String Patterns, Sorting and Grouping

We will go through some SQL practice problems that will provide hands-on experience with string patterns, sorting result sets and grouping result sets.

Software Used in this Lab

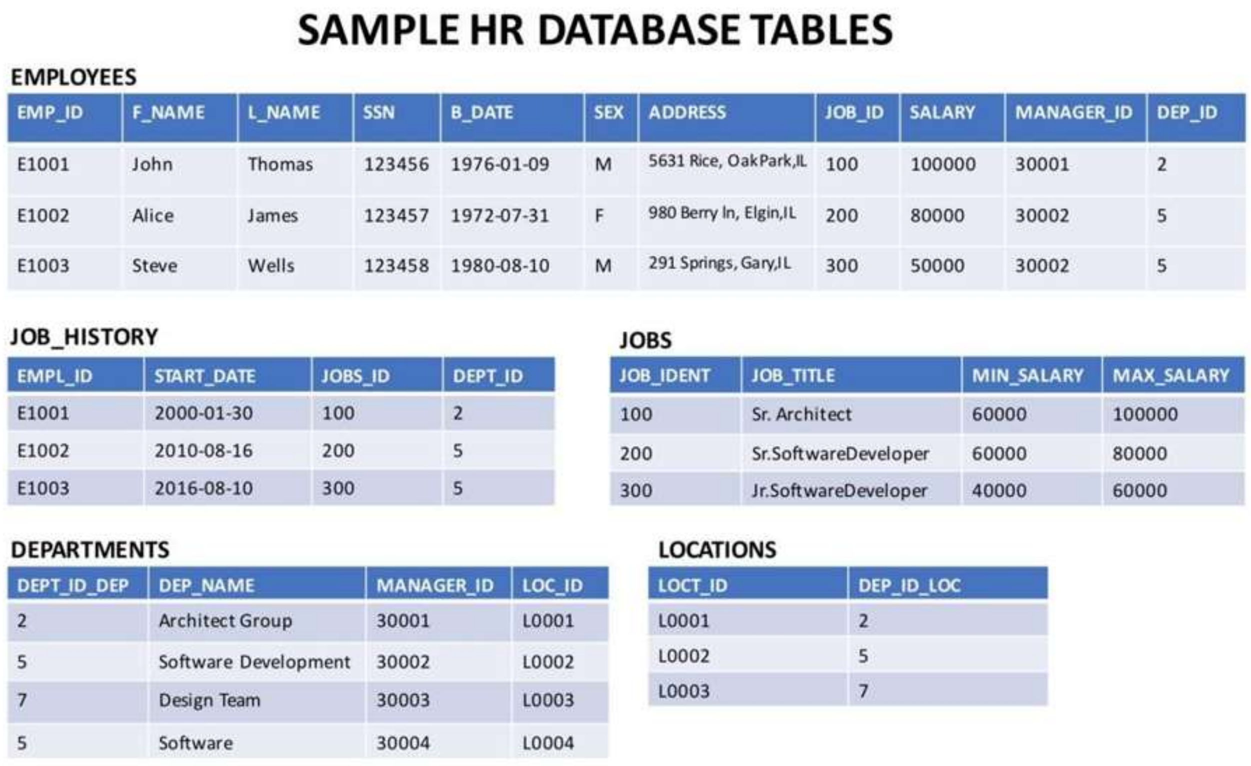
We will use an IBM Db2 Database. Db2 is a Relational Database Management System (RDBMS) from IBM, designed to store, analyze and retrieve data efficiently.

To complete this we will utilize a Db2 database service on IBM Cloud.

Database Used in this Lab

The database used in this lab is an internal database. We will be working on a sample HR database. This HR database schema consists of 5 tables called **EMPLOYEES**, **JOB_HISTORY**, **JOBS**, **DEPARTMENTS** and **LOCATIONS**. Each table has a few rows of sample data. The following diagram shows the tables for the HR database:

|



SAMPLE HR DATABASE TABLES

EMPLOYEES

EMP_ID	F_NAME	L_NAME	SSN	B_DATE	SEX	ADDRESS	JOB_ID	SALARY	MANAGER_ID	DEP_ID
E1001	John	Thomas	123456	1976-01-09	M	5631 Rice, OakPark,IL	100	100000	30001	2
E1002	Alice	James	123457	1972-07-31	F	980 Berry ln, Elgin,IL	200	80000	30002	5
E1003	Steve	Wells	123458	1980-08-10	M	291 Springs, Gary,IL	300	50000	30002	5

JOB_HISTORY

EMPL_ID	START_DATE	JOBS_ID	DEPT_ID
E1001	2000-01-30	100	2
E1002	2010-08-16	200	5
E1003	2016-08-10	300	5

JOBS

JOB_IDENT	JOB_TITLE	MIN_SALARY	MAX_SALARY
100	Sr. Architect	60000	100000
200	Sr.SoftwareDeveloper	60000	80000
300	Jr.SoftwareDeveloper	40000	60000

DEPARTMENTS

DEPT_ID_DEP	DEP_NAME	MANAGER_ID	LOC_ID
2	Architect Group	30001	L0001
5	Software Development	30002	L0002
7	Design Team	30003	L0003
5	Software	30004	L0004

LOCATIONS

LOCT_ID	DEP_ID_LOC
L0001	2
L0002	5
L0003	7

Objectives

- Simplifying a SELECT statement by using string patterns, ranges, or sets of values
- Sorting the result set in either ascending or descending order and identify which column to use for the sorting order
- Eliminating duplicates from a result set and further restrict a result set

Exercise 1: String Patterns

In this exercise, you will go through some SQL problems on String Patterns.

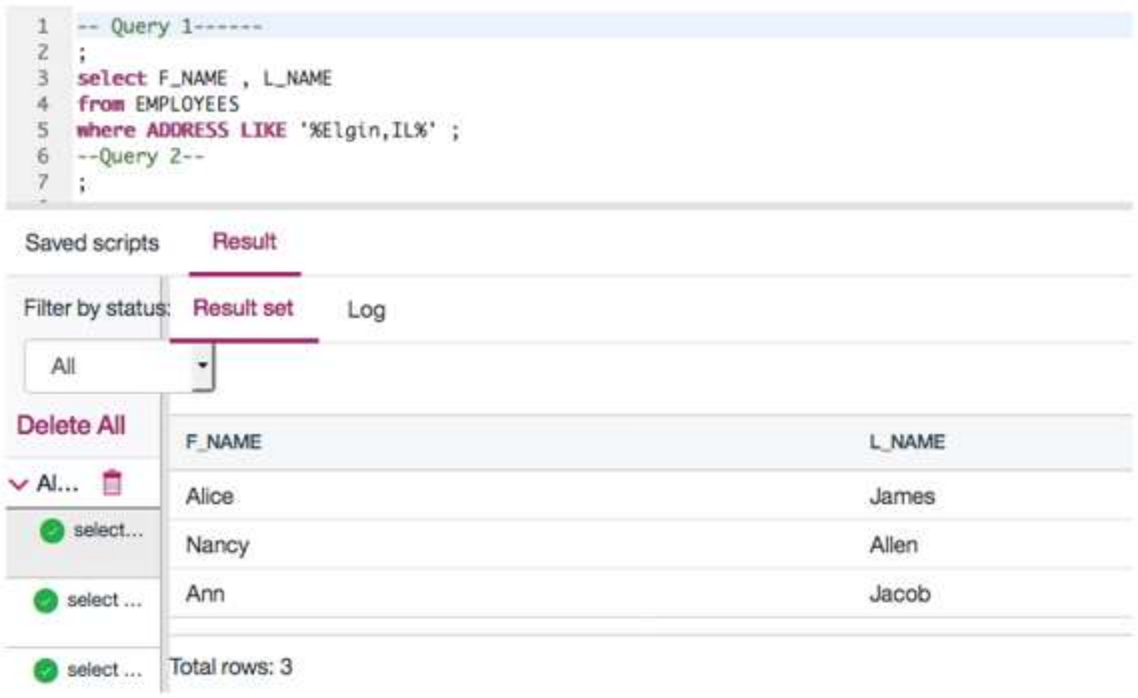
1. Problem:

   Retrieve all employees whose address is in Elgin,IL.

   Solution:

   ```
   SELECT F_NAME , L_NAME
   FROM EMPLOYEES
   WHERE ADDRESS LIKE '%Elgin,IL%';
   ```

   Output:



```
-- Query 1------
;
select F_NAME , L_NAME
from EMPLOYEES
where ADDRESS LIKE '%Elgin,IL%' ;
--Query 2--
;
```

F_NAME	L_NAME
Alice	James
Nancy	Allen
Ann	Jacob

Total rows: 3

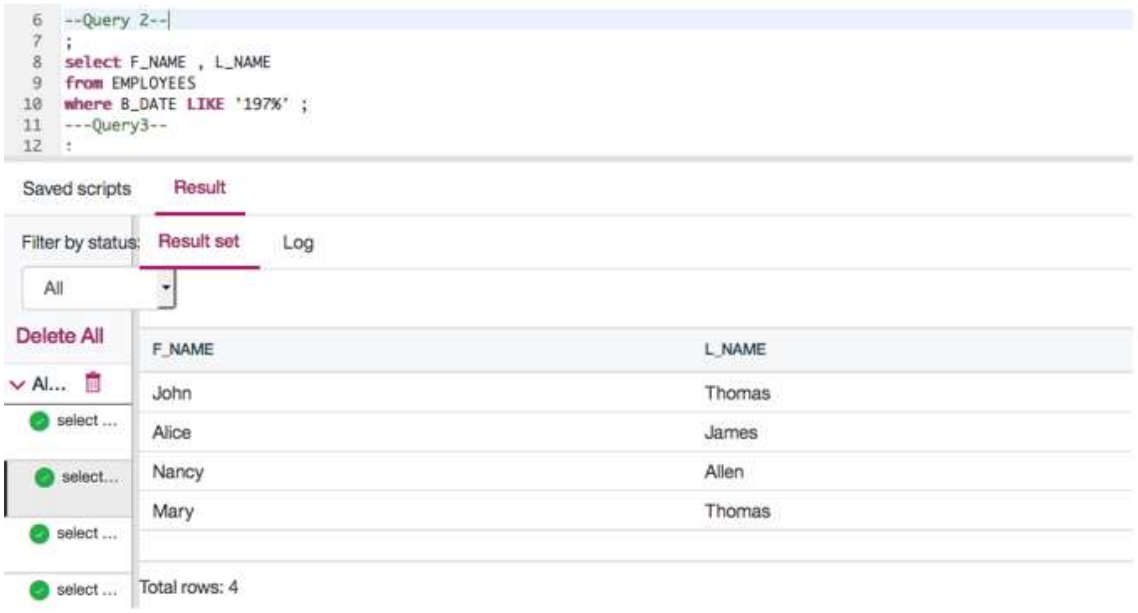
2. Problem:

   Retrieve all employees who were born during the 1970's.

   Solution:

```
SELECT F_NAME , L_NAME
FROM EMPLOYEES
WHERE B_DATE LIKE '197%';
```

   Output:



```
--Query 2--
;
select F_NAME , L_NAME
from EMPLOYEES
where B_DATE LIKE '197%' ;
---Query3--
;
```

F_NAME	L_NAME
John	Thomas
Alice	James
Nancy	Allen
Mary	Thomas

Total rows: 4

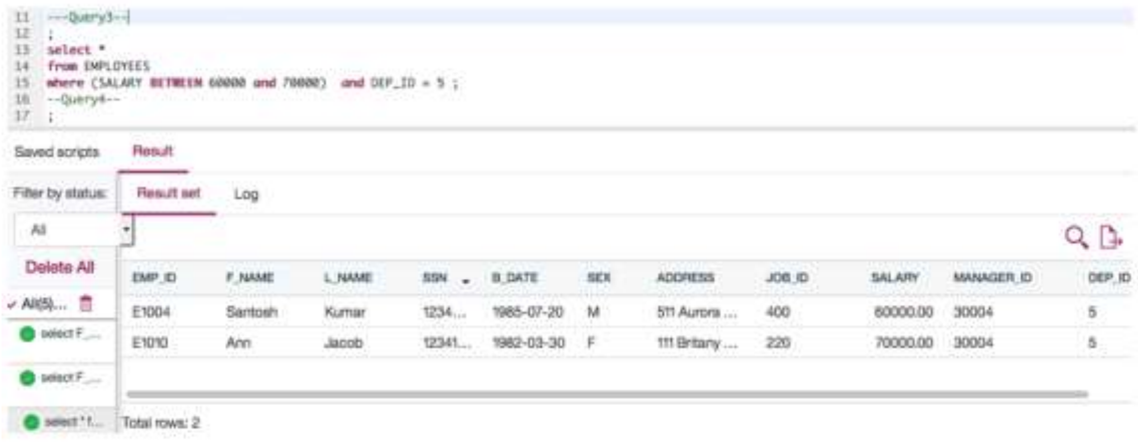
3. Problem:

   Retrieve all employees in department 5 whose salary is between 60000 and 70000.

   Solution:

```
SELECT *
FROM EMPLOYEES
WHERE (SALARY BETWEEN 60000 AND 70000) AND DEP_ID = 5;
```

   Output:



```
---Query3--
;
select *
from EMPLOYEES
where (SALARY BETWEEN 60000 and 70000)  and DEP_ID = 5 ;
--Query4--
;
```

EMP_ID	F_NAME	L_NAME	SSN	B_DATE	SEX	ADDRESS	JOB_ID	SALARY	MANAGER_ID	DEP_ID
E1004	Santosh	Kumar	1234...	1985-07-20	M	511 Aurora ...	400	60000.00	30004	5
E1010	Ann	Jacob	12341...	1982-03-30	F	111 Britany ...	220	70000.00	30004	5

Total rows: 2

Exercise 2: Sorting

In this exercise, you will go through some SQL problems on Sorting.

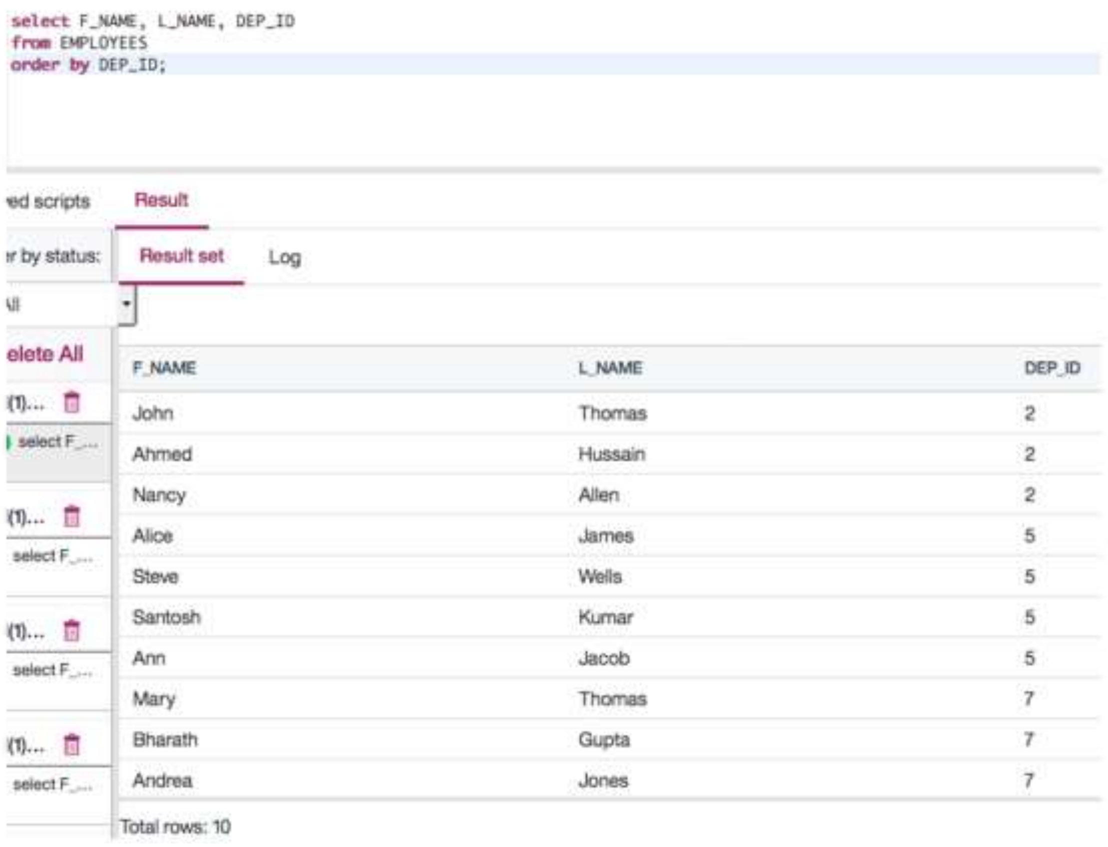
1. Problem:

   Retrieve a list of employees ordered by department ID.

   Solution:

   ```
   SELECT F_NAME, L_NAME, DEP_ID
   FROM EMPLOYEES
   ORDER BY DEP_ID;
   ```

   Output:



```
select F_NAME, L_NAME, DEP_ID
from EMPLOYEES
order by DEP_ID;
```

F_NAME	L_NAME	DEP_ID
John	Thomas	2
Ahmed	Hussain	2
Nancy	Allen	2
Alice	James	5
Steve	Wells	5
Santosh	Kumar	5
Ann	Jacob	5
Mary	Thomas	7
Bharath	Gupta	7
Andrea	Jones	7

Total rows: 10

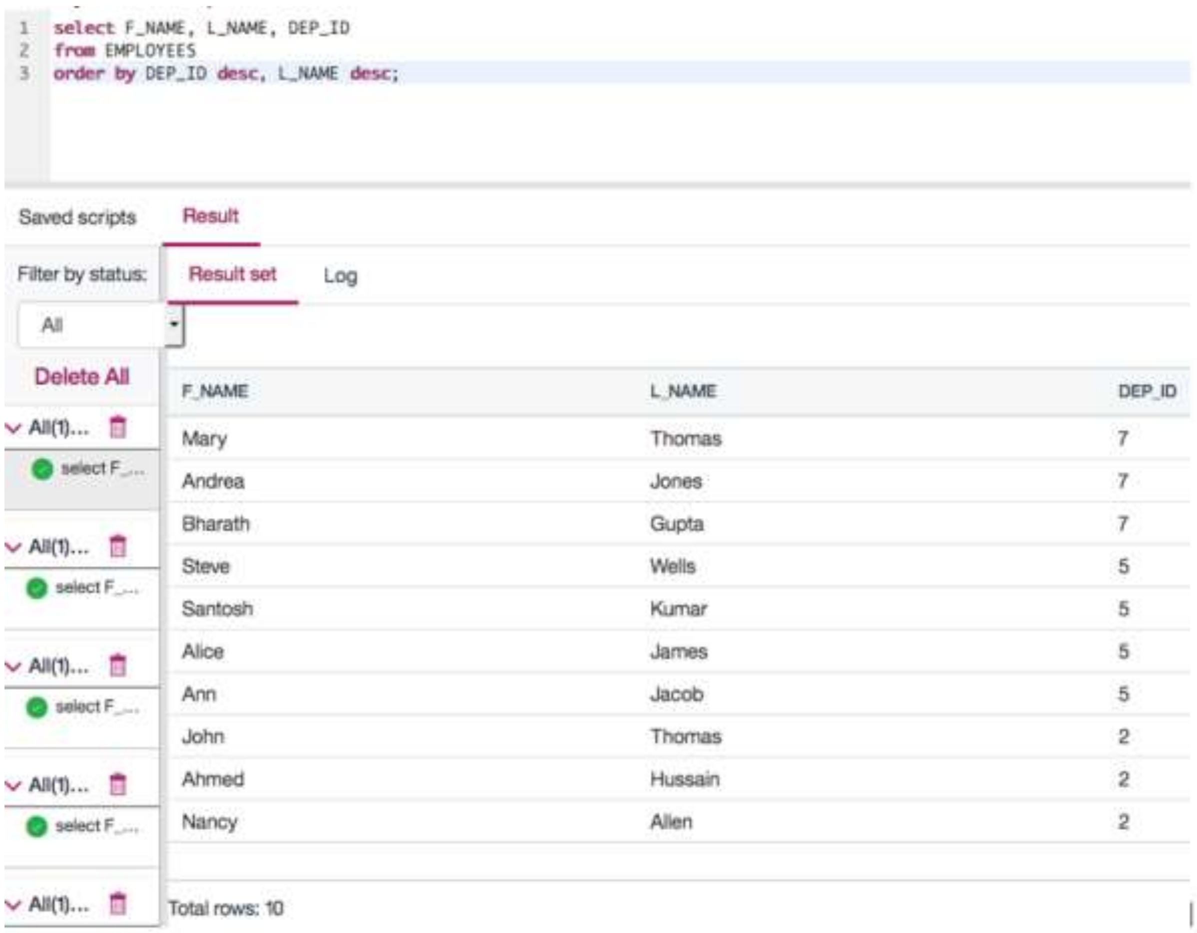
2. Problem:

   Retrieve a list of employees ordered in descending order by department ID and within each department ordered alphabetically in descending order by last name.

   Solution:

```
SELECT F_NAME, L_NAME, DEP_ID
FROM EMPLOYEES
ORDER BY DEP_ID DESC, L_NAME DESC;
```

   Output:



```
select F_NAME, L_NAME, DEP_ID
from EMPLOYEES
order by DEP_ID desc, L_NAME desc;
```

F_NAME	L_NAME	DEP_ID
Mary	Thomas	7
Andrea	Jones	7
Bharath	Gupta	7
Steve	Wells	5
Santosh	Kumar	5
Alice	James	5
Ann	Jacob	5
John	Thomas	2
Ahmed	Hussain	2
Nancy	Allen	2

Total rows: 10

3. (Optional) Problem:

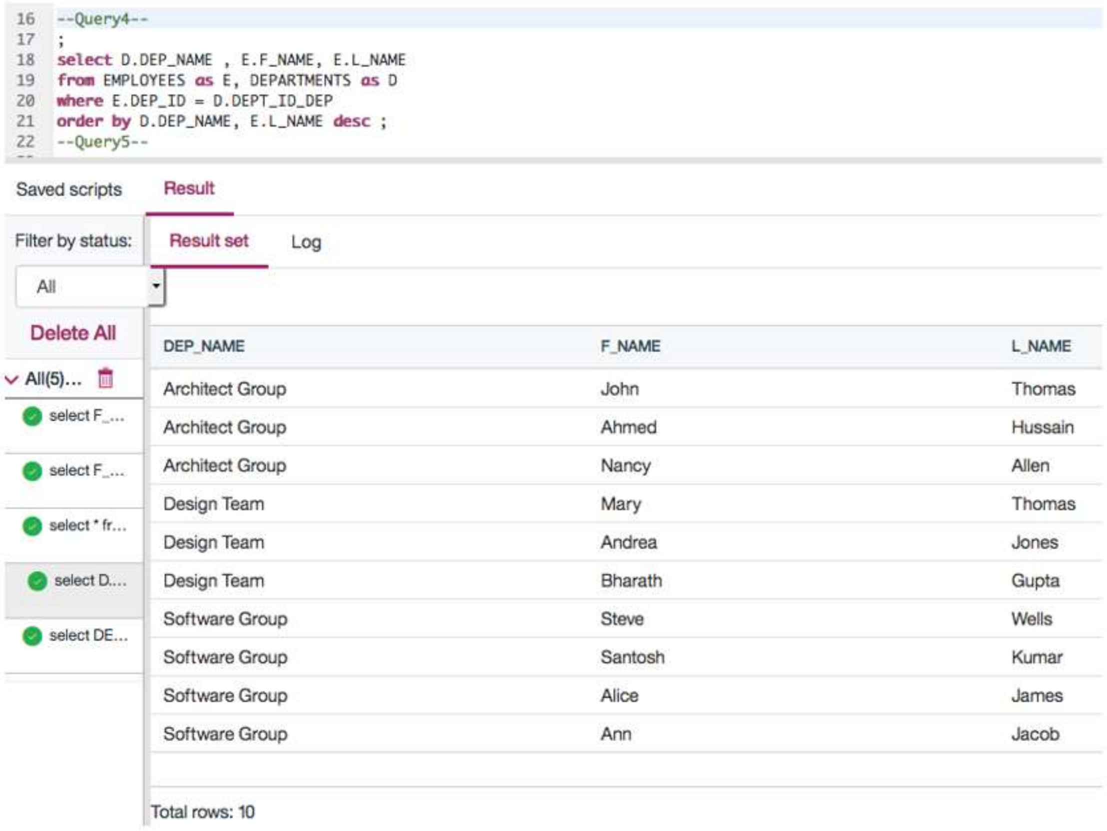
   In SQL problem 2 (Exercise 2 Problem 2), use department name instead of department ID. Retrieve a list of employees ordered by department name, and within each department ordered alphabetically in descending order by last name.

   Solution:

```
SELECT D.DEP_NAME , E.F_NAME, E.L_NAME
FROM EMPLOYEES as E, DEPARTMENTS as D
WHERE E.DEP_ID = D.DEPT_ID_DEP
ORDER BY D.DEP_NAME, E.L_NAME DESC;
```

   In the SQL Query above, `D` and `E` are aliases for the table names. Once we define an alias like `D` in our query, we can simply write `D.COLUMN_NAME` rather than the full form `DEPARTMENTS.COLUMN_NAME`.

   Output:



```
--Query4--
;
select D.DEP_NAME , E.F_NAME, E.L_NAME
from EMPLOYEES as E, DEPARTMENTS as D
where E.DEP_ID = D.DEPT_ID_DEP
order by D.DEP_NAME, E.L_NAME desc ;
--Query5--
```

DEP_NAME	F_NAME	L_NAME
Architect Group	John	Thomas
Architect Group	Ahmed	Hussain
Architect Group	Nancy	Allen
Design Team	Mary	Thomas
Design Team	Andrea	Jones
Design Team	Bharath	Gupta
Software Group	Steve	Wells
Software Group	Santosh	Kumar
Software Group	Alice	James
Software Group	Ann	Jacob

Total rows: 10

Exercise 3: Grouping

We will go through some SQL problems on Grouping.

NOTE: The SQL problems in this exercise involve usage of SQL Aggregate functions AVG and COUNT. COUNT has been covered earlier. AVG is a function that can be used to calculate the Average or Mean of all values of a specified column in the result set. For example, to retrieve the average salary for all employees in the EMPLOYEES table, issue the query: `SELECT AVG(SALARY) FROM EMPLOYEES;`.

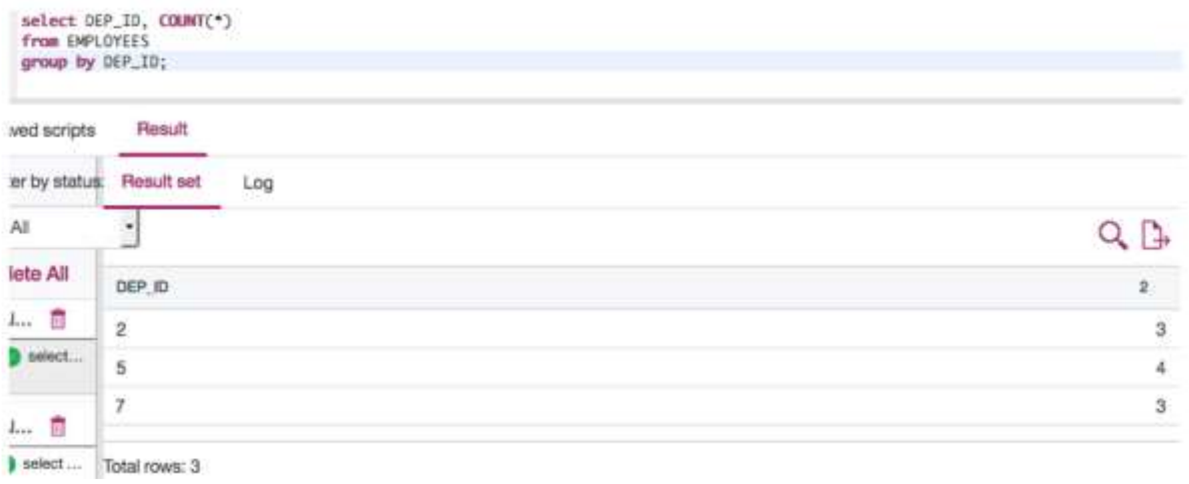
1. Problem:

   For each department ID retrieve the number of employees in the department.

   Solution:

   ```
   SELECT DEP_ID, COUNT(*)
   FROM EMPLOYEES
   GROUP BY DEP_ID;
   ```

   Output:



```
select DEP_ID, COUNT(*)
from EMPLOYEES
group by DEP_ID;
```

DEP_ID	2
2	3
5	4
7	3

Total rows: 3

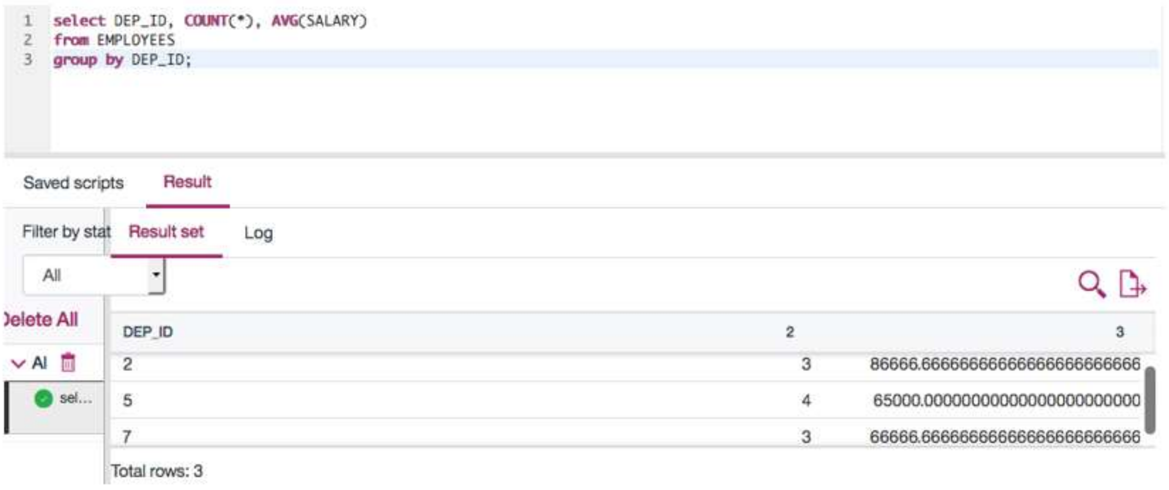
2. Problem:

   For each department retrieve the number of employees in the department, and the average employee salary in the department.

   Solution:

```
SELECT DEP_ID, COUNT(*), AVG(SALARY)
FROM EMPLOYEES
GROUP BY DEP_ID;
```

   Output:



```
select DEP_ID, COUNT(*), AVG(SALARY)
from EMPLOYEES
group by DEP_ID;
```

DEP_ID	2	3
2	3	86666.666666666666666666666
5	4	65000.000000000000000000000
7	3	66666.666666666666666666666

Total rows: 3

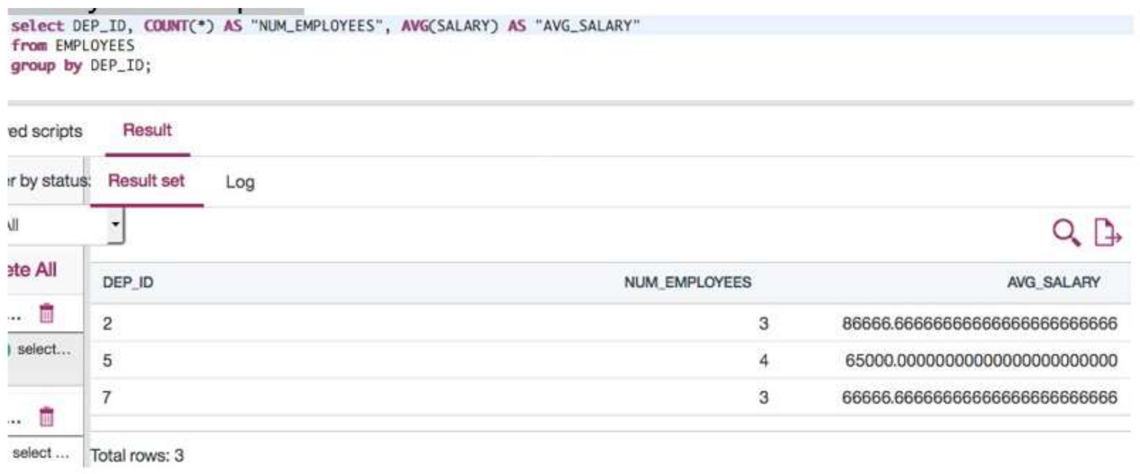
3. Problem:

   Label the computed columns in the result set of SQL problem 2 (Exercise 3 Problem 2) as NUM_EMPLOYEES and AVG_SALARY.

   Solution:

```
SELECT DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
FROM EMPLOYEES
GROUP BY DEP_ID;
```

   Output:



```
select DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
from EMPLOYEES
group by DEP_ID;
```

DEP_ID	NUM_EMPLOYEES	AVG_SALARY
2	3	86666.666666666666666666666
5	4	65000.000000000000000000000
7	3	66666.666666666666666666666

Total rows: 3

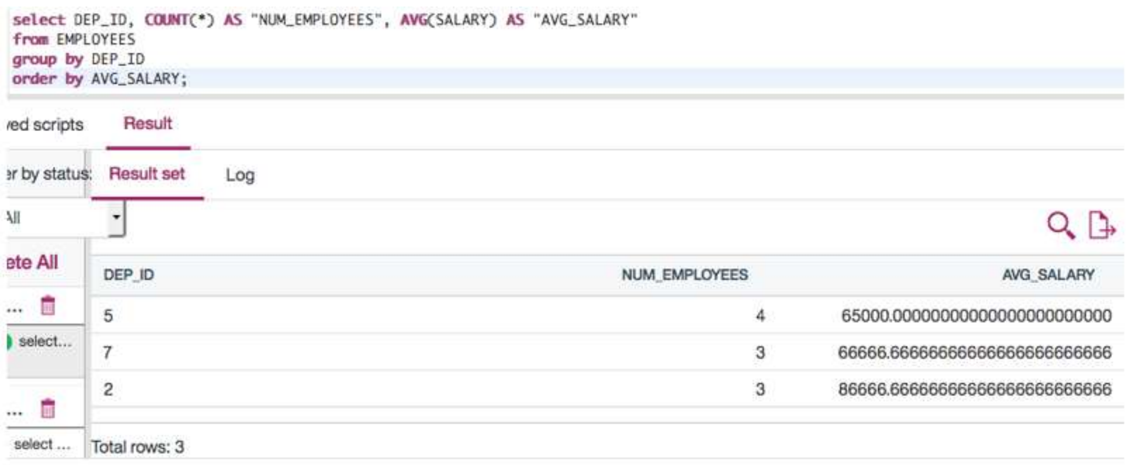
4. Problem:

In SQL problem 3 (Exercise 3 Problem 3), order the result set by Average Salary..

Solution:

```
SELECT DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
FROM EMPLOYEES
GROUP BY DEP_ID
ORDER BY AVG_SALARY;
```

Output:



```
select DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
from EMPLOYEES
group by DEP_ID
order by AVG_SALARY;
```

DEP_ID	NUM_EMPLOYEES	AVG_SALARY
5	4	65000.000000000000000000000
7	3	66666.666666666666666666666
2	3	86666.666666666666666666666

Total rows: 3

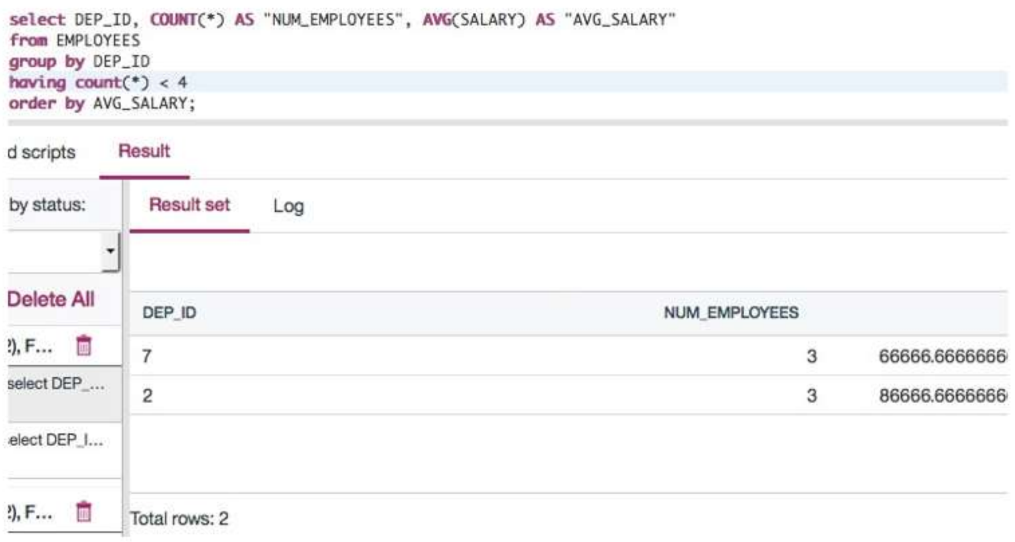
5. Problem:

   In SQL problem 4 (Exercise 3 Problem 4), limit the result to departments with fewer than 4 employees.

   Solution:

```
SELECT DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
FROM EMPLOYEES
GROUP BY DEP_ID
HAVING count(*) < 4
ORDER BY AVG_SALARY;
```

   Output:



```
select DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
from EMPLOYEES
group by DEP_ID
having count(*) < 4
order by AVG_SALARY;
```

DEP_ID	NUM_EMPLOYEES	
7	3	66666.6666666
2	3	86666.6666666

Total rows: 2