



Hands-on Lab : String Patterns, Sorting and Grouping

Estimated time needed: 35 minutes

In this lab, you 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

In this lab, you 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 lab you will utilize a Db2 database service on IBM Cloud. If you did not already complete this lab task earlier in this module, you will not yet have access to Db2 on IBM Cloud, and you will need to follow the lab below first:

- [Hands-on Lab : Sign up for IBM Cloud, Create Db2 service instance and Get started with the Db2 console](#)

Database Used in this Lab

The database used in this lab is an internal database. You 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_ID | JOB_TITLE | MIN_SALARY | MAX_SALARY |
|--------|----------------------|------------|------------|
| 100 | Sr. Architect | 60000 | 100000 |
| 200 | Sr.SoftwareDeveloper | 60000 | 80000 |
| 300 | Jr.SoftwareDeveloper | 40000 | 60000 |

DEPARTMENTS

| DEPT_ID | DEPT_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

| LOC_ID | DEPT_ID |
|--------|---------|
| L0001 | 2 |
| L0002 | 5 |
| L0003 | 7 |

NOTE: This lab requires you to have all 5 of these tables of the HR database populated with sample data on Db2. If you didn't complete the earlier lab in this module, you won't have the tables above populated with sample data on Db2, so you will need to go through the lab below first:

- [Hands-on Lab : Create tables using SQL scripts and Load data into tables](#)

Objectives

After completing this lab, you will be able to:

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

NOTE : Make sure that you are using the CSV file and datasets from the same instruction file.

Instructions

When you approach the exercises in this lab, follow the instructions to run the queries on Db2:

- Go to the [Resource List](#) of IBM Cloud by logging in where you can find the Db2 service instance that you created in a previous lab under **Services** section. Click on the **Db2-xx service**. Next, open the Db2 Console by clicking on **Open Console** button. Click on the 3-bar menu icon in the top left corner and go to the **Run SQL** page. The Run SQL tool enables you to run SQL statements.
 - If needed, follow [Hands-on Lab : Sign up for IBM Cloud, Create Db2 service instance and Get started with the Db2 console](#)

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.

▼ Hint

Use the LIKE operator to find similar strings.

▼ Solution

1. 1
2. 2
3. 3
1. SELECT F_NAME , L_NAME
2. FROM EMPLOYEES
3. WHERE ADDRESS LIKE '%Elgin,IL%';

Copied!

▼ Output

1 -- Query 1-----
2 ;
3 select F_NAME , L_NAME
4 from EMPLOYEES
5 where ADDRESS LIKE '%Elgin,IL%';
6 --Query 2--
7 ;
8

Saved scripts

Result

Filter by status: Result set Log

All

Delete All

▼ All...

select...

select ...

select ...

| 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.

▼ Hint

Use the LIKE operator to find similar strings.

▼ Solution

1. 1
2. 2
3. 3
1. SELECT F_NAME , L_NAME
2. FROM EMPLOYEES
3. WHERE B_DATE LIKE '197%';

Copied!

▼ Output

6 --Query 2--
7 ;
8 select F_NAME , L_NAME
9 from EMPLOYEES
10 where B_DATE LIKE '197%';
11 ---Query3--
12 ;

Saved scripts

Result

Filter by status: Result set Log

All

Delete All

▼ All...

select...

select ...

select ...

select ...

| 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.

▼ Hint

Use the keyword BETWEEN for this SQL problem.

▼ Solution

1. 1
2. 2
3. 3
1. SELECT *
2. FROM EMPLOYEES
3. WHERE (SALARY BETWEEN 60000 AND 70000) AND DEP_ID = 5;

Copied!

▼ Output

11 ---Query3--|

12 ;

13 select *

14 from EMPLOYEES

15 where (SALARY BETWEEN 60000 and 70000) and DEP_ID = 5 ;

16 --Query4--

17 ;

Saved scripts

Result

Filter by status:

Result set

Log

All

▼

🔍 📄

Delete All

✓ All(5)...

🗑

| 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 |

select F_...

select F_...

select * f...

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.

▼ Hint

Use the ORDER BY clause for this SQL problem. By default, the ORDER BY clause sorts the records in ascending order.

▼ Solution

1. 1
2. 2
3. 3
1. SELECT F_NAME, L_NAME, DEP_ID
2. FROM EMPLOYEES
3. ORDER BY DEP_ID;

Copied!

▼ Output

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

ed scripts

Result

er by status:

Result set

Log

All

▼

delete All

(1)...

select F_...

(1)...

select F_...

(1)...

select F_...

(1)...

select F_...

| 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.

▼ Hint

Use the ORDER BY clause with DESC for this SQL problem.

▼ Solution

1. 1

2. 2

3. 3
1. SELECT F_NAME, L_NAME, DEP_ID

2. FROM EMPLOYEES

3. ORDER BY DEP_ID DESC, L_NAME DESC;

Copied!

▼ Output

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

| Saved scripts | Result | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------------------|---|--------|--------|--------|------|--------|---|--------|-------|---|---------|-------|---|-------|-------|---|---------|-------|---|-------|-------|---|-----|-------|---|------|--------|---|-------|---------|---|-------|-------|---|----------------|--|--|
| Filter by status: | Result set Log | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| All | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Delete All | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ▼ All(1)... 🗑️ 🟢 select F_... | <table border="1"> <thead> <tr> <th>F_NAME</th><th>L_NAME</th><th>DEP_ID</th></tr> </thead> <tbody> <tr><td>Mary</td><td>Thomas</td><td>7</td></tr> <tr><td>Andrea</td><td>Jones</td><td>7</td></tr> <tr><td>Bharath</td><td>Gupta</td><td>7</td></tr> <tr><td>Steve</td><td>Wells</td><td>5</td></tr> <tr><td>Santosh</td><td>Kumar</td><td>5</td></tr> <tr><td>Alice</td><td>James</td><td>5</td></tr> <tr><td>Ann</td><td>Jacob</td><td>5</td></tr> <tr><td>John</td><td>Thomas</td><td>2</td></tr> <tr><td>Ahmed</td><td>Hussain</td><td>2</td></tr> <tr><td>Nancy</td><td>Allen</td><td>2</td></tr> <tr><td colspan="3">Total rows: 10</td></tr> </tbody> </table> | 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 | | |
| 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ▼ All(1)... 🗑️ 🟢 select F_... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ▼ All(1)... 🗑️ 🟢 select F_... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ▼ All(1)... 🗑️ 🟢 select F_... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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.

▼ Hint

Department name is in the DEPARTMENTS table. So your query will need to retrieve data from more than one table. Don't worry if you are not able to figure this SQL problem out. We cover working with multiple tables in the lecture **Working with Multiple Tables**.

▼ Solution

```

1. 1
2. 2
3. 3
4. 4
1. SELECT D.DEP_NAME , E.F_NAME, E.L_NAME
2. FROM EMPLOYEES as E, DEPARTMENTS as D
3. WHERE E.DEP_ID = D.DEPT_ID_DEP
4. ORDER BY D.DEP NAME, E.L NAME DESC;
```

Copied!

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

▼ Output

16 --Query4--
17 ;
18 select D.DEP_NAME , E.F_NAME, E.L_NAME
19 from EMPLOYEES as E, DEPARTMENTS as D
20 where E.DEP_ID = D.DEPT_ID_DEP
21 order by D.DEP_NAME, E.L_NAME desc ;
22 --Query5--
--

Saved scripts

Result

Filter by status: Result set Log

All

Delete All

✓ All(5)...
select F_...
select F_...
select * fr...
select D...
select DE...

| 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

In this exercise, you 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;. You will learn more about AVG and other aggregate functions later in the lecture **Built-in Database Functions**.

1. Problem:

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

▼ Hint

Use COUNT(*) to retrieve the total count of a column, and then GROUP BY.

▼ Solution

1. 1

2. 2

3. 3
1. SELECT DEP_ID, COUNT(*)

2. FROM EMPLOYEES

3. GROUP BY DEP_ID;

Copied!

▼ Output

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

Saved scripts

Result

Filter by status: Result set Log

All

Delete All

J...
select...
J...
select ...

| DEP_ID | |
|--------|---|
| 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..

▼ Hint

Use COUNT(*) to retrieve the total count of a column, and AVG() function to compute average salaries, and then GROUP BY.

▼ Solution

- ```
1. 1
2. 2
3. 3
1. SELECT DEP_ID, COUNT(*), AVG(SALARY)
2. FROM EMPLOYEES
3. GROUP BY DEP_ID;
```

Copied!

▼ Output

```

1 select DEP_ID, COUNT(*), AVG(SALARY)
2 from EMPLOYEES
3 group by DEP_ID;

```

Saved scripts    Result

Filter by stat    Result set    Log

All

|   | DEP_ID | 2                                | 3 |
|---|--------|----------------------------------|---|
| 2 | 3      | 86666.66666666666666666666666666 |   |
| 5 | 4      | 65000.00000000000000000000000000 |   |
| 7 | 3      | 66666.66666666666666666666666666 |   |

Total rows: 3

Delete All

✓ All ✕ sel...

### 3. Problem:

Label the computed columns in the result set of SQL problem 2 (Exercise 3 Problem 2) as NUM\_EMPLOYEES and AVG\_SALARY.

▼ Hint

Use SQL Aliases: column\_name AS alias\_name. For example, AVG(SALARY) AS "AVG\_SALARY".

▼ Solution

- ```
1. 1
2. 2
3. 3
1. SELECT DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
2. FROM EMPLOYEES
3. GROUP BY DEP_ID;
```

Copied!

▼ Output

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

4. Problem:

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

▼ Hint

Use ORDER BY after the GROUP BY.

▼ Solution

- 1. 1
- 2. 2
- 3. 3
- 4. 4
- 1. SELECT DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
- 2. FROM EMPLOYEES
- 3. GROUP BY DEP_ID
- 4. ORDER BY AVG_SALARY;

Copied!

▼ Output

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

ed scripts

Result

er by status: Result set Log

All

ete All

...

select...

...

select ...

| DEP_ID | NUM_EMPLOYEES | AVG_SALARY |
|--------|---------------|--------------------------------|
| 5 | 4 | 65000.000000000000000000000000 |
| 7 | 3 | 66666.666666666666666666666666 |
| 2 | 3 | 86666.666666666666666666666666 |

Total rows: 3

5. Problem:

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

▼ Hint

Use HAVING after the GROUP BY, and use the count() function in the HAVING clause instead of the column label.

▼ Solution

- 1. 1
- 2. 2
- 3. 3
- 4. 4
- 5. 5
- 1. SELECT DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY"
- 2. FROM EMPLOYEES
- 3. GROUP BY DEP_ID
- 4. HAVING count(*) < 4
- 5. ORDER BY AVG_SALARY;

Copied!

▼ 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;
```

d scripts

Result

by status: Result set Log

Delete All

!), F...

select DEP_...

elect DEP_I...

!), F...

| DEP_ID | NUM_EMPLOYEES | AVG_SALARY |
|--------|---------------|--------------------------------|
| 7 | 3 | 66666.666666666666666666666666 |
| 2 | 3 | 86666.666666666666666666666666 |

Total rows: 2

Solution Script

If you would like to run all the solution queries of the SQL problems of this lab with a script, download the script below. Upload the script to the Db2 console and run. Follow [Hands-on Lab : Create tables using SQL scripts and Load data into tables](#) on how to upload a script to Db2 console and run it.

- [StringPattern-Sorting-Grouping_Solution_Script.sql](#)

Congratulations! You have completed this lab, and you are ready for the next topic.

Author(s)

- [Rav Ahuja](#)
- [Sandip Saha Joy](#)

Changelog

| Date | Version | Changed by | Change Description |
|------------|---------|----------------------------|---------------------------------------|
| 2023-05-10 | 2.2 | Eric Hao & Vladislav Boyko | Updated Page Frames |
| 2020-12-24 | 2.1 | Steve Ryan | ID Reviewed |
| 2020-12-08 | 2.0 | Sandip Saha Joy | Created revised version from DB0201EN |
| 2020 | 1.0 | Rav Ahuja | Created initial version |

© IBM Corporation 2023. All rights reserved.