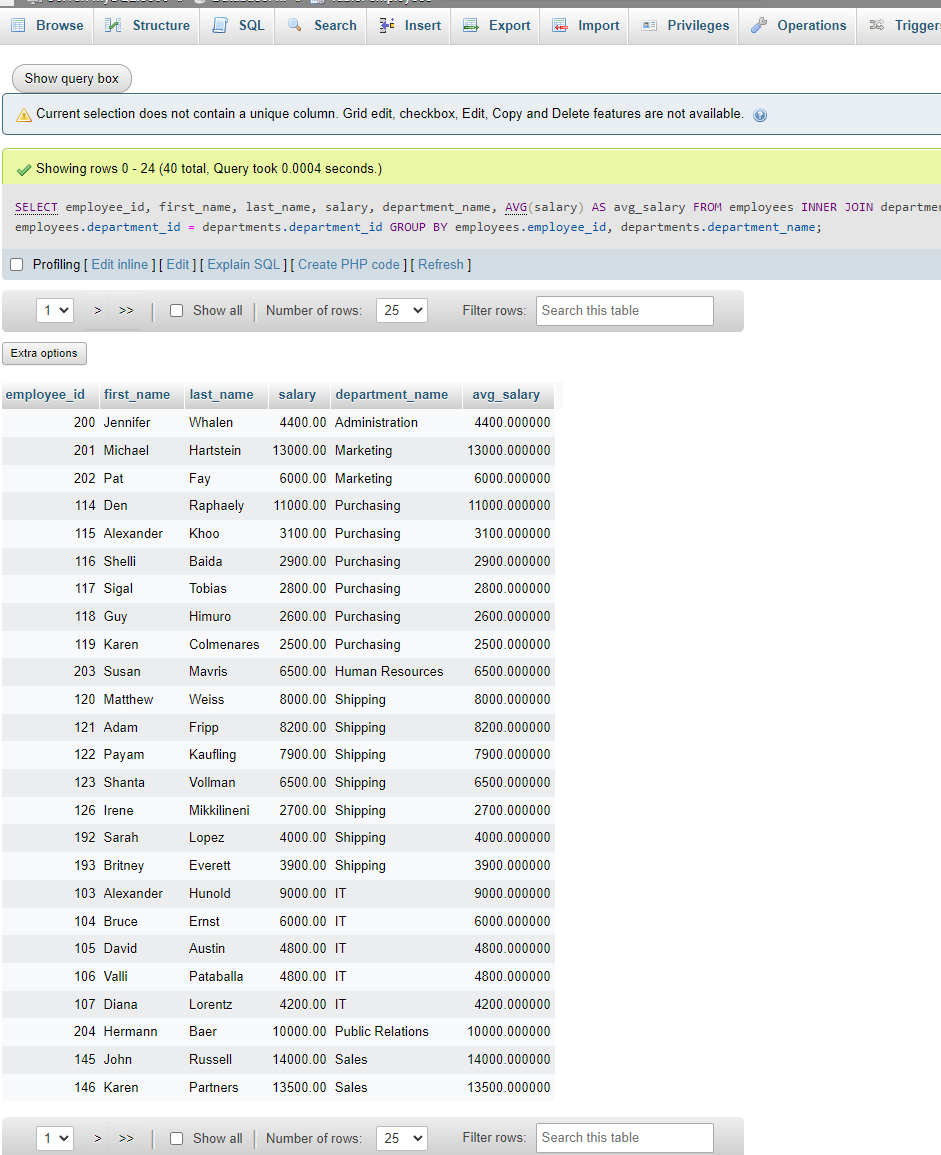
FROM employees

WHERE salary > (

SELECT AVG(salary)

FROM employees e

WHERE employees.department\_id = e.department\_id);



1. Find the Id, first name, second name, salary, department name and the average salary of all employees in their departments.

SELECT employee\_id, first\_name, last\_name, salary, department\_name, AVG(salary) AS avg\_salary

FROM employees

INNER JOIN departments ON employees.department\_id = departments.department\_id

GROUP BY employees.employee\_id, departments.department\_name;

Graphical user interface, application

Description automatically generated

1. Find the Id, first name and second name of all employees who have no dependents.

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

FROM employees

WHERE employee\_id NOT IN (

SELECT employee\_id FROM dependents)

