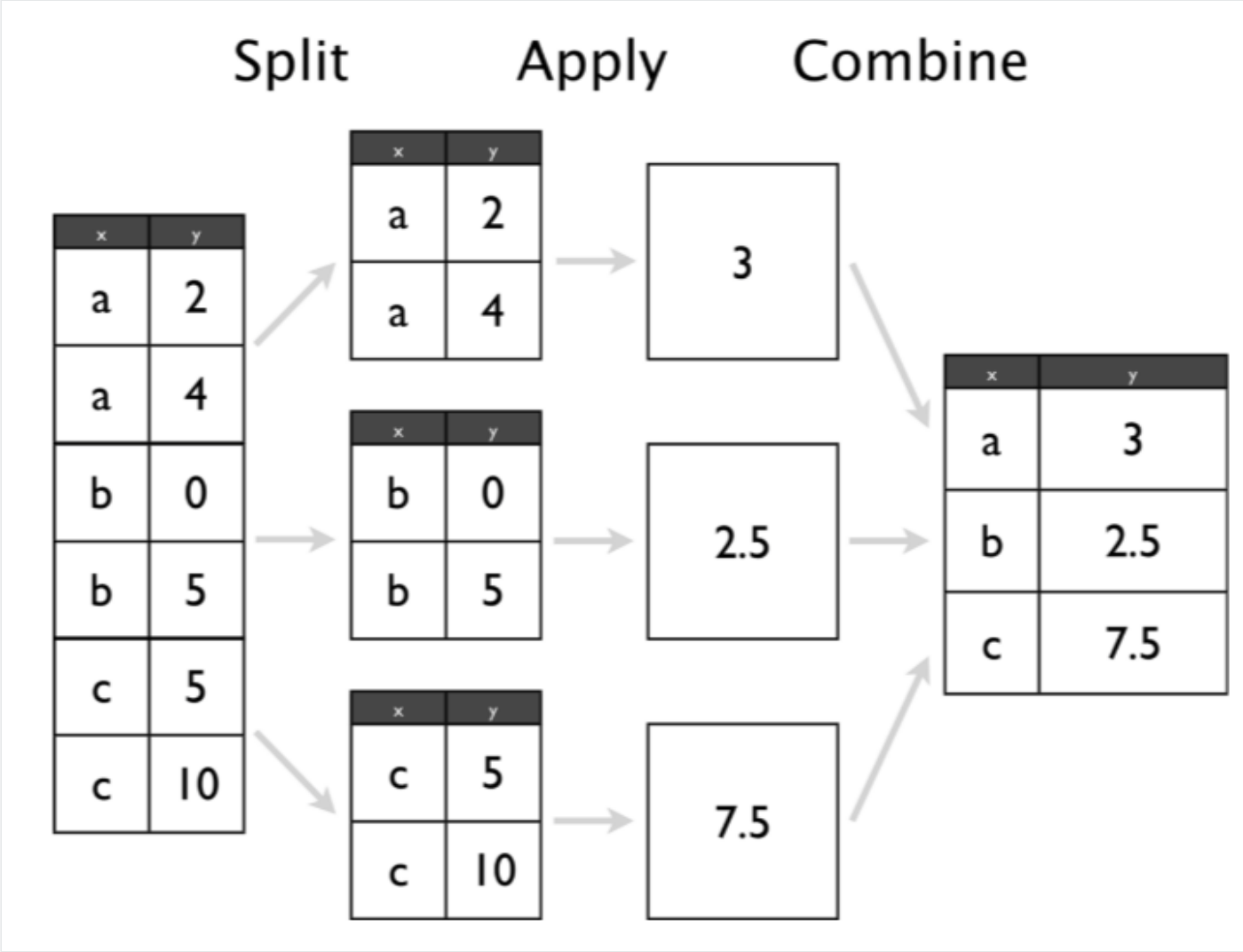
Agrupar por cláusula

La cláusula GROUP BY se utiliza en SQL con la instrucción SELECT para organizar datos similares en grupos. Combina los registros múltiples en una o más columnas usando algunas funciones. Generalmente, estas funciones son funciones agregadas como min(), max(),avg(), count() y sum() para combinar en columnas únicas o múltiples. Utiliza la estrategia **dividir-aplicar-combinar para el análisis de datos.**

- En el `split phase` , Divide los grupos con sus valores.
- En el `apply phase` , aplica la función de agregado y genera un solo valor.
- En el `combiner phase` , combina los grupos con valores únicos en un solo valor.



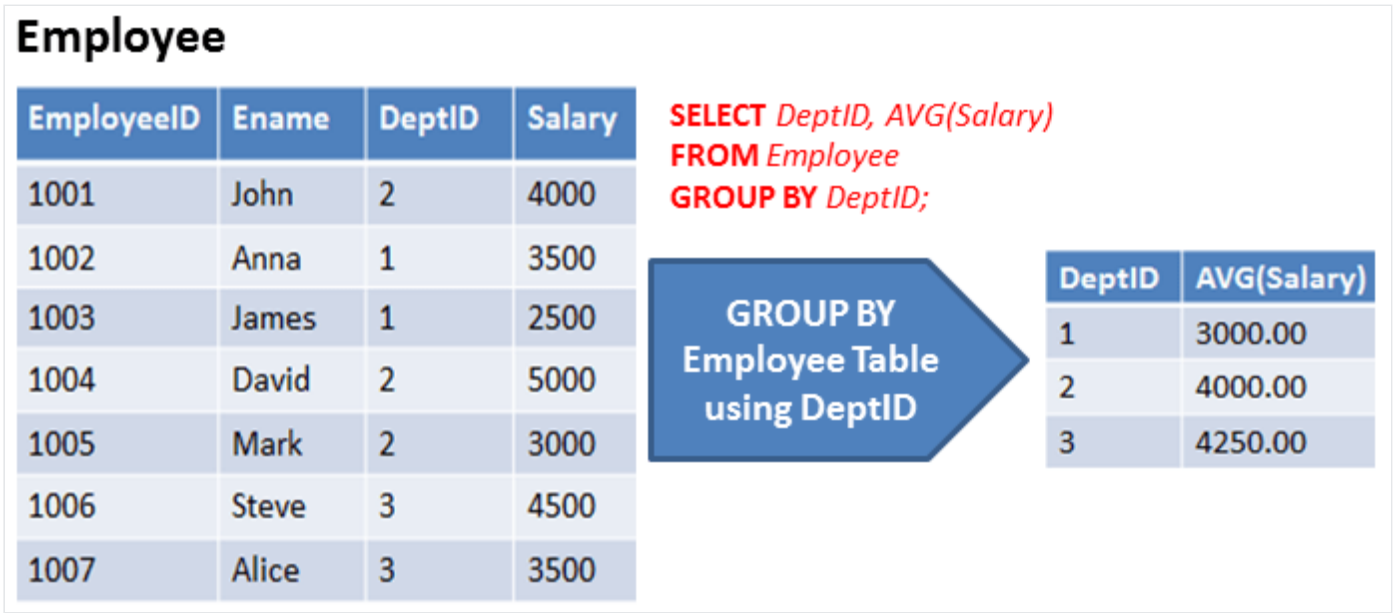
I

fuente de mago

Puntos para recordar:

- La cláusula GROUP BY se utiliza con la instrucción SELECT.
- GROUP BY agrega los resultados sobre la base de la columna seleccionada: COUNT, MAX, MIN, SUM, AVG, etc.
- GROUP BY devuelve solo un resultado por grupo de datos.

- La cláusula GROUP BY siempre sigue a la cláusula WHERE.
- La cláusula GROUP BY siempre precede a la cláusula ORDER BY (**http://slideplayer.com/slide/15440670/**).

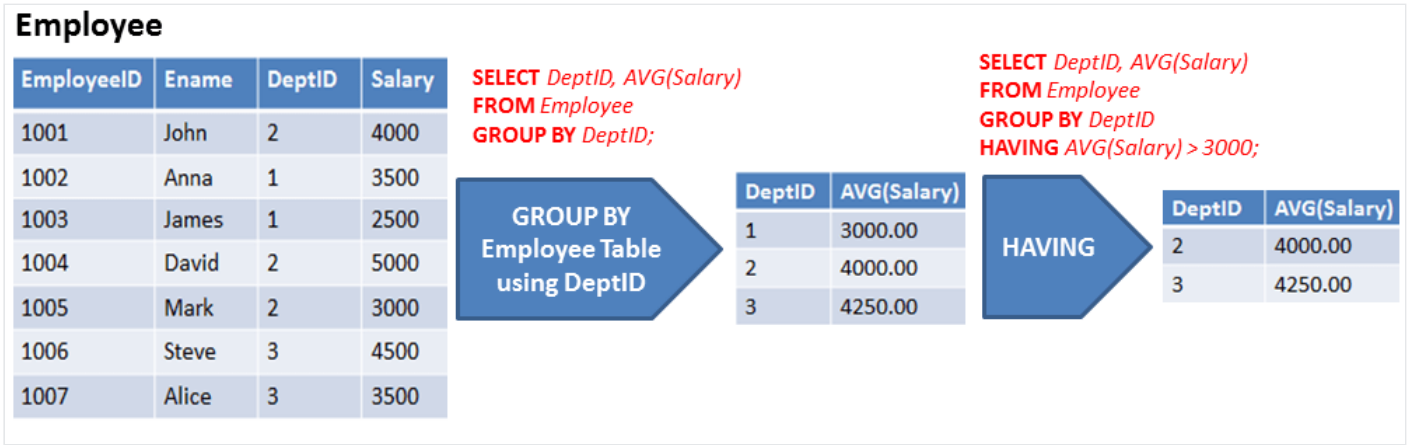


En el ejemplo anterior, la tabla se agrupa según la columna DeptID y el salario se agrega por departamento.

Tener cláusula

Cláusula HAVING utilizada en SQL como cláusula condicional con cláusula GROUP BY. Esta cláusula condicional devuelve filas donde los resultados de la función agregada coincidieron solo con las condiciones dadas. Se agregó en el SQL porque la cláusula WHERE no se puede combinar con resultados agregados, por lo que tiene un propósito diferente. El propósito principal de la Cláusula WHERE es tratar con registros individuales o no agregados.

- La cláusula HAVING siempre se utiliza en combinación con la cláusula GROUP BY.
- La cláusula HAVING restringe los datos en los registros de grupo en lugar de los registros individuales.
- WHERE y HAVING se pueden usar en una sola consulta.



En el ejemplo anterior, la tabla se agrupa en función de la columna DeptID y estas filas agrupadas se filtran mediante la cláusula HAVING con la condición AVG (Salario)> 3000.

Funciones agregadas

Funciones agregadas utilizadas para combinar el resultado de un grupo en uno solo, como COUNT, MAX, MIN, AVG, SUM, STDDEV y VARIANCE. Estas funciones también se conocen como funciones de varias filas.

- `SUM()` : Devuelve la suma o el total de cada grupo.
- `COUNT()` : Devuelve el número de filas de cada grupo.
- `AVG()` : Devuelve el promedio y la media de cada grupo.
- `MIN()` : Devuelve el valor mínimo de cada grupo.
- `MAX()` : Devuelve el valor mínimo de cada grupo.

Compare las cláusulas Have y Where en SQL

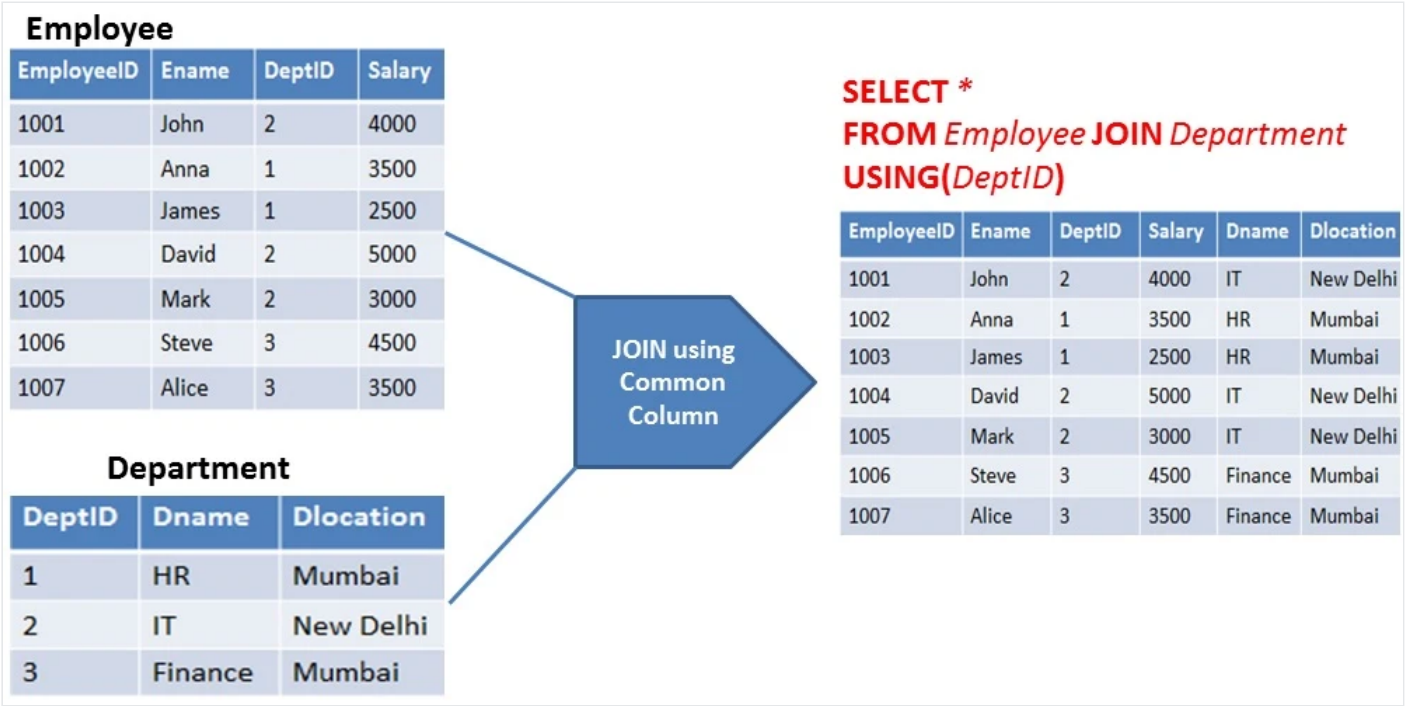
- En algunos casos, debe filtrar los registros individuales. En tales casos, puede usar la cláusula WHERE, mientras que en otros casos necesita filtrar los grupos con la condición específica. En tales casos, puede utilizar la cláusula HAVING.
- La cláusula WHERE filtra los registros tupla por tupla mientras que la cláusula HAVING filtra todo el grupo.
- Una consulta puede tener ambas cláusulas (cláusula WHERE y HAVING).
- La Cláusula Where se aplicó primero y luego la Cláusula Tener.
- La cláusula WHERE restringe los registros antes de la cláusula GROUP BY, mientras que la cláusula HAVING restringe los grupos después de que se ejecuta la cláusula GROUP BY.
- La cláusula WHERE se puede utilizar con SELECT, UPDATE, DELETE e INSERT, mientras que HAVING solo se puede utilizar con la instrucción SELECT.

Sno	Where Clause	Having Clause
1	The WHERE clause specifies the criteria which individual records must meet to be selcted by a query. It can be used without the GROUP BY clause	The HAVING clause cannot be used without the GROUP BY clause.
2	The WHERE clause selects rows before grouping.	The HAVING clause selects rows after grouping.
3	The WHERE clause cannot contain aggregate functions	The HAVING clause can contain aggregate functions.
4	WHERE clause is used to impose condition on SELECT statement as well as single row function and is used before GROUP BY clause	HAVING clause is used to impose condition on GROUP Function and is used after GROUP BY clause in the query
5	SELECT Column,AVG(Column_nmae)FROM Table_name WHERE Column > value GROUP BY Column_nmae	SELECT Columnq, AVG(Column_nmae)FROM Table_name WHERE Column > value GROUP BY Column_nmae Having column_name>or<value

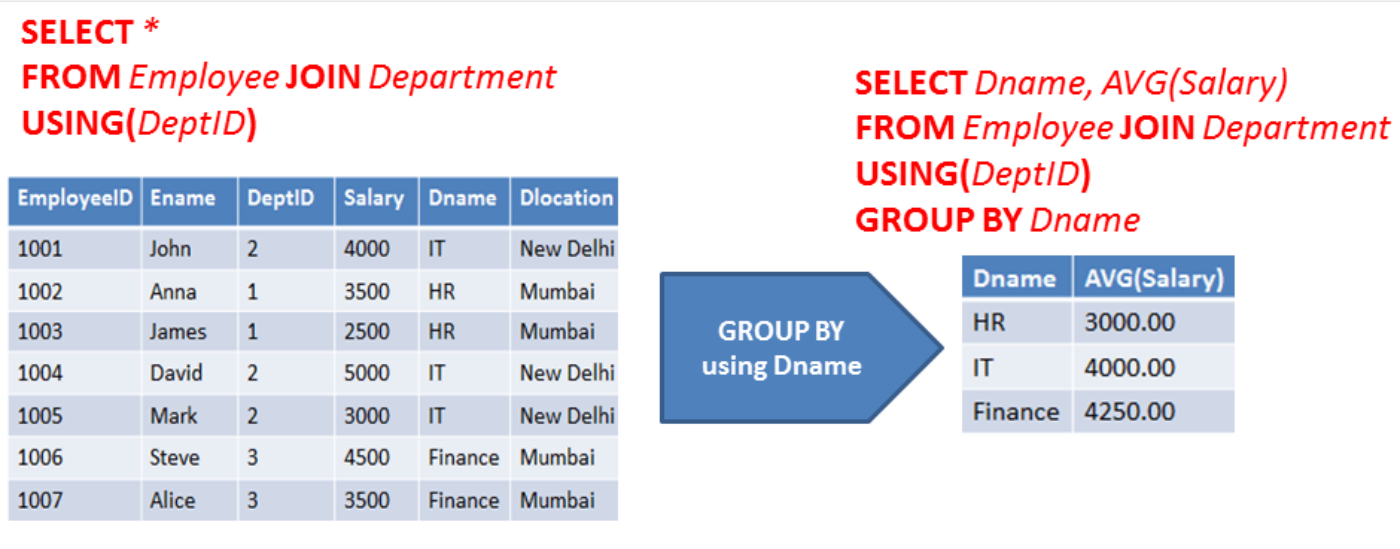
Fuente de imagen

GROUP BY con JOIN Ejemplo

La base de datos relacional normalizada desglosa la tabla compleja en tablas pequeñas, lo que le ayuda a eliminar la redundancia de datos, la inconsistencia y garantizar que no haya pérdida de información. Las tablas normalizadas requieren unir datos de varias tablas.



En el ejemplo anterior, el Empleado y el Departamento se unen mediante la columna común DeptID.



EmployeeID	Ename	DeptID	Salary	Dname	Dlocation
1001	John	2	4000	IT	New Delhi
1002	Anna	1	3500	HR	Mumbai
1003	James	1	2500	HR	Mumbai
1004	David	2	5000	IT	New Delhi
1005	Mark	2	3000	IT	New Delhi
1006	Steve	3	4500	Finance	Mumbai
1007	Alice	3	3500	Finance	Mumbai

Dname	AVG(Salary)
HR	3000.00
IT	4000.00
Finance	4250.00

En el ejemplo anterior, JOIN y GROUP BY ambas cláusulas se usan juntas en una sola consulta. Después de unir ambas tablas (Empleado y Departamento), se unió la tabla agrupada por nombre de Departamento.

GROUP BY Comparación con otra cláusula

Comparar GROUP BY y DISTINCT

DISTINCT devuelve los valores únicos presentes en la columna, mientras que GROUP BY devuelve elementos únicos/distintos con la columna resultante agregada. En el siguiente ejemplo, puede ver los valores DISTINCT en la dept tabla.

```
mysql> SELECT DISTINCT dept_id
    -> FROM dept;
+---------+
| dept_id |
+---------+
|       1 |
|       2 |
|       3 |
+---------+
```

Comparar GRUPO POR y ORDEN POR

ORDER BY devuelve elementos ordenados en orden ascendente y descendente, mientras que GROUP BY devuelve elementos únicos con la columna resultante agregada. En el siguiente ejemplo, puede ver ORDER BY o tabla de salarios ordenados.

```
mysql> SELECT *
    -> FROM emp
    -> ORDER BY salary;
+------+---------+---------+--------+-------+
| eid  | ename   | dept_id | salary | bonus |
+------+---------+---------+--------+-------+
|  103 | michell |       1 |   2500 |   400 |
|  105 | mark    |       2 |   3000 |  1000 |
|  102 | anna    |       1 |   3500 |   600 |
|  101 | john    |       2 |   4000 |     0 |
|  107 | harshal |       3 |   4000 |   100 |
|  106 | steve   |       3 |   4500 |     0 |
|  104 | david   |       2 |   5000 |   500 |
+------+---------+---------+--------+-------+
```

Asignación de práctica práctica

Nombre de la tabla: Libros

Columnas: ISBN, Título, Fecha de publicación, Precio, Editorial

Escriba consultas SQL para las siguientes declaraciones y comparta sus respuestas en los comentarios:

1. Determine cuántos libros hay en cada categoría.
2. Determine cuántos libros hay en la categoría Gestión.
3. Determine el precio promedio del libro de cada categoría.
4. Indique el precio del libro menos costoso en cada categoría.

Fuente: *esta tarea está inspirada en el libro "Oracle 11g SQL" de John Casteel.*

Conclusión

¡Felicitaciones, has llegado al final de este tutorial!

En este tutorial, ha cubierto muchos detalles sobre las cláusulas GROUP BY y HAVING. Ha aprendido qué son las cláusulas GROUP BY y HAVING con ejemplos, comparación entre cláusulas HAVING y WHERE en SQL, GROUP BY con JOIN y comparación GROUP BY con DISTINCT y ORDER BY. En la última sección, tiene una tarea de práctica práctica para evaluar su conocimiento.

Con suerte, ahora puede utilizar el concepto de cláusula GROUP BY y HAVING para analizar sus propios conjuntos de datos. ¡Gracias por leer este tutorial!

Si está interesado en aprender más sobre **SQL** , tome el curso **de SQL Intermedio** de DataCamp .