# Laboratorio Práctico: Patrones de Cadenas, Ordenamiento y Agrupación en **MySQL**

#### Tiempo estimado: 30 minutos

En este laboratorio, aprenderás a crear tablas y cargar datos en el servicio de base de datos MySQL utilizando la herramienta de interfaz gráfica de usuario (GUI) phpMyAdmin.

#### **Objetivos**

Después de completar este laboratorio, podrás:

- Filtrar la salida de una consulta SELECT utilizando patrones de cadenas, rangos o conjuntos de valores.
- Ordenar el conjunto de resultados en orden ascendente o descendente de acuerdo con una columna predefinida.
- Agrupar los resultados de una consulta en función de un parámetro seleccionado para refinar aún más la respuesta.

#### Software Utilizado en este Laboratorio

En este laboratorio, utilizarás MySQL. MySQL es un Sistema de Gestión de Bases de Datos Relacional (RDBMS) diseñado para almacenar, manipular y recuperar datos de manera eficiente.



Para completar este laboratorio, utilizarás el servicio de base de datos relacional MySQL disponible como parte de IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs es un entorno de laboratorio virtual utilizado en este curso.

#### Base de Datos Utilizada en este Laboratorio

La base de datos utilizada en este laboratorio es una base de datos interna. Estarás trabajando en una base de datos de recursos humanos (HR) de muestra. Este esquema de base de datos de HR consta de 5 tablas llamadas EMPLOYEES, JOB HISTORY, JOBS, DEPARTMENTS y LOCATIONS. Cada tabla tiene algunas filas de datos de ejemplo. El siguiente diagrama muestra las tablas de la base de datos de HR:

#### SAMPLE HR DATABASE TABLES **EMPLOYEES** 123456 1976-01-09 5631 Rice, OakPark,IL 30001 E1001 100 100000 E1002 Alice 123457 1972-07-31 980 Berry In, Elgin,IL 200 E1003 123458 1980-08-10 30002 JOB\_HISTORY JOBS EMPL\_ID JOB\_IDENT JOB\_TITLE E1001 2000-01-30 100 2 100 Sr. Architect 60000 100000 2010-08-16 200 Sr.SoftwareDeveloper 60000 80000 E1003 Jr.SoftwareDeveloper **DEPARTMENTS** LOCATIONS LOCT\_IE L0001 Architect Group L0001 L0002 L0003 Design Team 30003

## Cargar la base de datos

Usando las habilidades adquiridas en los módulos anteriores, primero debes crear la base de datos en MySQL. Sigue los pasos a continuación:

- 1. Abre la interfaz de phpMyAdmin desde el Skills Network Toolbox en Cloud IDE.
- 2. Crea una base de datos en blanco llamada 'HR'. Utiliza el script compartido en el enlace a continuación para crear las tablas requeridas. Script Create Tables.sql
- 3. Descarga los archivos en los enlaces a continuación a tu máquina local (si no lo has hecho ya en laboratorios anteriores). Departments.csv

Jobs.csv

JobsHistory.csv Locations.csv

4. Utiliza cada uno de estos archivos en la interfaz como datos para las respectivas tablas en la base de datos 'HR'.

## Patrones de Cadenas

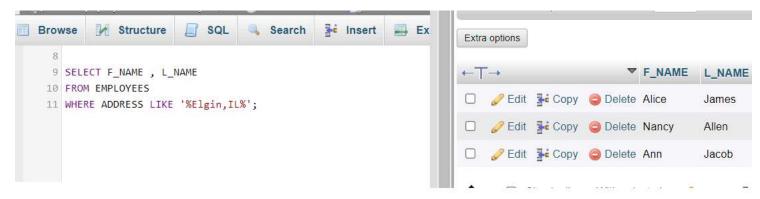
about:blank 1/9

Puedes usar patrones de cadenas para filtrar la respuesta de una consulta. Veamos el siguiente ejemplo:

Supongamos que necesitas recuperar los nombres F\_NAME y apellidos L\_NAME de todos los empleados que viven en Elgin, IL. Puedes usar el operador LIKE para recuperar cadenas que contengan el texto mencionado. El código se verá como se muestra a continuación.

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

Al ejecutar, la salida de la consulta debería aparecer como se muestra a continuación:



Ahora suponga que desea identificar a los empleados que nacieron durante los años 70. La consulta anterior se puede modificar a:

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

La salida de esta consulta será:

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

Delete John Thomas

Edit ** Copy ** Delete Alice James

Edit ** Copy ** Delete Nancy Allen

Edit ** Copy ** Delete Mary Thomas
```

Tenga en cuenta que en el primer ejemplo, el signo % se utiliza tanto antes como después del texto requerido. Esto indica que la cadena de dirección puede tener más caracteres, tanto antes como después del texto requerido.

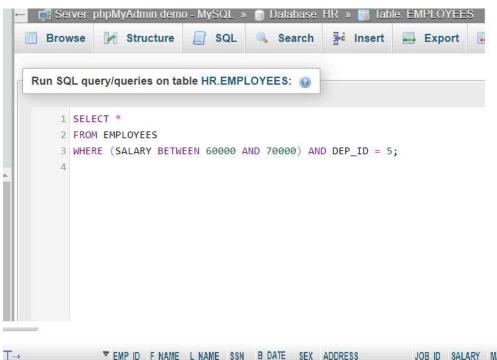
En el segundo ejemplo, dado que la fecha de nacimiento en los registros de Eployees comienza con el año de nacimiento, el signo % se aplica después de 197%, lo que indica que el año de nacimiento puede ser cualquier cosa entre 1970 y 1979. Además, el signo % también permite cualquier fecha posible a lo largo de los años seleccionados.

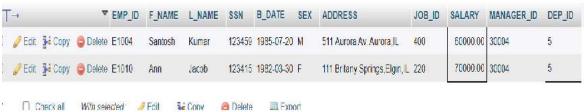
Considere un ejemplo más específico. Recuperemos todos los registros de empleados en el departamento 5 donde el salario esté entre 60000 y 70000. La consulta que se utilizará es

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

about:blank 2/9

La salida de la consulta se puede ver en la imagen a continuación.





## **Ordenamiento**

Puedes ordenar las entradas recuperadas en función de uno o más parámetros.

Primero, supón que debes recuperar una lista de empleados ordenada por ID de departamento.

El ordenamiento se realiza utilizando la cláusula ORDER BY en tu consulta SQL. Por defecto, la cláusula ORDER BY ordena los registros en orden ascendente.

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

La salida para esta consulta será como se muestra a continuación.

about:blank 3/9

```
about:blank
```

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

	Edit	<b>∄</b> Copy	Delete	John	Thomas	2
	Edit	₫ Сору	Delete	Ahmed	Hussain	2
	Edit	<b>≩</b> сору	Delete	Nancy	Allen	2
٨	Edit	<b>3</b> сору	Delete	Alice	James	5
	Edit	<b>∄</b> сору	Delete	Steve	Wells	5
	Edit	₫ Сору	Delete	Santosh	Kumar	5
	Edit	<b>3</b> € Сору	Delete	Ann	Jacob	5
	Edit	<b>∄</b> copy	Delete	Mary	Thomas	7
	Edit	<b>3</b> € Copy	Delete	Bharath	Gupta	7
	Edit	<b>∄</b> сору	Delete	Andrea	Jones	7

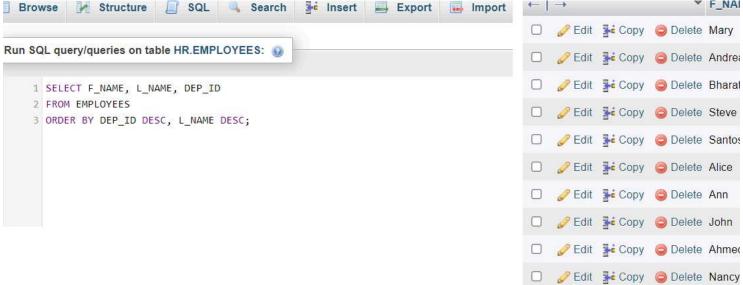
Ahora, obtenga la salida de la misma consulta en orden descendente por ID de departamento, y dentro de cada departamento, los registros deben estar ordenados en orden alfabético descendente por apellido. Para el orden descendente, puede utilizar la cláusula DESC.

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

La salida será como se muestra en la imagen a continuación.

about:blank

7/5/25, 12:39 about:blank F\_NAI Browse Structure SQL Search insert Export Import 🧷 Edit 👫 Copy 🥥 Delete Mary



# Agrupación

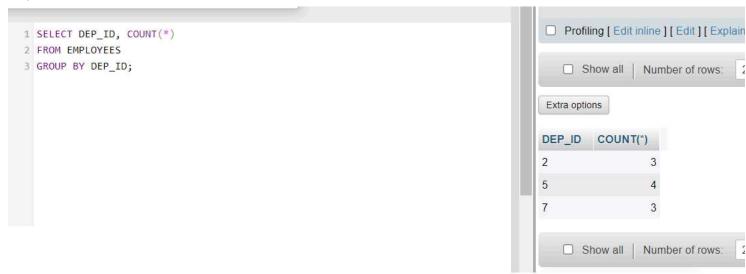
En este ejercicio, resolverás algunos problemas de SQL sobre Agrupación.

NOTA: Los problemas de SQL en este ejercicio implican el uso de las funciones de agregación SQL AVG y COUNT. COUNT se ha cubierto anteriormente. AVG es una función que se puede utilizar para calcular el promedio o media de todos los valores de una columna específica en el conjunto de resultados. Por ejemplo, para recuperar el salario promedio de todos los empleados en la tabla EMPLOYEES, emite la consulta: SELECT AVG(SALARY) FROM EMPLOYEES;.

Un buen ejemplo de agrupación sería si, para cada ID de departamento, deseamos recuperar el número de empleados en el departamento.

SELECT DEP\_ID, COUNT(\*) FROM EMPLOYEES
GROUP BY DEP\_ID;

about:blank 5/9



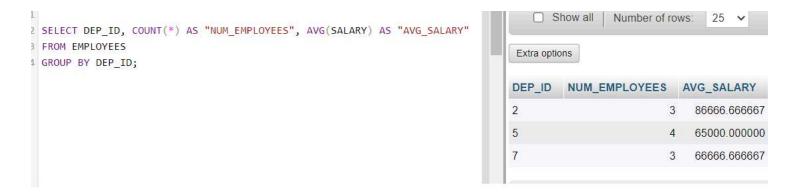
Ahora, para cada departamento, recupera el número de empleados en el departamento y el salario promedio de los empleados en el departamento. Para esto, puedes usar COUNT(\*) para obtener el conteo total de una columna y la función AVG() para calcular los salarios promedio, y luego usar GROUP BY.

SELECT DEP\_ID, COUNT(\*), AVG(SALARY)
FROM EMPLOYEES
GROUP BY DEP\_ID;



Puedes refinar tu salida utilizando etiquetas apropiadas para las columnas de datos recuperados. Etiqueta las columnas calculadas en el conjunto de resultados del último problema como NUM\_EMPLOYEES y AVG\_SALARY.

SELECT DEP\_ID, COUNT(\*) AS "NUM\_EMPLOYEES", AVG(SALARY) AS "AVG\_SALARY" FROM EMPLOYEES GROUP BY DEP\_ID;

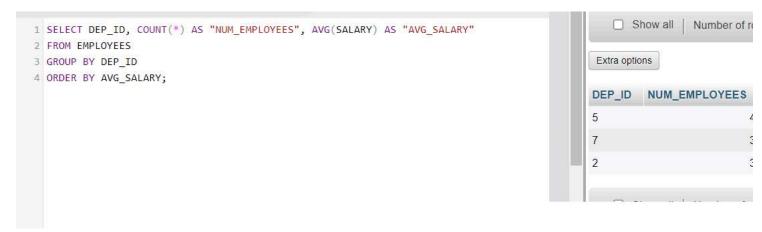


about:blank 6/9

También puedes combinar el uso de las cláusulas GROUP BY y ORDER BY para ordenar la salida de cada grupo de acuerdo con un parámetro específico. Es importante señalar que en tal caso, la cláusula ORDER BY debe usarse después de la cláusula GROUP BY. Por ejemplo, podemos ordenar el resultado de la consulta anterior por salario promedio. La consulta SQL se convertiría así en

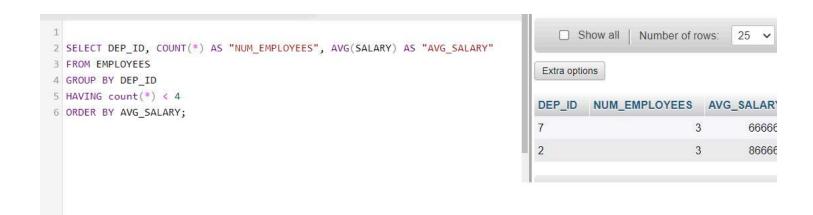
```
SELECT DEP_ID, COUNT(*) AS "NUM_EMPLOYEES", AVG(SALARY) AS "AVG_SALARY" FROM EMPLOYEES GROUP BY DEP_ID ORDER BY AVG_SALARY;
```

La salida de la consulta debería verse así:



En caso de que necesites filtrar una respuesta agrupada, debes usar la cláusula HAVING. En el ejemplo anterior, si deseamos limitar el resultado a departamentos con menos de 4 empleados, tendremos que usar HAVING después del GROUP BY y utilizar la función count() en la cláusula HAVING en lugar de la etiqueta de columna.

```
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;
```



## Preguntas de Práctica

- 1. Recupera la lista de todos los empleados, nombres y apellidos, cuyos nombres comienzan con 'S'.
- ▼ Haz clic aquí para la Solución

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

- 2. Organiza todos los registros de la tabla EMPLOYEES en orden ascendente de la fecha de nacimiento.
- ▼ Haz clic aquí para la solución

```
SELECT *
FROM EMPLOYEES
ORDER BY B_DATE;
```

- 3. Agrupa los registros en función de los IDs de departamento y filtra aquellos que tengan un salario promedio mayor o igual a 60000. Muestra el ID de departamento y el salario promedio.
- ▼ Haz clic aquí para la solución

```
SELECT DEP_ID, AVG(SALARY)
FROM EMPLOYEES
GROUP BY DEP_ID
HAVING AVG(SALARY) >= 60000;
```

- 4. Para el problema anterior, ordena los resultados de cada grupo en orden descendente de salario promedio.
- ▼ Haz clic aquí para la solución

SELECT DEP\_ID, AVG(SALARY) FROM EMPLOYEES GROUP BY DEP\_ID HAVING AVG(SALARY) >= 60000 ORDER BY AVG(SALARY) DESC;

## Conclusión

¡Felicidades! Has completado este laboratorio.

Al final de este laboratorio, serás capaz de:

- Utilizar patrones de cadena para filtrar los datos recuperados.
- Ordenar los datos recuperados según uno o más parámetros utilizando la declaración ORDER BY.
- Agrupar los datos con respecto a un parámetro.

### Author(s)

Abhishek Gagneja

Lakshmi Holla

Malika Singla



© IBM Corporation 2023. Todos los derechos reservados.

about:blank 9/9