Name: Bryce Ferreira
Roll No.: 2203129
En No: MITU20BTCS0076
Class: TY CSE CORE 3 Batch A

Exp- 02

Part A

Practical Objective:

i) To learn SQL Aggregate Functions
ii) To understand Group by, Having Clause, Order by clause
iii) To learn Nested Queries
iv) To understand Joins

Prerequisite: Understanding of basic DDL and DML commands

Software: MySQL

CO Mapping:
CO1: Apply the concepts of database design and SQL.

Practical Outcomes: At the end of this practical student will be able to:

1. Perform Aggerate functions.
2. Perform Nested Queries
3. Apply joins on table.

Theory:

Aggregate Functions

AVG: returns average value
    Avg(<ColumnName>)

MIN: returns minimum value
    min(<ColumnName>)

COUNT: returns no of rows where expression is not NULL
    count(<ColumnName>)

COUNT(*): returns no of rows in the table including duplicates and those with NULL
    count(*)

MAX: returns maximum value
    max(<ColumnName>)

SUM: returns sum of the values
    sum(<ColumnName>)

ORDER BY:

The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default. The basic syntax of the ORDER BY clause is as follows –

```
SELECT column-list
FROM table_name
[WHERE condition]
```

```
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort that column should be in the column-list.

Group by clause: this optional clause tells Oracle to group rows based on distinct values that exists for specified columns.

```
Select <columnname 1><columnname 2>...<columnname n>,
Aggregate_function(<expression>) from tablename Where <condition>
Group by <columnname 1><columnname 2>...<columnname n>;
```

Having clause: imposes a condition on group by clause.

```
 Select <columnname 1><columnname 2>...<columnname n>,
Aggregate_function(<expression>) from tablename Where <condition>
Group by <columnname 1><columnname 2>...<columnname n>
Having <condition>;
```

Subquery:

In SQL a Subquery can be simply defined as a query within another query. In other words we can say that a Subquery is a query that is embedded in WHERE clause of another SQL query. Important rules for Subqueries:

- You can place the Subquery in a number of SQL clauses: WHERE clause, HAVING clause, FROM clause.
- Subqueries can be used with SELECT, UPDATE, INSERT, DELETE statements along with expression operator. It could be equality operator or comparison operator such as =, >, =, <= and Like operator.
- A subquery is a query within another query. The outer query is called as **main query** and inner query is called as **subquery**.
- The subquery generally executes first, and its output is used to complete the query condition for the main or outer query.
- Subquery must be enclosed in parentheses.
- Subqueries are on the right side of the comparison operator.
- ORDER BY command **cannot** be used in a Subquery.
- GROUPBY command can be used to perform same function as ORDER BY command.

SQL JOIN

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

SQL INNER JOIN

The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.

Syntax

SELECT *column_name(s)*
FROM *table1*
INNER JOIN *table2*
ON *table1.column_name=table2.column_name*;

Inner join can also be categorized as theta, equi and natural join.

SQL LEFT JOIN

The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

Syntax

SELECT *column_name(s)*
FROM *table1*
LEFT JOIN *table2*
ON *table1.column_name=table2.column_name*;

SQL RIGHT JOIN

The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

Syntax

SELECT *column_name(s)*
FROM *table1*
RIGHT JOIN *table2*
ON *table1.column_name=table2.column_name*;

SQL FULL OUTER JOIN

The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).

The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.

Syntax

SELECT *column_name(s)*
FROM *table1*
FULL OUTER JOIN *table2*
ON *table1.column_name=table2.column_name*;

Procedure:

1. Formulate the query for given problem.
2. Write the SQL query with proper input.
3. Execute the query.

Practice Exercise:

1. Display the total expenditure of company on the salary of employees.
2. Find average salary of clerks.
3. Find average salary of managers and salesman.
4. Find employees with maximum annual income.
5. Find the employee with minimum monthly income.
6. Find the number of employees earning more than the average salary of employees.
7. List the details of the department where maximum number of emps are working.
8. Find the total salary department wise.
9. Find total salary average salary Job wise.
10. Find the name of department taking maximum salary.
11. Find name of department taking minimum salary.
12. Write a query to display all the information of the employees whose salary is within the range 1000 and 3000.
13. Write a query to display all the information of the employees whose salary is within the range of smallest salary and 2500.
14. Display all the information of an employee whose id is any of the number 134, 159 and 183.
15. Write a query to display the name (first name and last name) for those employees who gets more salary than the employee whose ID is 163
16. Display the name of those employees who are getting highest salary.
17. Delete those employees who joined the company before 31-dec-82 while there dept location is ‘NEW YORK’ or ‘CHICAGO’.
18. Display employee name, job, deptname, location for all who are working as managers.
19. Display those employees who are working in the same dept where his manager is working.
20. Display employee name, his job, his dept name, his manager name, his sal and arrange it based on salary under department wise.

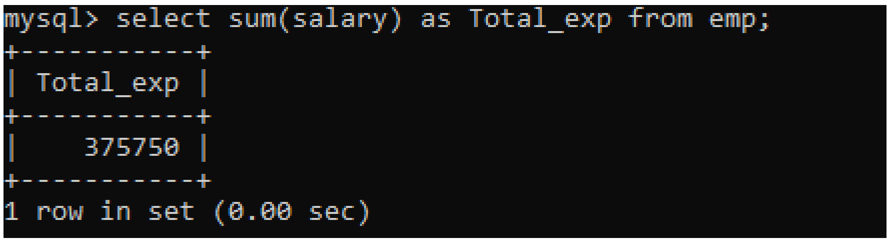
Instructions:

1. Write and execute the query in Oracle SQL server.
2. Paste the snapshot of the output in input & output section.

Part B

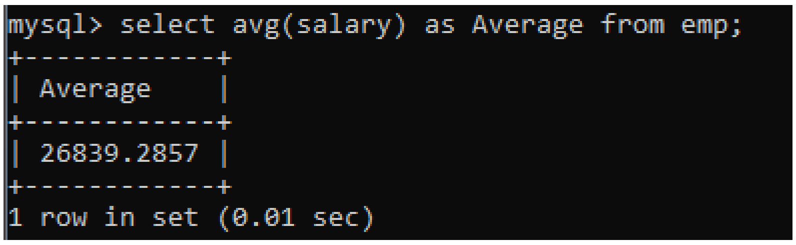
Code and Output:

1.



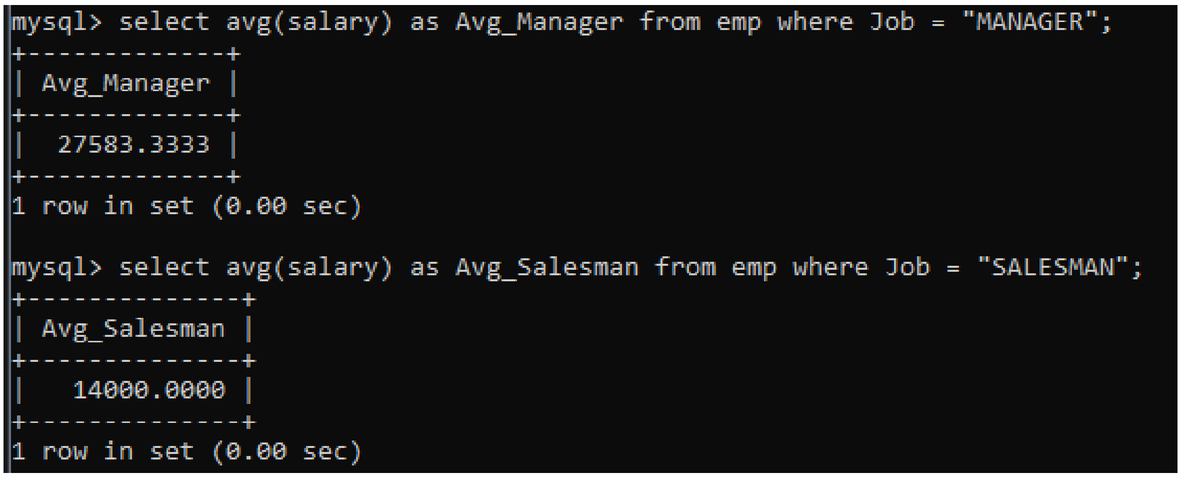
```
mysql> select sum(salary) as Total_exp from emp;
+-----------+
| Total_exp |
+-----------+
|    375750 |
+-----------+
1 row in set (0.00 sec)
```

2.



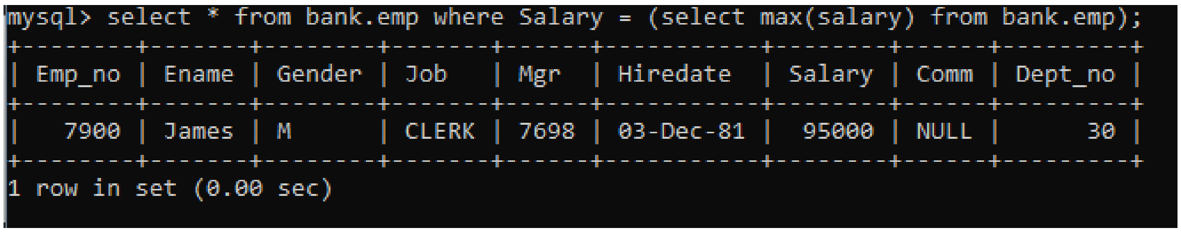
```
mysql> select avg(salary) as Average from emp;
+------------+
| Average    |
+------------+
| 26839.2857 |
+------------+
1 row in set (0.01 sec)
```

3.



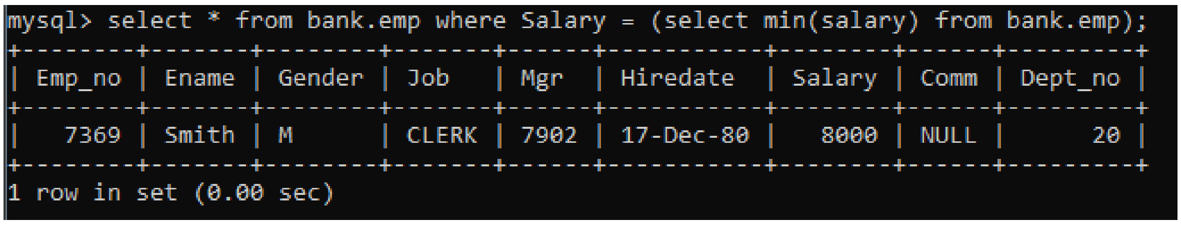
```
mysql> select avg(salary) as Avg_Manager from emp where Job = "MANAGER";
+-------------+
| Avg_Manager |
+-------------+
|  27583.3333 |
+-------------+
1 row in set (0.00 sec)

mysql> select avg(salary) as Avg_Salesman from emp where Job = "SALESMAN";
+--------------+
| Avg_Salesman |
+--------------+
|   14000.0000 |
+--------------+
1 row in set (0.00 sec)
```

4.

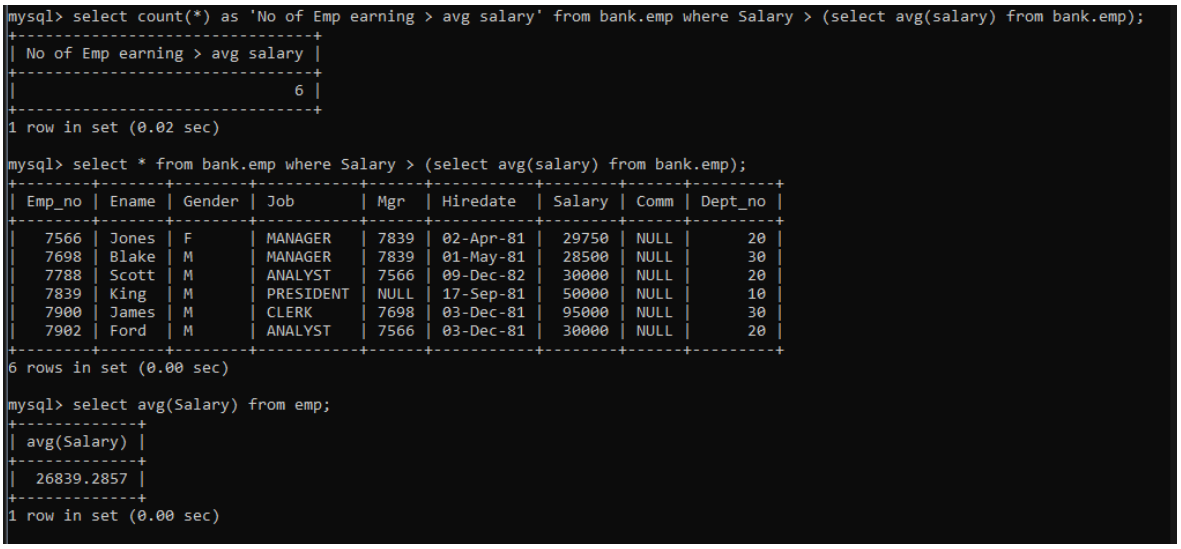


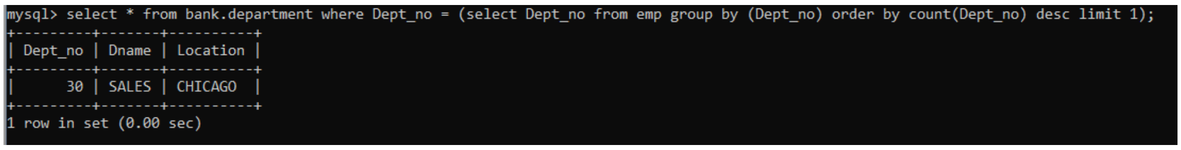
```
mysql> select * from bank.emp where Salary = (select max(salary) from bank.emp);
+--------+-------+--------+-------+------+-----------+--------+------+---------+
| Emp_no | Ename | Gender | Job   | Mgr  | Hiredate  | Salary | Comm | Dept_no |
+--------+-------+--------+-------+------+-----------+--------+------+---------+
|   7900 | James | M      | CLERK | 7698 | 03-Dec-81 |  95000 | NULL |      30 |
+--------+-------+--------+-------+------+-----------+--------+------+---------+
1 row in set (0.00 sec)
```

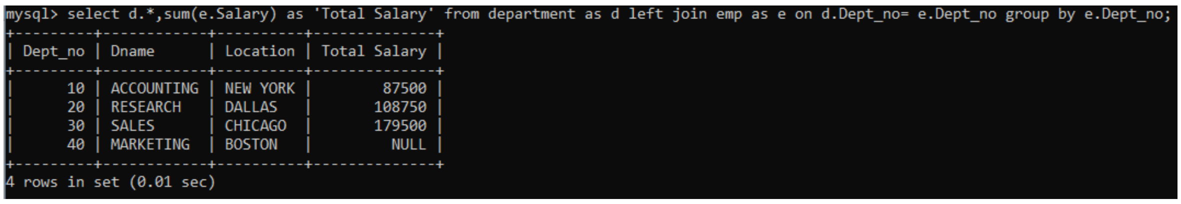
5.

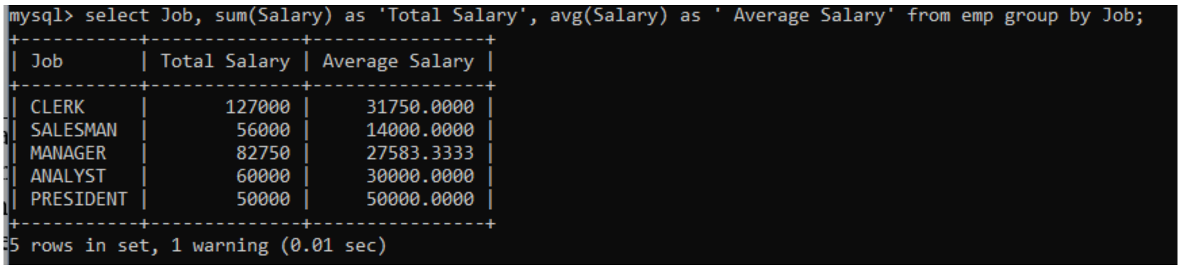


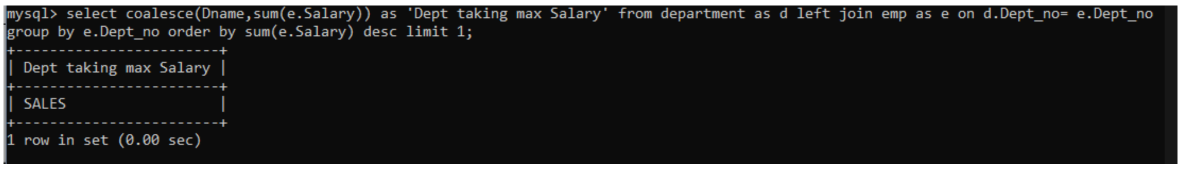
```
mysql> select * from bank.emp where Salary = (select min(salary) from bank.emp);
+--------+-------+--------+-------+------+-----------+--------+------+---------+
| Emp_no | Ename | Gender | Job   | Mgr  | Hiredate  | Salary | Comm | Dept_no |
+--------+-------+--------+-------+------+-----------+--------+------+---------+
|   7369 | Smith | M      | CLERK | 7902 | 17-Dec-80 |   8000 | NULL |      20 |
+--------+-------+--------+-------+------+-----------+--------+------+---------+
1 row in set (0.00 sec)
```

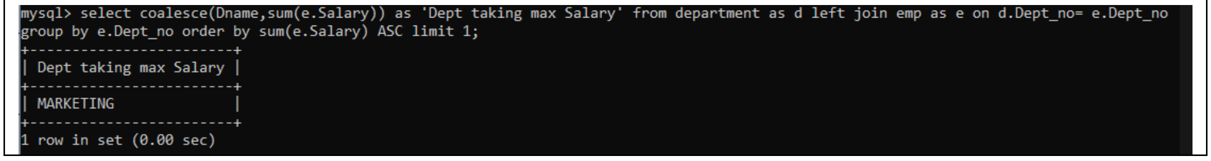
6.

```
mysql> select count(*) as 'No of Emp earning > avg salary' from bank.emp where Salary > (select avg(salary) from bank.emp);
+--------------------------------+
| No of Emp earning > avg salary |
+--------------------------------+
|                              6 |
+--------------------------------+
1 row in set (0.02 sec)

mysql> select * from bank.emp where Salary > (select avg(salary) from bank.emp);
+--------+-------+--------+-----------+------+-----------+--------+------+---------+
| Emp_no | Ename | Gender | Job       | Mgr  | Hiredate  | Salary | Comm | Dept_no |
+--------+-------+--------+-----------+------+-----------+--------+------+---------+
|   7566 | Jones | F      | MANAGER   | 7839 | 02-Apr-81 |  29750 | NULL |      20 |
|   7698 | Blake | M      | MANAGER   | 7839 | 01-May-81 |  28500 | NULL |      30 |
|   7788 | Scott | M      | ANALYST   | 7566 | 09-Dec-82 |  30000 | NULL |      20 |
|   7839 | King  | M      | PRESIDENT | NULL | 17-Sep-81 |  50000 | NULL |      10 |
|   7900 | James | M      | CLERK     | 7698 | 03-Dec-81 |  95000 | NULL |      30 |
|   7902 | Ford  | M      | ANALYST   | 7566 | 03-Dec-81 |  30000 | NULL |      20 |
+--------+-------+--------+-----------+------+-----------+--------+------+---------+
6 rows in set (0.00 sec)

mysql> select avg(Salary) from emp;
+-------------+
| avg(Salary) |
+-------------+
|  26839.2857 |
+-------------+
1 row in set (0.00 sec)
```

7.

```
mysql> select * from bank.department where Dept_no = (select Dept_no from emp group by (Dept_no) order by count(Dept_no) desc limit 1);
+---------+-------+----------+
| Dept_no | Dname | Location |
+---------+-------+----------+
|      30 | SALES | CHICAGO  |
+---------+-------+----------+
1 row in set (0.00 sec)
```

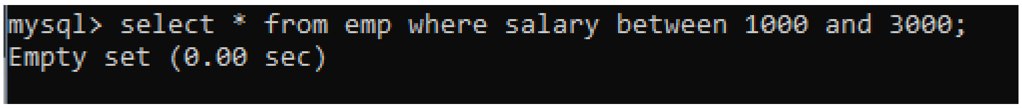
8.

```
mysql> select d.*,sum(e.Salary) as 'Total Salary' from department as d left join emp as e on d.Dept_no= e.Dept_no group by e.Dept_no;
+---------+------------+----------+--------------+
| Dept_no | Dname      | Location | Total Salary |
+---------+------------+----------+--------------+
|      10 | ACCOUNTING | NEW YORK |        87500 |
|      20 | RESEARCH   | DALLAS   |       108750 |
|      30 | SALES      | CHICAGO  |       179500 |
|      40 | MARKETING  | BOSTON   |         NULL |
+---------+------------+----------+--------------+
4 rows in set (0.01 sec)
```

9.

```
mysql> select Job, sum(Salary) as 'Total Salary', avg(Salary) as ' Average Salary' from emp group by Job;
+-----------+--------------+-----------------+
| Job       | Total Salary | Average Salary  |
+-----------+--------------+-----------------+
| CLERK     |       127000 |      31750.0000 |
| SALESMAN  |        56000 |      14000.0000 |
| MANAGER   |        82750 |      27583.3333 |
| ANALYST   |        60000 |      30000.0000 |
| PRESIDENT |        50000 |      50000.0000 |
+-----------+--------------+-----------------+
5 rows in set, 1 warning (0.01 sec)
```

10.

```
mysql> select coalesce(Dname,sum(e.Salary)) as 'Dept taking max Salary' from department as d left join emp as e on d.Dept_no= e.Dept_no
group by e.Dept_no order by sum(e.Salary) desc limit 1;
+------------------------+
| Dept taking max Salary |
+------------------------+
| SALES                  |
+------------------------+
1 row in set (0.00 sec)
```

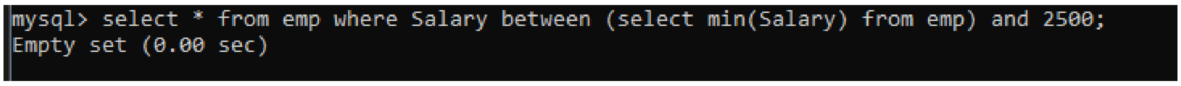
11.

```
mysql> select coalesce(Dname,sum(e.Salary)) as 'Dept taking max Salary' from department as d left join emp as e on d.Dept_no= e.Dept_no
group by e.Dept_no order by sum(e.Salary) ASC limit 1;
+------------------------+
| Dept taking max Salary |
+------------------------+
| MARKETING              |
+------------------------+
1 row in set (0.00 sec)
```

12.



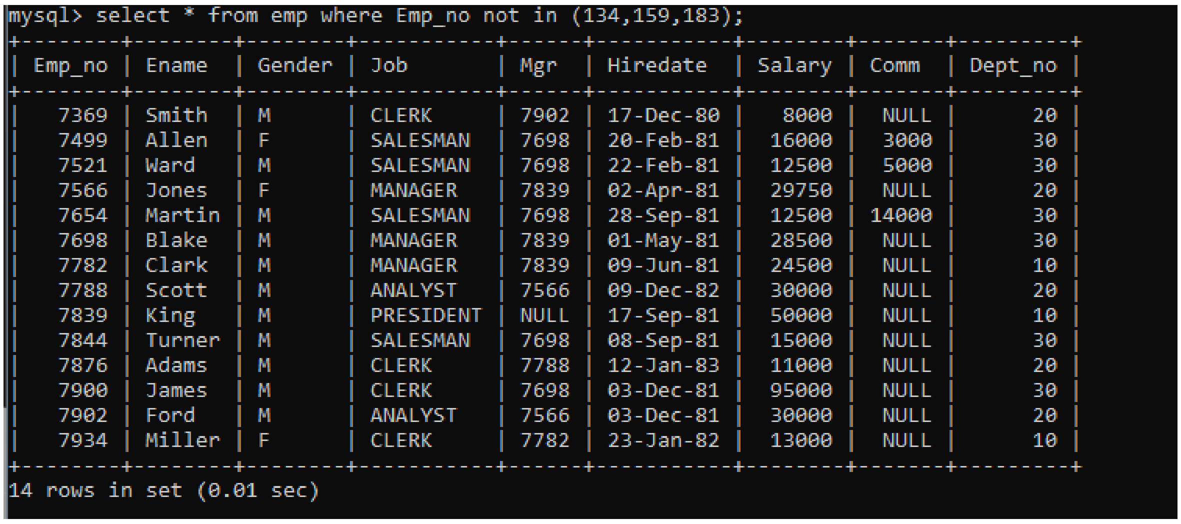
```
mysql> select * from emp where salary between 1000 and 3000;
Empty set (0.00 sec)
```

13.



```
mysql> select * from emp where Salary between (select min(Salary) from emp) and 2500;
Empty set (0.00 sec)
```

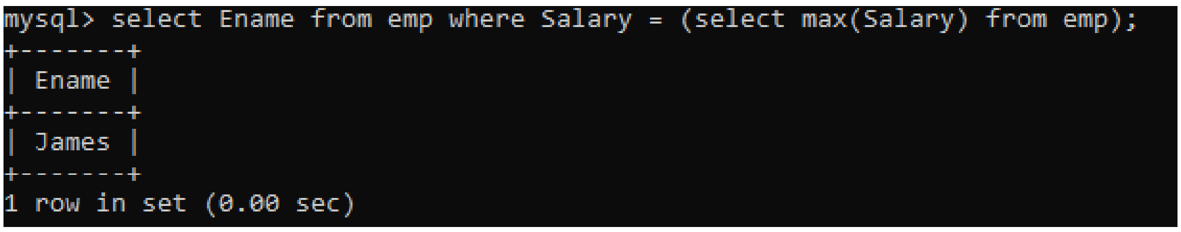
14.



```
mysql> select * from emp where Emp_no not in (134,159,183);
+--------+--------+--------+-----------+------+-----------+--------+-------+---------+
| Emp_no | Ename  | Gender | Job       | Mgr  | Hiredate  | Salary | Comm  | Dept_no |
+--------+--------+--------+-----------+------+-----------+--------+-------+---------+
|   7369 | Smith  | M      | CLERK     | 7902 | 17-Dec-80 |   8000 | NULL  |      20 |
|   7499 | Allen  | F      | SALESMAN  | 7698 | 20-Feb-81 |  16000 | 3000  |      30 |
|   7521 | Ward   | M      | SALESMAN  | 7698 | 22-Feb-81 |  12500 | 5000  |      30 |
|   7566 | Jones  | F      | MANAGER   | 7839 | 02-Apr-81 |  29750 | NULL  |      20 |
|   7654 | Martin | M      | SALESMAN  | 7698 | 28-Sep-81 |  12500 | 14000 |      30 |
|   7698 | Blake  | M      | MANAGER   | 7839 | 01-May-81 |  28500 | NULL  |      30 |
|   7782 | Clark  | M      | MANAGER   | 7839 | 09-Jun-81 |  24500 | NULL  |      10 |
|   7788 | Scott  | M      | ANALYST   | 7566 | 09-Dec-82 |  30000 | NULL  |      20 |
|   7839 | King   | M      | PRESIDENT | NULL | 17-Sep-81 |  50000 | NULL  |      10 |
|   7844 | Turner | M      | SALESMAN  | 7698 | 08-Sep-81 |  15000 | NULL  |      30 |
|   7876 | Adams  | M      | CLERK     | 7788 | 12-Jan-83 |  11000 | NULL  |      20 |
|   7900 | James  | M      | CLERK     | 7698 | 03-Dec-81 |  95000 | NULL  |      30 |
|   7902 | Ford   | M      | ANALYST   | 7566 | 03-Dec-81 |  30000 | NULL  |      20 |
|   7934 | Miller | F      | CLERK     | 7782 | 23-Jan-82 |  13000 | NULL  |      10 |
+--------+--------+--------+-----------+------+-----------+--------+-------+---------+
14 rows in set (0.01 sec)
```

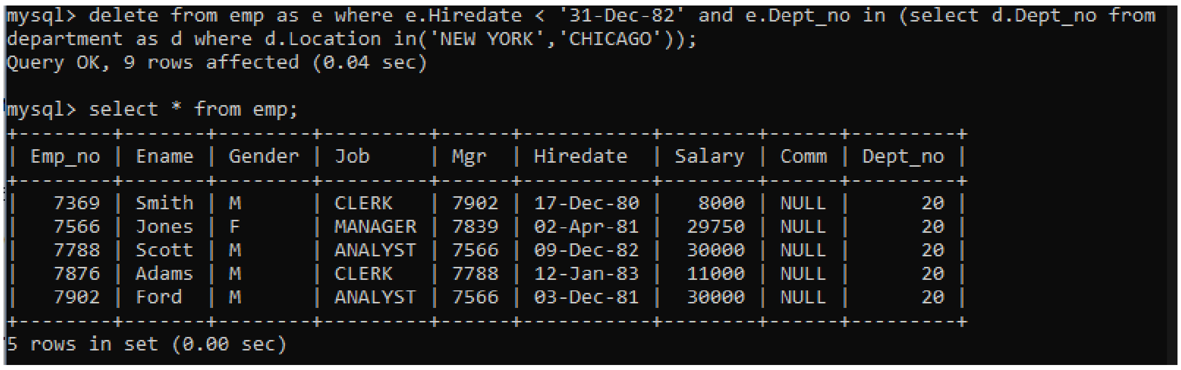
15.

16.



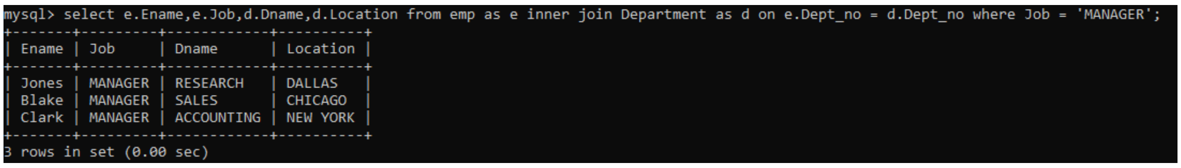
```
mysql> select Ename from emp where Salary = (select max(Salary) from emp);
+-------+
| Ename |
+-------+
| James |
+-------+
1 row in set (0.00 sec)
```

17.



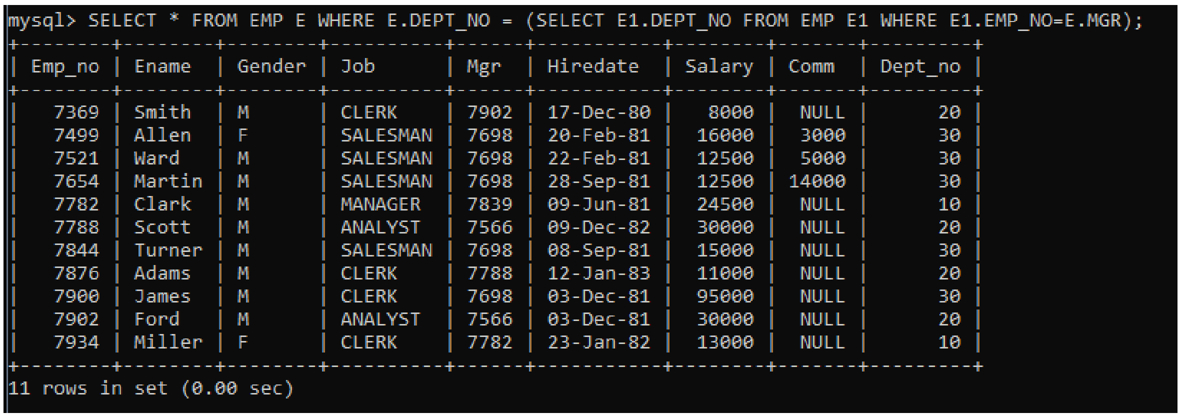
```
mysql> delete from emp as e where e.Hiredate < '31-Dec-82' and e.Dept_no in (select d.Dept_no from
department as d where d.Location in('NEW YORK','CHICAGO'));
Query OK, 9 rows affected (0.04 sec)

mysql> select * from emp;
+--------+-------+--------+---------+------+-----------+--------+------+---------+
| Emp_no | Ename | Gender | Job     | Mgr  | Hiredate  | Salary | Comm | Dept_no |
+--------+-------+--------+---------+------+-----------+--------+------+---------+
|   7369 | Smith | M      | CLERK   | 7902 | 17-Dec-80 |   8000 | NULL |      20 |
|   7566 | Jones | F      | MANAGER | 7839 | 02-Apr-81 |  29750 | NULL |      20 |
|   7788 | Scott | M      | ANALYST | 7566 | 09-Dec-82 |  30000 | NULL |      20 |
|   7876 | Adams | M      | CLERK   | 7788 | 12-Jan-83 |  11000 | NULL |      20 |
|   7902 | Ford  | M      | ANALYST | 7566 | 03-Dec-81 |  30000 | NULL |      20 |
+--------+-------+--------+---------+------+-----------+--------+------+---------+
5 rows in set (0.00 sec)
```

18.



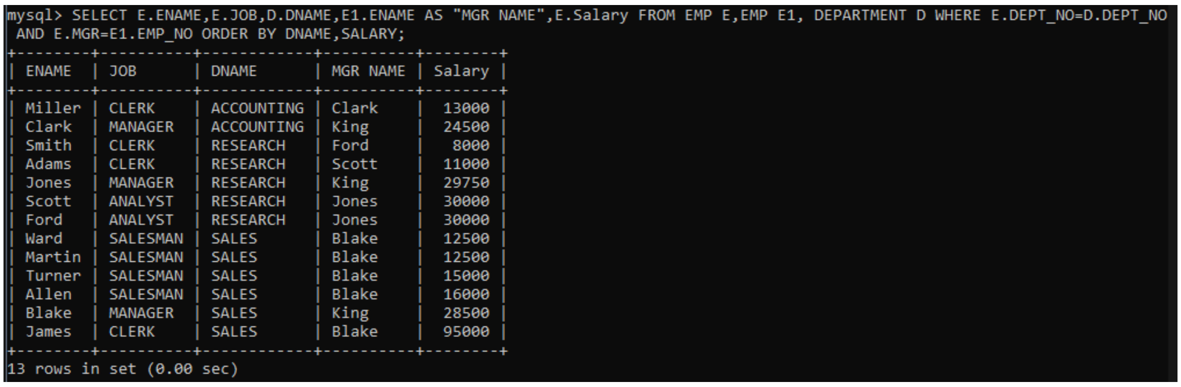
```
mysql> select e.Ename,e.Job,d.Dname,d.Location from emp as e inner join Department as d on e.Dept_no = d.Dept_no where Job = 'MANAGER';
+-------+---------+------------+----------+
| Ename | Job     | Dname      | Location |
+-------+---------+------------+----------+
| Jones | MANAGER | RESEARCH   | DALLAS   |
| Blake | MANAGER | SALES      | CHICAGO  |
| Clark | MANAGER | ACCOUNTING | NEW YORK |
+-------+---------+------------+----------+
3 rows in set (0.00 sec)
```

19.



```
mysql> SELECT * FROM EMP E WHERE E.DEPT_NO = (SELECT E1.DEPT_NO FROM EMP E1 WHERE E1.EMP_NO=E.MGR);
+--------+--------+--------+----------+------+-----------+--------+-------+---------+
| Emp_no | Ename  | Gender | Job      | Mgr  | Hiredate  | Salary | Comm  | Dept_no |
+--------+--------+--------+----------+------+-----------+--------+-------+---------+
|   7369 | Smith  | M      | CLERK    | 7902 | 17-Dec-80 |   8000 | NULL  |      20 |
|   7499 | Allen  | F      | SALESMAN | 7698 | 20-Feb-81 |  16000 | 3000  |      30 |
|   7521 | Ward   | M      | SALESMAN | 7698 | 22-Feb-81 |  12500 | 5000  |      30 |
|   7654 | Martin | M      | SALESMAN | 7698 | 28-Sep-81 |  12500 | 14000 |      30 |
|   7782 | Clark  | M      | MANAGER  | 7839 | 09-Jun-81 |  24500 | NULL  |      10 |
|   7788 | Scott  | M      | ANALYST  | 7566 | 09-Dec-82 |  30000 | NULL  |      20 |
|   7844 | Turner | M      | SALESMAN | 7698 | 08-Sep-81 |  15000 | NULL  |      30 |
|   7876 | Adams  | M      | CLERK    | 7788 | 12-Jan-83 |  11000 | NULL  |      20 |
|   7900 | James  | M      | CLERK    | 7698 | 03-Dec-81 |  95000 | NULL  |      30 |
|   7902 | Ford   | M      | ANALYST  | 7566 | 03-Dec-81 |  30000 | NULL  |      20 |
|   7934 | Miller | F      | CLERK    | 7782 | 23-Jan-82 |  13000 | NULL  |      10 |
+--------+--------+--------+----------+------+-----------+--------+-------+---------+
11 rows in set (0.00 sec)
```

20.



```
mysql> SELECT E.ENAME,E.JOB,D.DNAME,E1.ENAME AS "MGR NAME",E.Salary FROM EMP E,EMP E1, DEPARTMENT D WHERE E.DEPT_NO=D.DEPT_NO
 AND E.MGR=E1.EMP_NO ORDER BY DNAME,SALARY;
+--------+----------+------------+----------+--------+
| ENAME  | JOB      | DNAME      | MGR NAME | Salary |
+--------+----------+------------+----------+--------+
| Miller | CLERK    | ACCOUNTING | Clark    |  13000 |
| Clark  | MANAGER  | ACCOUNTING | King     |  24500 |
| Smith  | CLERK    | RESEARCH   | Ford     |   8000 |
| Adams  | CLERK    | RESEARCH   | Scott    |  11000 |
| Jones  | MANAGER  | RESEARCH   | King     |  29750 |
| Scott  | ANALYST  | RESEARCH   | Jones    |  30000 |
| Ford   | ANALYST  | RESEARCH   | Jones    |  30000 |
| Ward   | SALESMAN | SALES      | Blake    |  12500 |
| Martin | SALESMAN | SALES      | Blake    |  12500 |
| Turner | SALESMAN | SALES      | Blake    |  15000 |
| Allen  | SALESMAN | SALES      | Blake    |  16000 |
| Blake  | MANAGER  | SALES      | King     |  28500 |
| James  | CLERK    | SALES      | Blake    |  95000 |
+--------+----------+------------+----------+--------+
13 rows in set (0.00 sec)
```

Observation & Learning:
Write your observation and learning after performing the task.

Conclusion:
Write statement of conclusion here.

Questions:
1. What is the use of aggregate function?
2. How different number of rows can be counted?
3. What is the difference between having and where clause?
4. Dose WHERE clause work with aggregate functions?