¡Claro! Aquí tienes algunos **tips clave** para realizar ejercicios con este tipo de relaciones en SQL, donde no tienes claves foráneas directas pero debes relacionar tablas mediante criterios específicos (como rangos de valores, condiciones lógicas, etc.):

**1️⃣ Entender el tipo de relación entre las tablas**

Antes de escribir la consulta, asegúrate de **comprender cómo se relacionan las tablas**. En el caso de las tablas EMP y SALGRADE, la relación no es directa (sin claves foráneas) pero se establece a través de una condición lógica (el salario de un empleado y los rangos de salario en SALGRADE).

**2️⃣ Identificar las columnas clave para la relación**

Identifica las **columnas** que permitirán que las tablas se unan:

* En este caso, no tenemos un campo común como DEPTNO, sino que la relación se establece a través de la **comparación del salario** del empleado con los rangos de salario de la tabla SALGRADE.

Por lo tanto, asegúrate de que entiendes qué datos compararás entre las tablas. **Busca columnas de comparación lógicas** como rangos, fechas o intervalos.

**3️⃣ Usar la condición adecuada para el JOIN**

Cuando no hay una clave foránea directa, puedes usar condiciones como:

* **BETWEEN**: Para rangos numéricos, como los salarios que deben estar entre LOSAL y HISAL.
* ON e.SAL BETWEEN s.LOSAL AND s.HISAL
* **=**: Para comparar valores exactos, como ID de empleados con ID de otra tabla, si corresponde.
* **IN** o **LIKE**: Para comparar con una lista de valores o patrones.

**4️⃣ Realizar una planificación previa del JOIN**

A veces es útil dibujar un **diagrama simple** de cómo se conectan las tablas, especialmente si no hay una relación directa entre ellas. En este caso, imagina cómo cada salario de EMP se debe comparar con el rango de la tabla SALGRADE. Esto ayuda a tener claro qué debe hacer cada parte de la consulta.

**5️⃣ Filtrar correctamente los resultados**

Usa **WHERE** o **HAVING** para filtrar los resultados y limitar la información que necesitas:

* Si solo te interesa un empleado, usa **WHERE e.EMPNO = 7698**.
* Si necesitas filtrar por un rango de salario específico, usa **WHERE e.SAL > 1000**.

**6️⃣ Practicar ejercicios similares**

Es fundamental que practiques diferentes **tipos de condiciones de JOIN**, como por ejemplo:

* **JOIN por rangos**: Relacionar una tabla con rangos de valores (como lo estamos haciendo con SAL y SALGRADE).
* **JOIN por fechas**: Relacionar eventos o registros de acuerdo con un rango de fechas (por ejemplo, registros de ventas que estén entre dos fechas específicas).
* **JOIN por coincidencias de patrones**: Relacionar registros de tablas basados en coincidencias parciales de texto con **LIKE**.

**7️⃣ Utilizar funciones de agregación si es necesario**

En algunos ejercicios, es posible que necesites calcular el **promedio**, **máximo** o **mínimo** de un conjunto de valores antes de aplicar un JOIN. Por ejemplo, si necesitas encontrar el rango de salario más común entre un grupo de empleados, usarías algo como:

SELECT AVG(e.SAL)

FROM EMP e

JOIN SALGRADE s

ON e.SAL BETWEEN s.LOSAL AND s.HISAL;

**8️⃣ Verificar si es necesario un INNER JOIN o un LEFT JOIN**

* Usa **INNER JOIN** si solo te interesa obtener los resultados cuando hay coincidencias entre las tablas.
* SELECT e.EMPNO, e.ENAME, e.SAL, s.GRADE
* FROM EMP e
* JOIN SALGRADE s
* ON e.SAL BETWEEN s.LOSAL AND s.HISAL;
* Usa **LEFT JOIN** si deseas obtener todos los empleados, incluso si no tienen una categoría salarial (en caso de que no coincidan con ningún rango de SALGRADE).

**9️⃣ Confirmar resultados con ejemplos de datos**

Para comprobar que tu consulta está correcta, crea un **ejemplo sencillo** de datos y ejecuta la consulta. Por ejemplo, si tienes un empleado con salario 2500, asegúrate de que el rango correspondiente de SALGRADE sea el adecuado.

**10️⃣ Optimizar consultas con índices**

Si las tablas son muy grandes, es útil asegurarte de que tienes **índices** en las columnas de comparación (por ejemplo, SAL en EMP y LOSAL, HISAL en SALGRADE) para hacer las consultas más rápidas.

**Ejemplo de ejercicio completo:**

**Ejercicio:** Dado un salario de un empleado, encuentra su categoría salarial.

**Pasos:**

1. Identificar las columnas relevantes:
   * EMP.SAL para el salario del empleado.
   * SALGRADE.LOSAL y SALGRADE.HISAL para los rangos salariales.
2. Escribir la consulta de JOIN:
3. SELECT e.EMPNO, e.ENAME, e.SAL, s.GRADE
4. FROM EMP e
5. JOIN SALGRADE s
6. ON e.SAL BETWEEN s.LOSAL AND s.HISAL;
7. Verificar los resultados para empleados con diferentes salarios para asegurarte de que el GRADE se asigna correctamente.

**Resumen final:**

* **Comprende la relación** entre las tablas (en este caso, usando rangos).
* **Usa condiciones adecuadas** para el JOIN (como BETWEEN para rangos).
* **Filtra los resultados** según sea necesario para obtener solo lo que buscas.
* **Practica** con diferentes tipos de relaciones no directas (rangos, fechas, patrones).
* **Verifica los resultados** para comprobar que las condiciones son correctas.

Si te quedan dudas o necesitas más ejemplos, ¡aquí estoy! 😄

Para crear el procedimiento ObtenerCategoriaSalarial, lo que necesitamos hacer es consultar la tabla **SALGRADE** según el salario del empleado y devolver el **GRADE** correspondiente.

Dado que tienes la base de datos y las tablas ya creadas, aquí te dejo el procedimiento en **PL/SQL** para determinar la categoría salarial (grade) de un empleado:

**Procedimiento: ObtenerCategoriaSalarial**

CREATE OR REPLACE PROCEDURE ObtenerCategoriaSalarial(

p\_empno IN EMP.EMPNO%TYPE, -- Número de empleado

p\_grade OUT SALGRADE.GRADE%TYPE -- Salida: grado salarial

)

IS

v\_sal NUMBER; -- Variable para almacenar el salario del empleado

BEGIN

-- Obtener el salario del empleado

SELECT SAL INTO v\_sal

FROM EMP

WHERE EMPNO = p\_empno;

-- Consultar el grado salarial basado en el salario del empleado

SELECT GRADE INTO p\_grade

FROM SALGRADE

WHERE v\_sal BETWEEN LOSAL AND HISAL;

-- Aquí no es necesario COMMIT porque solo estamos consultando

END ObtenerCategoriaSalarial;

/

**Explicación del procedimiento:**

1. **Entrada (p\_empno)**: El procedimiento recibe un número de empleado (p\_empno), que es el que se usará para consultar el salario en la tabla EMP.
2. **Salida (p\_grade)**: El procedimiento devuelve el grado salarial (GRADE) correspondiente en la tabla SALGRADE.
3. **Lógica**:
   * Primero, obtenemos el salario del empleado a través de su EMPNO de la tabla EMP.
   * Luego, usamos ese salario para buscar el GRADE correspondiente en la tabla SALGRADE, que tiene rangos de salarios definidos entre LOSAL y HISAL.

Esto es posible porque la columna SAL en la tabla EMP debe caer dentro de un rango de la tabla SALGRADE para determinar su GRADE.

**Ejemplo de uso del procedimiento:**

Imaginemos que quieres saber el grado salarial de un empleado con el EMPNO = 7698.

1. **Ejecutas el procedimiento**:

SET SERVEROUTPUT ON;

DECLARE

v\_grade SALGRADE.GRADE%TYPE;

BEGIN

ObtenerCategoriaSalarial(7698, v\_grade);

DBMS\_OUTPUT.PUT\_LINE('El grado salarial es: ' || v\_grade);

END;

/

1. **Resultado esperado**:

Si el salario de 7698 está en el rango definido en SALGRADE para un determinado GRADE, se imprimirá el valor de GRADE. Por ejemplo:

El grado salarial es: 4

Este número depende del salario del empleado. Si el salario de 7698 está entre LOSAL y HISAL para el GRADE 4, devolverá ese valor.

**Verificación**

Después de ejecutar el procedimiento, puedes usar la siguiente consulta para verificar que los valores de salario y grado coinciden:

SELECT EMPNO, ENAME, SAL

FROM EMP

WHERE EMPNO = 7698;

SELECT GRADE, LOSAL, HISAL

FROM SALGRADE;

Estas consultas te ayudarán a confirmar que el cálculo es correcto comparando el salario con los rangos de la tabla SALGRADE.

Si tienes alguna duda sobre cómo usar el procedimiento o necesitas más detalles, ¡avísame! 😊