# Faculty of Computing

**CS220: Database Systems**

**Class: BESE-13AB**

# Lab 06: Multiple row Functions (aggregating data using group functions)

# Date: 16th October, 2023

# Time: 10:00-1:00

# Instructor: Dr. Bilal Ali

# Lab Engineer: Sundas Dawood

**Name:** Aimen Munawar

**Class:** BESE-13-A

**CMS ID:** 415867

# Lab 06: Multiple row Functions(aggregating data using group functions)

**Introduction**

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

Relational Algebra is a meta-language and forms underlying basis of SQL query language. It has six basic operators including: select, project, union, set difference, rename, and cross product. The operators take one or two relations as inputs and produce a new relation as a result.

**Objectives**

After completing this lab, you should be able to do the following:

* Identify the available group functions
* Describe the use of group functions
* Group data using the GROUP BY clause
* Include or exclude grouped rows by using the HAVING clause
* Extracting data from multiple tables

**Tools/Software Requirement**

MySQL workbench

**Description**

This lab further addresses functions. It focuses on obtaining summary information, such as averages, for groups of rows. It discusses how to group rows in a table into smaller sets and how to specify search criteria for groups of rows.

**Instructions**

Execute the company.sql script to create company schema first. After that, practice the given examples and all group functions of SQL. At the end, attempt the questions given as lab tasks in the manual.

**NOTE:**

Reference link for database creation:

https://justinsomnia.org/2009/04/the-emp-and-dept-tables-for-mysql/

# SQL Group by Clause:

## Lab Practice 1: using the SUM function

For example, you could also use the SUM function to return the department-id and the total salary (in the associated department).

**SELECT deptno, SUM(sal) as "Total salary"  
FROM emp  
GROUP BY deptno;**

Because you have listed one column in your SELECT statement that is not encapsulated in the SUM function, you must use a GROUP BY clause. The department field must, therefore, be listed in the GROUP BY section.

## Lab Practice 2: using the COUNT function

For example, you could use the COUNT function to return the department-id and the number of employees (in the associated department) that make over $25,000 / year.

**SELECT deptno, COUNT(\*) as "Number of employees"  
FROM emp  
WHERE sal > 2500  
GROUP BY deptno;**

## Lab Practice 3: using the MIN function

For example, you could also use the MIN function to return the department-id and the minimum salary in the department.

**SELECT deptno, MIN(sal) as "Lowest salary"  
FROM emp  
GROUP BY deptno;**

## Lab Practice 4: using the MAX function

For example, you could also use the MAX function to return the department-id and the maximum salary in the department.

**Formulate the query yourself.**

# SQL : Having Clause

## Lab Practice 5: using the SUM function

For example, you could also use the SUM function to return the department-id and the total sales (in the associated department). The HAVING clause will filter the results so that only departments with sales greater than $1000 will be returned.

## SELECT deptno, SUM(sal) as "Total sales"

## FROM emp

## GROUP BY deptno

## HAVING SUM(sal) > 10000;

## Lab Practice 6: using the COUNT function

For example, you could use the COUNT function to return the name of the department and the number of employees (in the associated department) that make over $25,000 / year. The HAVING clause will filter the results so that only departments with more than 10 employees will be returned.

**SELECT deptno, COUNT(\*) as "Number of employees"  
FROM emp  
WHERE sal > 2500  
GROUP BY deptno  
HAVING COUNT(\*) > 1;**

## Lab Practice 7: using the MAX function

For example, you could also use the MAX function to return the id of each department and the maximum salary in the department. The HAVING clause will return only those departments whose maximum salary is less than $50,000.

**SELECT deptno, MAX(sal) as "Highest salary"  
FROM emp  
GROUP BY deptno  
HAVING MAX(sal) < 5000;**

**ALIAS**

SQL aliases are used to temporarily rename a table or a column heading.

## SQL Aliases

SQL aliases are used to give a database table, or a column in a table, a temporary name. Basically aliases are created to make column names more readable.

### SQL Alias Syntax for Columns

SELECT column\_name AS alias\_name  
FROM table\_name;

### SQL Alias Syntax for Tables

SELECT column\_name(s)  
FROM table\_nameAS alias\_name;

**Example:**

SELECT first\_name AS Customer

FROM sakila.customer;

AS keyword is optional, even if you remove it a column or table alias is formulated.

**Lab TASKS:**

**Lab task 1:**

**Create Library Database - GROUP BY Clause**

**1. Create a new database named `LibraryDB`.**

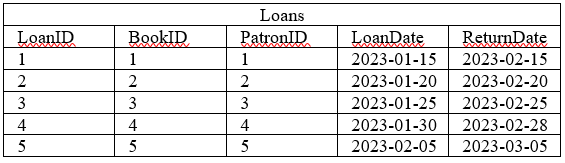
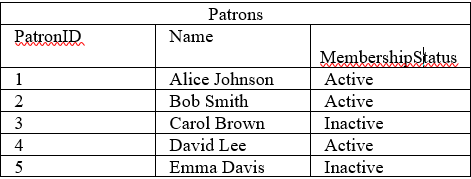
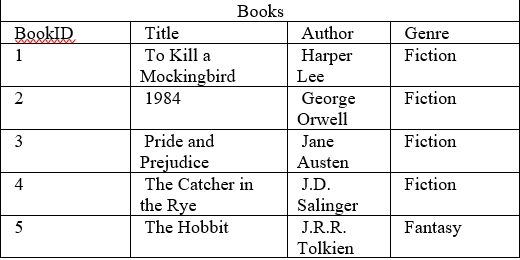
**2. Create the following tables:**

**- `Books`: (BookID, Title, Author, Genre)**

**- `Patrons`: (PatronID, Name, MembershipStatus)**

**- `Loans`: (LoanID, BookID, PatronID, LoanDate, ReturnDate)**

**3. Insert sample data into the tables.**



**My Code:**

create database LibraryDB;

create table Books(

BookID int,

Title varchar(255),

Author varchar(255),

Genre varchar(255),

CONSTRAINT PK\_Books PRIMARY KEY (BookID)

);

create table Patrons(

PatronID int,

Name varchar(255),

MembershipStatus varchar(255),

CONSTRAINT PK\_Patrons PRIMARY KEY (PatronID)

);

create table Loans(

LoanID int,

BookID int,

PatronID int,

LoanDate date,

ReturnDate date,

CONSTRAINT PK\_Loans PRIMARY KEY (LoanID),

CONSTRAINT foreign\_loans1 FOREIGN KEY (BookID) REFERENCES Books(BookID),

CONSTRAINT foreign\_loans2 FOREIGN KEY (PatronID) REFERENCES patrons(PatronID)

);

INSERT INTO Books values ('1','To Kill a Mockingbird','Harper Lee','Fiction'),

('2','1984','George Orwell','Fiction'),

('3','Pride and Prejudice','Jane Austen','Fiction'),

('4','The Catcher in the Rye','J.D. Salinger','Fiction'),

('5','The Hobbit','J.R.R. Tolkien','Fantasy');

INSERT INTO Patrons values('1','Alice Johnson','Active'),

('2','Bob Smith','Active'),

('3','Carol Brown','InActive'),

('4','David Lee','Active'),

('5','Emma Davis','InActive');

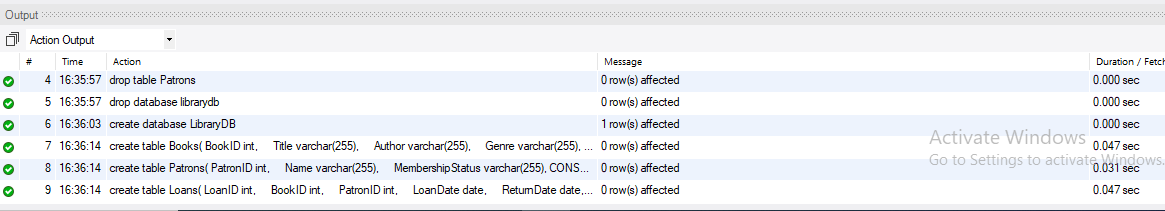
INSERT INTO Loans values('1','1','1','2023-01-15','2023-02-15'),

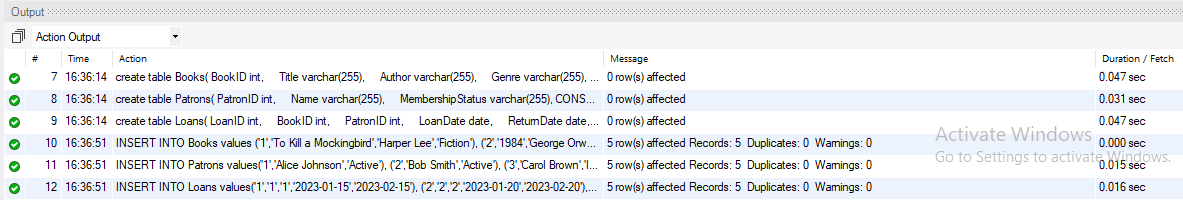
('2','2','2','2023-01-20','2023-02-20'),

('3','3','3','2023-01-25','2023-02-25'),

('4','4','4','2023-01-30','2023-02-28'),

('5','5','5','2023-02-05','2023-03-05');





* Write an SQL query to find the total number of books in each genre.

**Code:**

SELECT Genre,count(Genre) as Genre\_Count

from Books

group by Genre;

**Output Screenshot:**

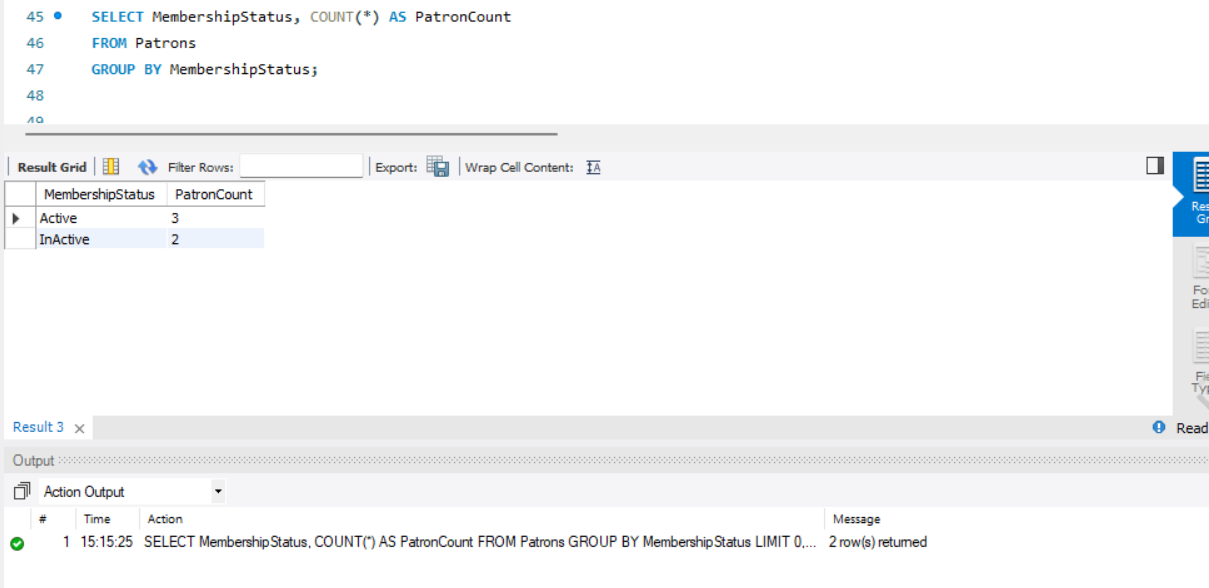
* Write an SQL query to find the number of active and inactive patrons.

**Code:**

SELECT MembershipStatus, COUNT(\*) AS PatronCount

FROM Patrons

GROUP BY MembershipStatus;

**Output Screenshot:**

* Write an SQL query to find the average loan duration for each book title (in days).

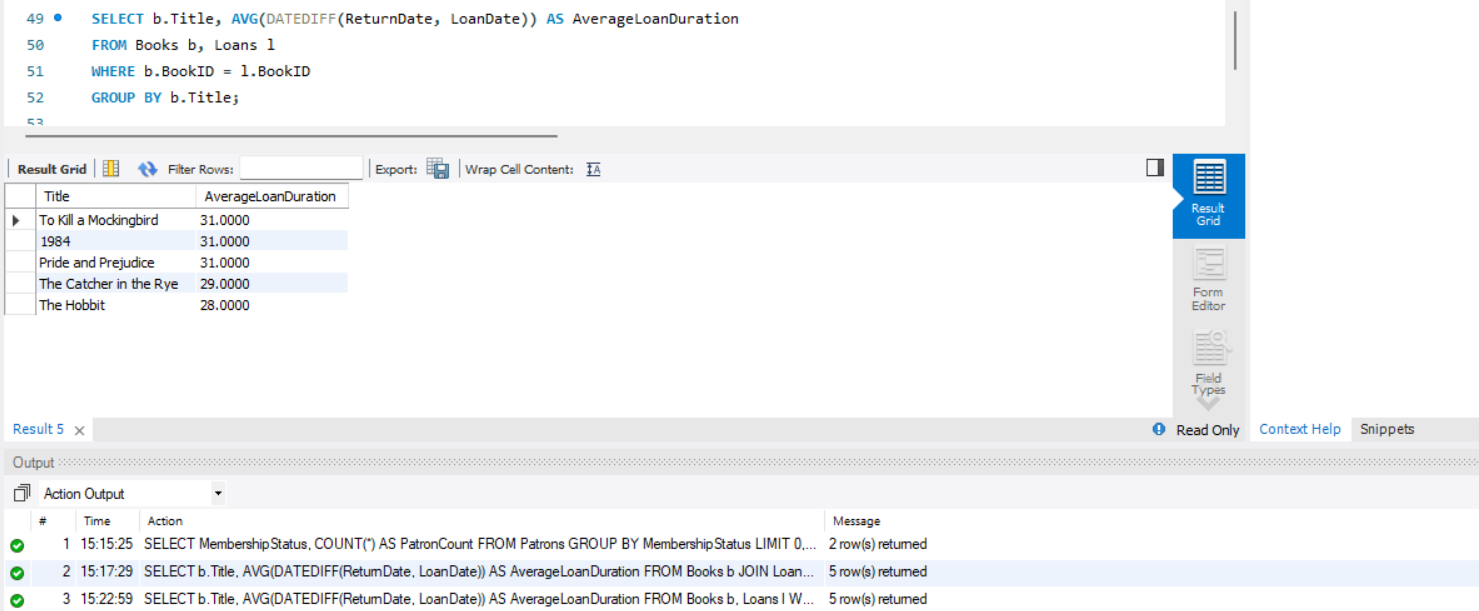
**Code:**

SELECT b.Title, AVG(DATEDIFF(ReturnDate, LoanDate)) AS AverageLoanDuration

FROM Books b, Loans l

WHERE b.BookID = l.BookID

GROUP BY b.Title;

**My output ScreenShot:**

* Write an SQL query to find the total number of loans made by each patron.

**Code:**

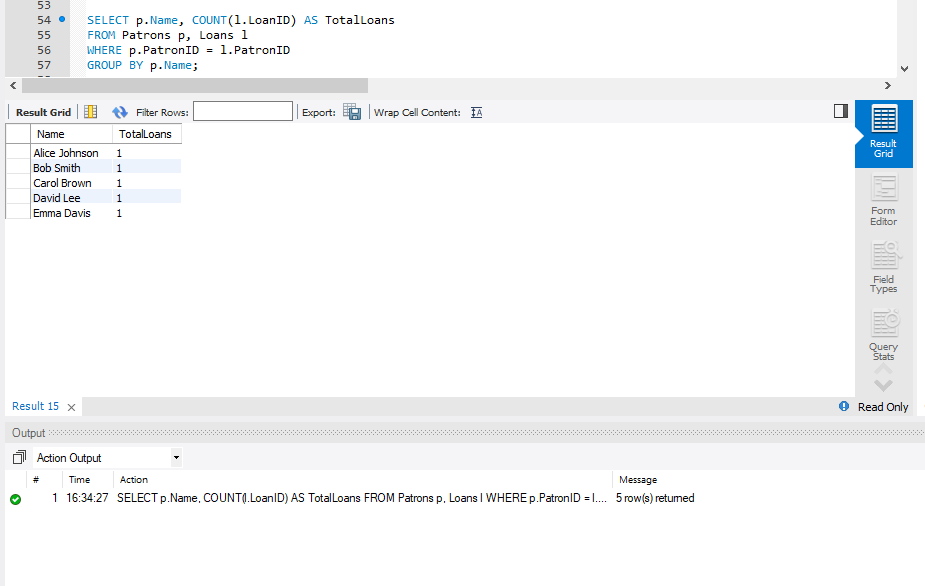
SELECT p.Name, COUNT(l.LoanID) AS TotalLoans

FROM Patrons p, Loans l

WHERE p.PatronID = l.PatronID

GROUP BY p.Name;

**My Output ScreenShot:**

****

* Write an SQL query to find the patron with the most loans and the number of loans they have.

**Code:**

SELECT Patrons.Name, COUNT(Loans.LoanID) AS NumberOfLoans

FROM Patrons, Loans

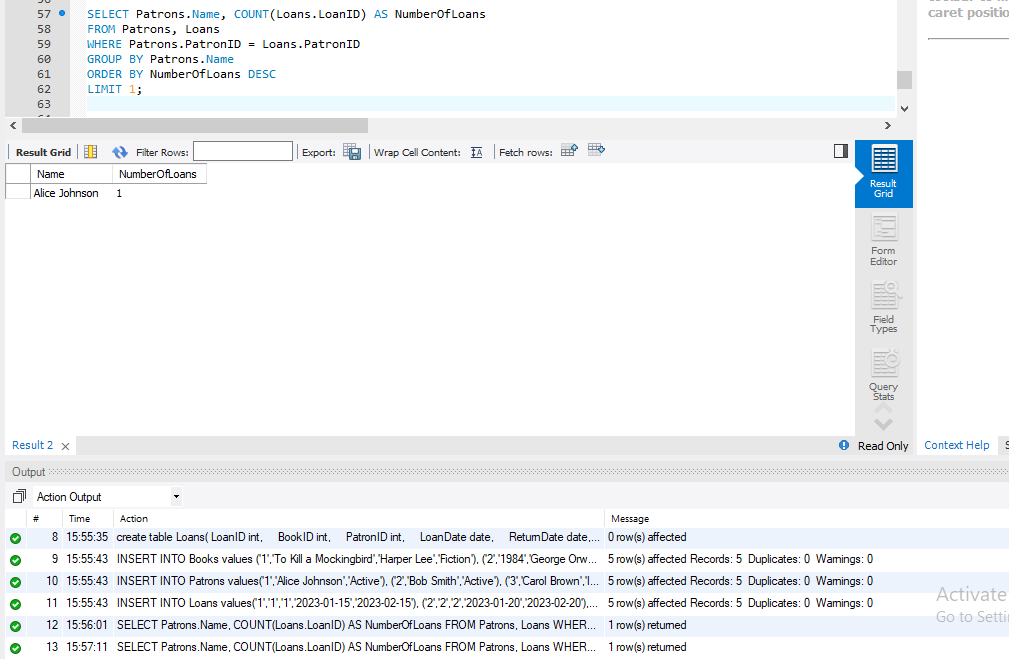
WHERE Patrons.PatronID = Loans.PatronID

GROUP BY Patrons.Name

ORDER BY NumberOfLoans DESC

LIMIT 1;

**My Output ScreenShot:**

****

* Write an SQL query to find the genre with the highest number of books and the number of books in that genre.

**Code:**

SELECT Genre, COUNT(BookID) AS NumberOfBooks

FROM Books

GROUP BY Genre

HAVING COUNT(BookID) = (

SELECT MAX(CountBooks)

FROM (

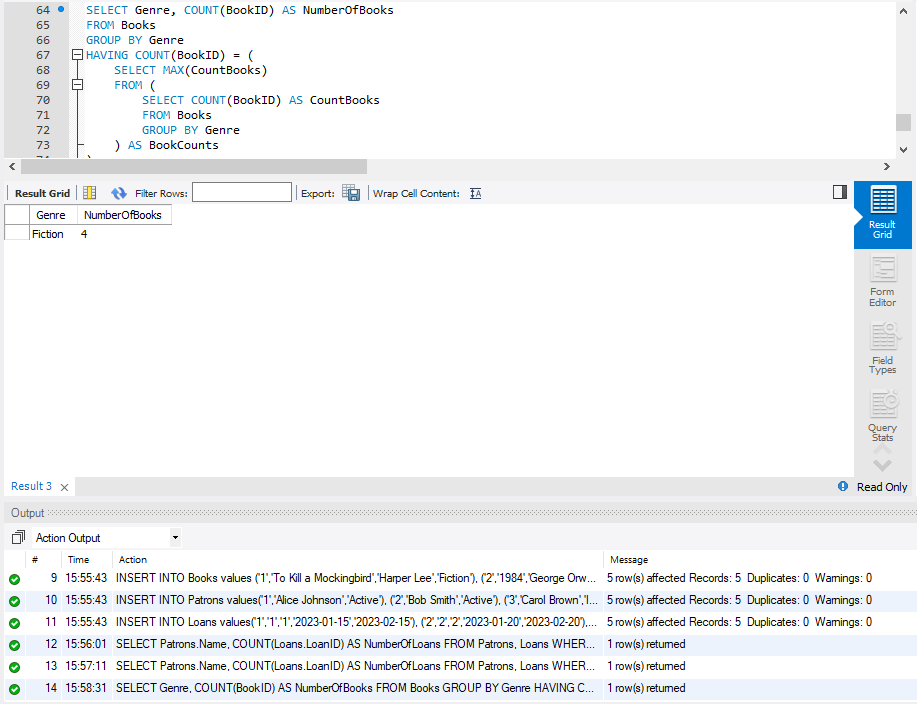
SELECT COUNT(BookID) AS CountBooks

FROM Books

GROUP BY Genre

) AS BookCounts

);

**My Output ScreenShot:**

**Lab task 2:**

**Formulate SQL queries for following information needs and execute them in MySQL server.**

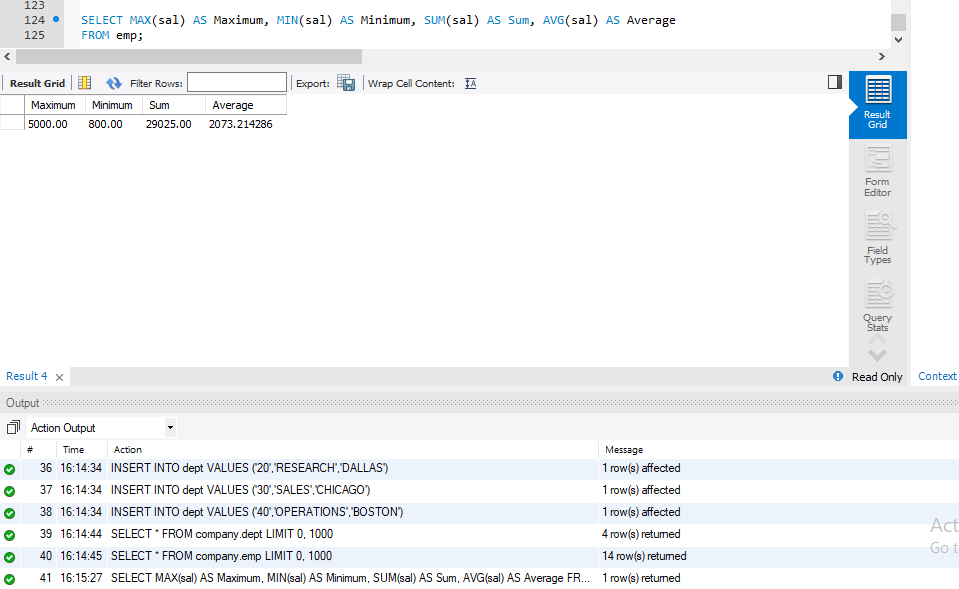
* Find the highest, lowest, sum and average salary of all employees. Label the columns as Maximum, Minimum, Sum and Average respectively. Save your query.

**Code:**

SELECT MAX(sal) AS Maximum, MIN(sal) AS Minimum, SUM(sal) AS Sum, AVG(sal) AS Average

FROM emp;

**My Output Screenshot:**

****

* Find the highest, lowest, sum and average salary for each job type. Label the columns as Maximum, Minimum, Sum and Average respectively. Save your query.

**Code:**

SELECT job,

MAX(sal) AS Maximum,

MIN(sal) AS Minimum,

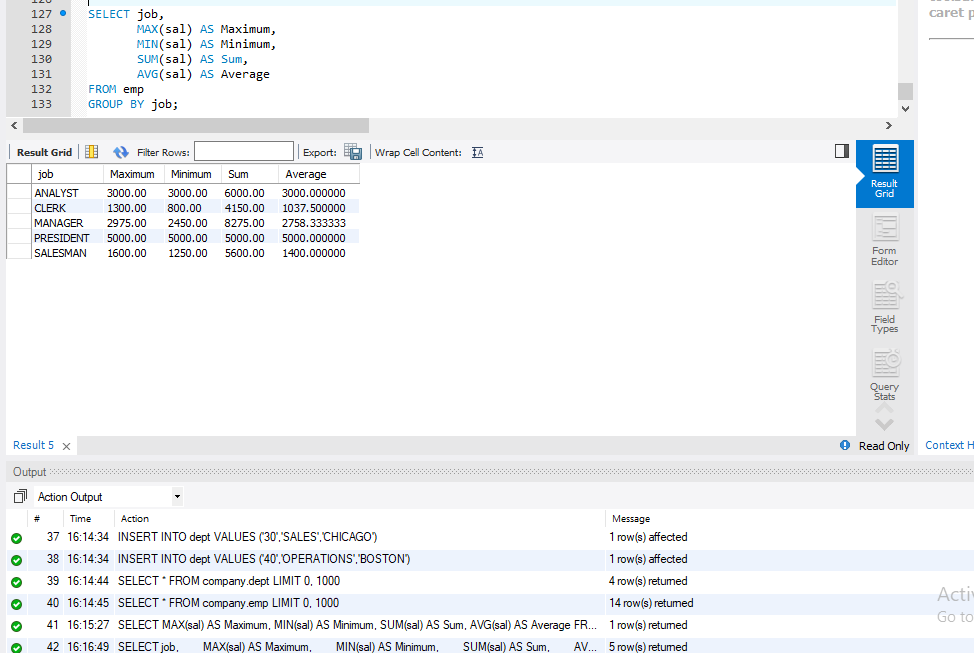
SUM(sal) AS Sum,

AVG(sal) AS Average

FROM emp

GROUP BY job;

**My Output Screenshot:**



* Lists the number of employees in each job, sorted high to low.

**Code:**

SELECT job,

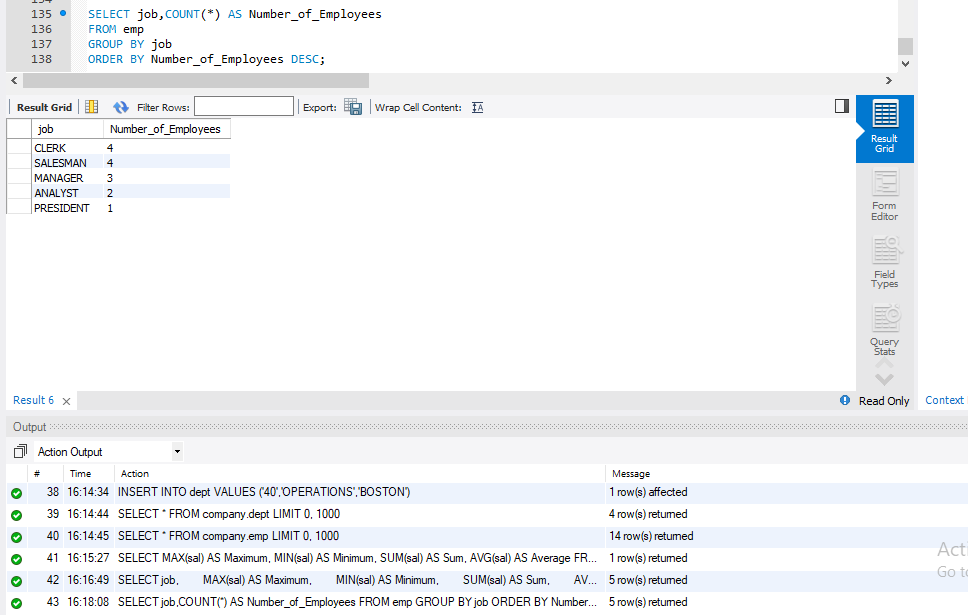
COUNT(\*) AS Number\_of\_Employees

FROM emp

GROUP BY job

ORDER BY Number\_of\_Employees DESC;

**My Output Screenshot:**



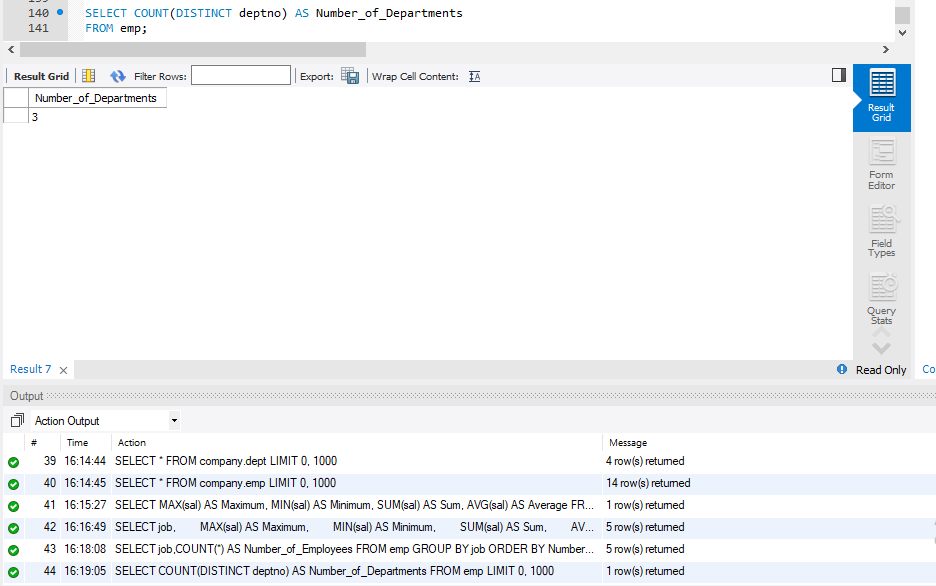
* Display the number of distinct department values in the EMPLOYEES table.

**Code:**

SELECT COUNT(DISTINCT deptno) AS Number\_of\_Departments

FROM emp;

**My Output Screenshot:**



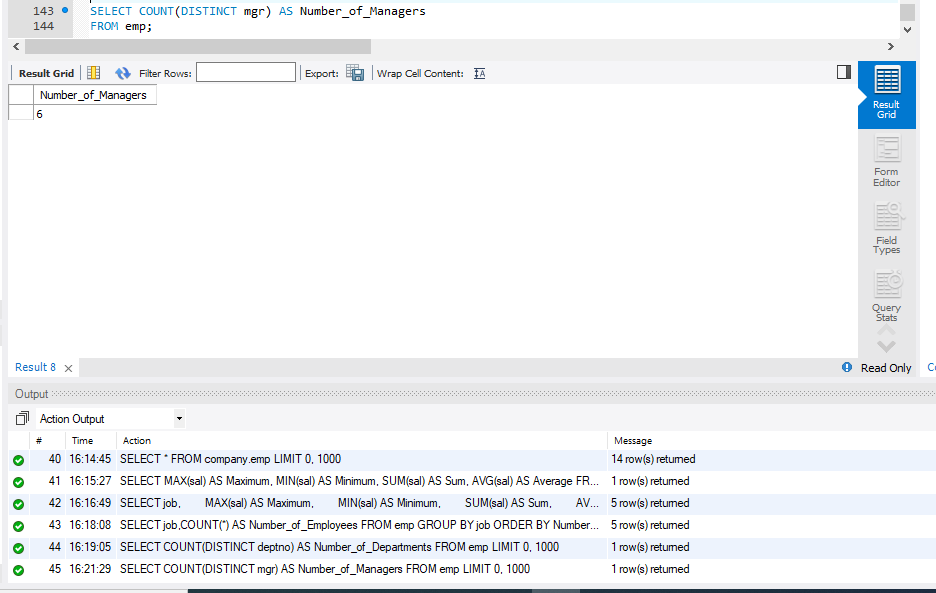
* Determine the number of managers without listing them. Label the column as Number of Mangers.

**Code:**

SELECT COUNT(DISTINCT mgr) AS Number\_of\_Managers

FROM emp;

**My Output Screenshot:**



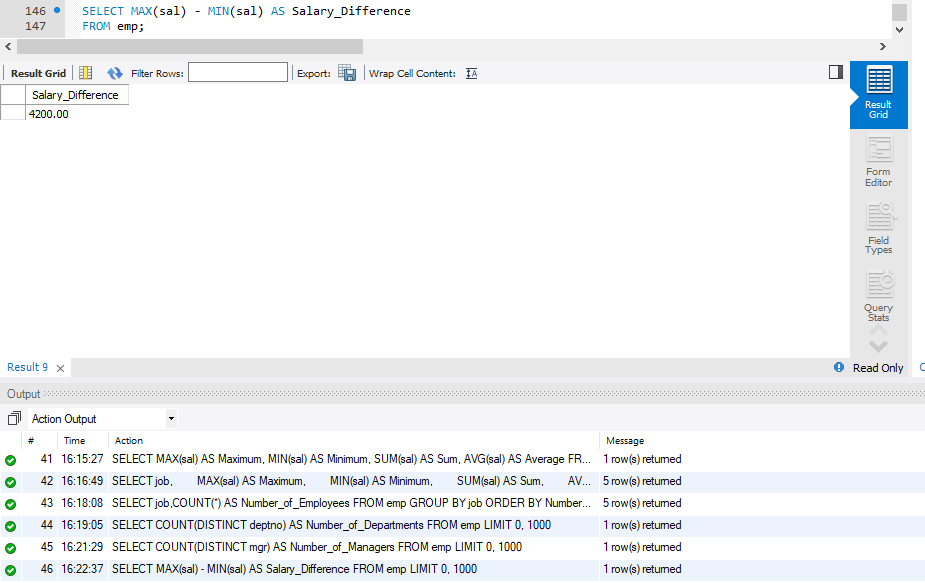
* Find the difference between highest and lowest salaries.

**Code:**

SELECT MAX(sal) - MIN(sal) AS Salary\_Difference

FROM emp;

**My Output Screenshot:**



* Formulate a query to display the manager number and the salary of the lowest-paid. employee for that manager. Exclude any groups where the minimum salary is 6000 or less. Sort the output in descending order of salary.

**Code:**

SELECT mgr, MIN(sal) AS Lowest\_Salary

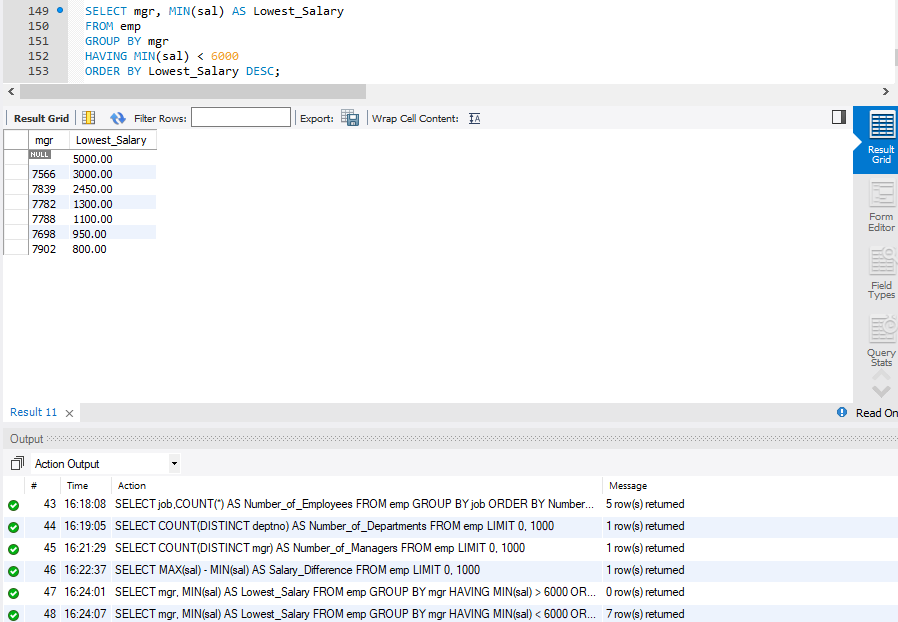
FROM emp

GROUP BY mgr

HAVING MIN(sal) < 6000

ORDER BY Lowest\_Salary DESC;

**My Output Screenshot:**



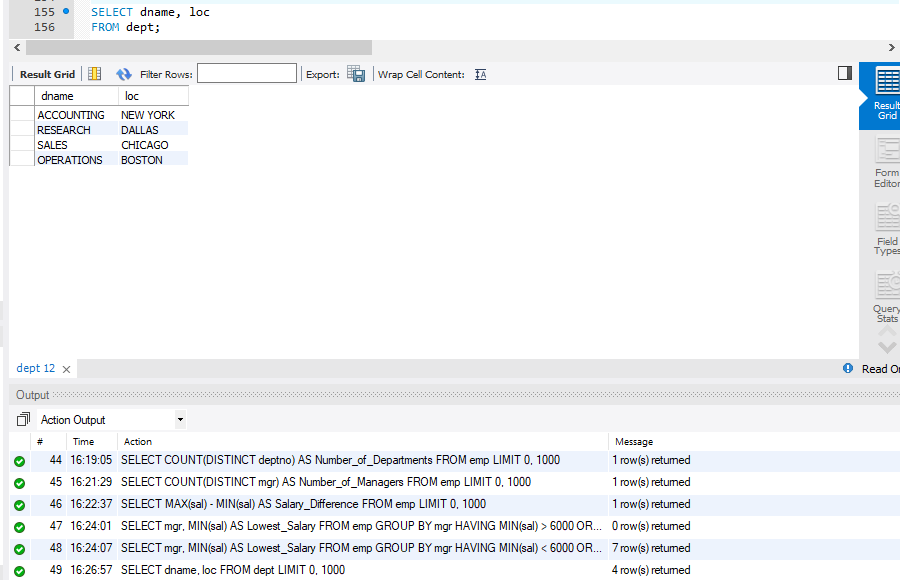
* Give the department names and their locations.

**Code:**

SELECT dname, loc

FROM dept;

**My Output Screenshot:**



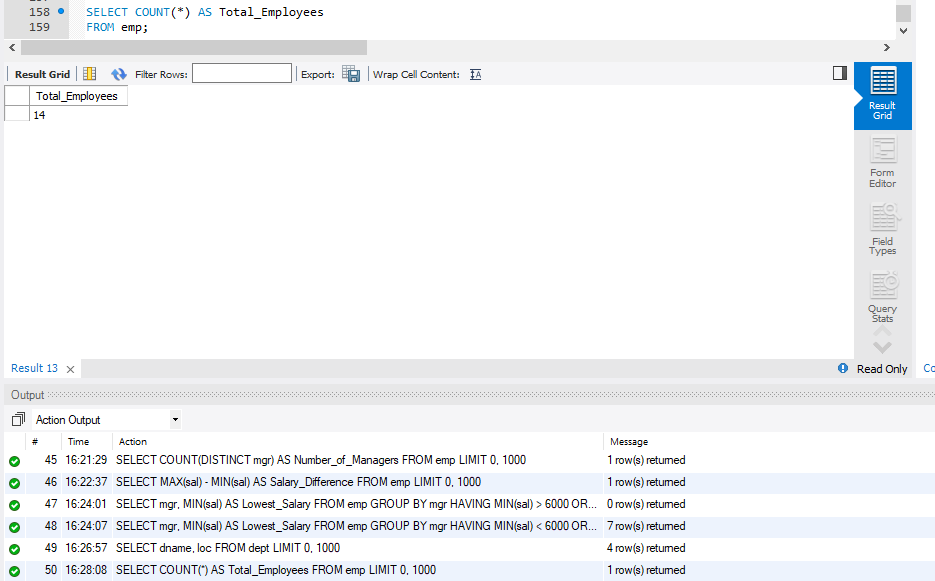
* Retrieve total no. of employees in the Company.

**Code:**

SELECT COUNT(\*) AS Total\_Employees

FROM emp;

**My Output Screenshot:**

****

**Deliverables**

Save all queries and their results in the word document that you are executing, including examples and the questions. Relational algebra queries expressions save in plain text file or word doc .Upload this document to LMS. Late submissions will not be accepted.