Claves para entender `HAVING`:

1. **`WHERE` vs. `HAVING`:**
   - `WHERE` filtra filas antes de agrupar.
   - `HAVING` filtra después de agrupar.
2. **Orden correcto:**
   - Primero se aplican `WHERE`, `GROUP BY` y las funciones de agregación.
   - Luego se aplica `HAVING`.

Name Null? Type ---------- -------- ------------

EMPLOYEE_ID NOT NULL NUMBER(6)

FIRST_NAME VARCHAR2(20)

LAST_NAME NOT NULL VARCHAR2(25)

EMAIL NOT NULL VARCHAR2(25)

PHONE_NUMBER VARCHAR2(20)

HIRE_DATE NOT NULL DATE

JOB_ID NOT NULL VARCHAR2(10)

SALARY NUMBER(8,2)

COMMISSION_PCT NUMBER(2,2)

MANAGER_ID NUMBER(6)

DEPARTMENT_ID NUMBER(4)

Consultas con dificultad ascendente para explicar el uso de `HAVING`:

1. Básica: Total de empleados por departamento con más de 3 empleados

```
SELECT department_id, COUNT(*) AS total_employees
FROM hr.employees
GROUP BY department_id
HAVING COUNT(*) > 3;
```

- **Explicación:**
  - Agrupa a los empleados por `department_id`.
  - Cuenta cuántos empleados hay en cada departamento.
  - Filtra solo los departamentos con más de 3 empleados.

2. Intermedia: Promedio salarial por departamento, mostrando solo aquellos con un promedio mayor a 5000

```
SELECT department_id, AVG(salary) AS avg_salary
FROM hr.employees
GROUP BY department_id
HAVING AVG(salary) > 5000;
```

- **Explicación:**
  - Agrupa los empleados por departamento.
  - Calcula el salario promedio en cada departamento.
  - Filtra los resultados para mostrar solo los departamentos con un salario promedio mayor a 5000.

3. Avanzada: Departamentos con salarios totales superiores a 20,000 y al menos 2 empleados

```
SELECT department_id, SUM(salary) AS total_salary, COUNT(*) AS
num_employees
FROM hr.employees
GROUP BY department_id
HAVING SUM(salary) > 20000 AND COUNT(*) >= 2;
```

- **Explicación:**
  - Agrupa los empleados por `department_id`.
  - Calcula el salario total (`SUM(salary)`) y el número de empleados (`COUNT(*)`) por departamento.
  - Filtra los departamentos donde el salario total supera los 20,000 **y** tienen al menos 2 empleados.

Aquí tienes un ejemplo de una **consulta con subconsulta** sencilla que utiliza la tabla `hr.employees` para responder una pregunta específica:

Consulta: Encuentra los empleados cuyo salario está por encima del promedio de todos los empleados.

```
SELECT employee_id, first_name, last_name, salary
FROM hr.employees
WHERE salary > (SELECT AVG(salary) FROM hr.employees);
```

Explicación Paso a Paso:

1. **Subconsulta (INNER QUERY):**
2. `SELECT AVG(salary) FROM hr.employees;`
   - Esta subconsulta calcula el salario promedio de todos los empleados en la tabla `hr.employees`.
3. **Consulta Principal (OUTER QUERY):**
4. `SELECT employee_id, first_name, last_name, salary`

```
5. FROM hr.employees
6. WHERE salary > (valor_calculado);
```

   - La consulta principal selecciona los `employee_id`, `first_name`, `last_name` y `salary` de los empleados.
   - Solo muestra aquellos empleados cuyo salario es mayor al resultado de la subconsulta (el promedio).

7. **Relación Entre Ambas Consultas:**
   - La subconsulta devuelve un valor (el salario promedio) que se utiliza como criterio en la cláusula `WHERE` de la consulta principal.

Ventaja de las Subconsultas:

- Las subconsultas permiten descomponer problemas complejos en pasos más pequeños y fáciles de entender.
- En este ejemplo, calculamos el promedio en un paso separado y luego usamos ese valor para filtrar los resultados.

Aquí tienes un ejemplo de una **subconsulta con `IN`**, utilizando la tabla `hr.employees`:

Consulta: Encuentra a los empleados que trabajan en departamentos con más de 5 empleados.

```
SELECT employee_id, first_name, last_name, department_id
FROM hr.employees
WHERE department_id IN (
    SELECT department_id
    FROM hr.employees
    GROUP BY department_id
    HAVING COUNT(*) > 5
);
```

Explicación Paso a Paso:

1. **Subconsulta (INNER QUERY):**

```
2. SELECT department_id
3. FROM hr.employees
4. GROUP BY department_id
5. HAVING COUNT(*) > 5;
```

   - Agrupa a los empleados por `department_id`.
   - Calcula cuántos empleados hay en cada departamento.
   - Devuelve los `department_id` de aquellos departamentos con más de 5 empleados.

6. **Consulta Principal (OUTER QUERY):**

```
7. SELECT employee_id, first_name, last_name, department_id
8. FROM hr.employees
9. WHERE department_id IN (resultados_subconsulta);
```

   - Busca en la tabla `hr.employees` todos los empleados cuyo `department_id` está en la lista de departamentos generada por la subconsulta.

10. **Relación Entre Ambas Consultas:**

    - La subconsulta crea una lista de `department_id` que cumplen con el criterio (más de 5 empleados).
    - La consulta principal filtra los empleados que pertenecen a esos departamentos.

¿Por qué usar `IN`?

- La cláusula `IN` es útil cuando necesitas verificar si un valor pertenece a un conjunto de resultados generados dinámicamente por una subconsulta.
- En este ejemplo, se usa para relacionar empleados con departamentos basados en un criterio agregado.