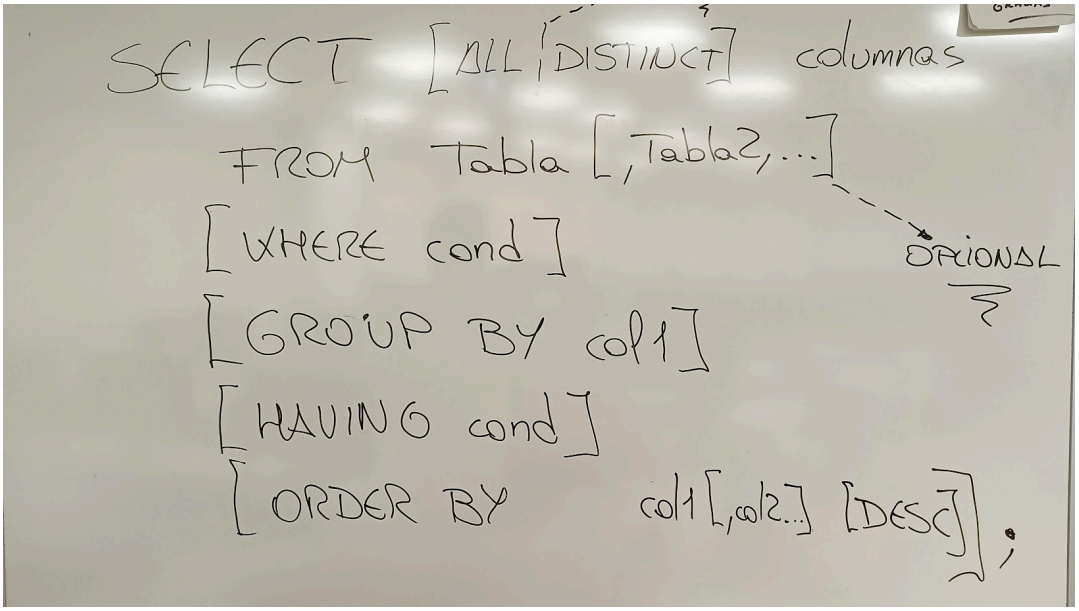
Bases de Datos Oracle SQL



Estructura de una Consulta SELECT

Una consulta básica en SQL sigue esta estructura:

SELECT [DISTINCT] columna1, columna2, ...
FROM nombre_tabla
[WHERE condición]
[GROUP BY columna(s)]
[HAVING condición]
[ORDER BY columna(s) [ASC|DESC]];

Componentes Principales:

1. **SELECT:** Especifica las columnas que deseas consultar. Puedes usar * para seleccionar todas las columnas de una tabla.
2. **FROM:** Indica la tabla de donde provienen los datos.
3. **WHERE:** Filtra los resultados según una condición.
4. **GROUP BY:** Agrupa resultados basados en una o más columnas.
5. **HAVING:** Filtra los grupos generados por `GROUP BY`.
6. **ORDER BY:** Ordena los resultados de la consulta.
    - Orden ascendente por defecto: `ORDER BY columna`.
    - Orden descendente: `ORDER BY columna DESC`.

Eliminación de Duplicados con DISTINCT

El uso de `DISTINCT` elimina filas duplicadas en una columna o conjunto de columnas.

Ejemplo:

```
SELECT DISTINCT oficio
FROM emple
ORDER BY oficio;
```

Resultado:

```
OFICIO
----------
ANALISTA
DIRECTOR
EMPLEADO
PRESIDENTE
VENDEDOR
```

Nota: Con `DISTINCT`, los valores repetidos se omiten del resultado.

Orden Descendente con DISTINCT

```
SELECT DISTINCT oficio
FROM emple
ORDER BY oficio DESC;
```

Resultado:

```
OFICIO
----------
VENDEDOR
PRESIDENTE
EMPLEADO
DIRECTOR
ANALISTA
```

Uso de la Cláusula WHERE

La cláusula `WHERE` filtra filas según condiciones específicas. Para búsquedas de texto, es útil utilizar la función `UPPER()` para evitar problemas con la distinción entre mayúsculas y minúsculas.

Ejemplo: Filtrar empleados con el cargo "DIRECTOR"

```
SELECT apellido, salario
FROM emple
WHERE UPPER(oficio) = 'DIRECTOR';
```

Resultado:

```
APELLIDO      SALARIO
---------- ----------
JIMENEZ          2900
NEGRO            3005
CEREZO           2885
```

Notas Importantes sobre `WHERE`:

- **Búsquedas Insensibles a Mayúsculas:** Usar `UPPER(campo) = 'VALOR'` asegura que el filtro sea insensible a las mayúsculas.
- **Evitar Errores:** Verifica que los nombres de columnas y tipos de datos coincidan correctamente.

Ordenar Resultados con ORDER BY

El ordenamiento se realiza al final de la consulta:

1. Ordenar por una columna específica:

```
ORDER BY columna;
```

2. Ordenar de forma descendente:

```
ORDER BY columna DESC;
```

3. Ordenar por múltiples columnas:

```
ORDER BY columna1, columna2 DESC;
```

Ejemplo: Empleados del departamento 20 ordenados por apellido:

```
SELECT apellido, oficio, emp_no, dept_no
FROM emple
WHERE dept_no = 20
```

ORDER BY apellido;

Ejemplo: Analistas ordenados por número de empleado:

```
SELECT apellido, oficio, emp_no, dept_no
FROM emple
WHERE UPPER(oficio) = 'ANALISTA'
ORDER BY emp_no;
```

Cálculo de Promedios

Puedes realizar cálculos en las consultas utilizando operadores aritméticos:

Ejemplo: Calcular la media de tres notas por alumno:

```
SELECT nombre_alumno,
      (nota1 + nota2 + nota3) / 3 AS promedio
FROM notas_alumnos;
```

Validaciones y Estructuras de Ejercicios Prácticos

Ejemplo 1: Consultas con Relaciones entre Tablas

En este ejercicio, se consulta información combinada de dos tablas:

```
SELECT emple.apellido, emple.oficio, depart.loc
FROM emple, depart
WHERE emple.dept_no = depart.dept_no
  AND emple.oficio = 'ANALISTA';
```

Lecciones Aprendidas:

- Relacionar tablas usando claves foráneas (`emple.dept_no = depart.dept_no`).
- Filtrar resultados con múltiples condiciones (`emple.oficio = 'ANALISTA'`).

Ejemplo 2: Subconsultas en Condiciones

Seleccionar empleados bajo la dirección de un jefe específico:

```
SELECT *
FROM emple, depart
WHERE emple.dir = (
  SELECT emp_no
  FROM emple
  WHERE apellido = 'CEREZO'
);
```

Lecciones Aprendidas:

- Uso de subconsultas en el `WHERE` para filtrar datos específicos.
- Seleccionar columnas de múltiples tablas relacionadas.

Ejemplo 4: Excluir Valores Específicos

Consultar departamentos sin empleados asociados:

```
SELECT *
FROM depart
WHERE dept_no NOT IN (
  SELECT dept_no
  FROM emple
);
```

Lecciones Aprendidas:

- `NOT IN` para excluir subconjuntos específicos de datos.
- Identificar relaciones vacías entre tablas.

Ejemplo 6: Uso de Operadores Agregados

Seleccionar empleados cuyo salario sea mayor que cualquier salario en un departamento específico:

```
SELECT apellido, salario
FROM emple
WHERE salario > ALL (
  SELECT salario
  FROM emple
  WHERE dept_no = 20
);
```

Lecciones Aprendidas:

- Uso de ALL para comparar con todos los valores de un subconjunto.
- Subconsultas con filtros adicionales.

Ejemplo 7: Rangos y Condiciones Múltiples

Seleccionar registros dentro de un rango específico:

```
SELECT *
FROM libreria
WHERE ejemplares >= 8 AND ejemplares <= 15;
```

Lecciones Aprendidas:

- Especificar condiciones de rango con operadores (>=, <=).
- Combinar condiciones con AND para mayor precisión.

Ejemplo 9: Comparaciones con Subconsultas

Consultar temas con menos ejemplares que un tema específico:

```
SELECT tema
FROM libreria
WHERE ejemplares < (
  SELECT ejemplares
  FROM libreria
  WHERE tema = 'MEDICINA'
);
```

Lecciones Aprendidas:

- Subconsultas para comparar valores específicos en la misma tabla.
- Identificar registros relativos a un criterio de referencia.

Ejemplo 11: Búsquedas con Patrones

Seleccionar asignaturas con un patrón específico en su nombre:

```
SELECT nombre
FROM asignaturas
```

```
WHERE nombre LIKE '%o%o%o%'
  AND cod IN (
    SELECT cod
    FROM notas
    WHERE dni IN (
      SELECT dni
      FROM alumnos
      WHERE pobla = 'Madrid'
    )
  );
```

Lecciones Aprendidas:

- Uso de `LIKE` para buscar patrones en texto.
- Combinación de múltiples subconsultas para filtrar datos complejos.

Buenas Prácticas al Escribir Consultas SQL

Comparaciones de Texto: Siempre usa `UPPER()` para evitar problemas de mayúsculas/minúsculas.

```
WHERE UPPER(oficio) = 'ANALISTA';
```

1. **Selección de Columnas:** Especifica las columnas necesarias en lugar de usar `SELECT *` para mejorar el rendimiento.

2. **Uso de DISTINCT:** Úsalo solo cuando necesites eliminar duplicados, ya que puede afectar el rendimiento en tablas grandes.

3. **Ordenamiento:** Aplica `ORDER BY` sólo cuando el orden de los resultados sea relevante.

Funciones Comunes en SQL

Operaciones Aritméticas:

```
-- Cálculo de salario anual
SELECT salario * 12 AS salario_anual
FROM emple;

-- Promedios
SELECT (nota1 + nota2 + nota3) / 3 AS promedio
```

FROM notas_alumnos;

Funciones de Texto:

1. **Conversión a Mayúsculas/Minúsculas:**

```
SELECT UPPER(nombre), LOWER(apellido)
FROM emple;
```

2. **Concatenación de Texto:**

```
SELECT nombre || ' ' || apellido AS nombre_completo
FROM emple;
```

Redondeo:

```
SELECT ROUND(salario, 2) AS salario_redondeado
FROM emple;
```

EN CASO DE QUE TE METAN LOS DATOS MUY MAL Y QUIERA SEGUIR VIENDO TODO:

→ EN EL WHERE PODEMOS UTILIZAR UPPER(CAMPO) = ‘VALOR A BUSCAR’, LO SUBE A MAYÚSCULAS EN EL SELECT PERO EN NADA MÁS.

```
SQL> SELECT APELLIDO, SALARIO
  2  FROM TEMPLE
  3  WHERE UPPER(OFICIO) = 'DIRECTOR';
```

//Convertimos todo a mayúsculas(nos aseguramos que seleccione todos los directores a pesar de no estar escrito en MAYÚS)

```
APELLIDO      SALARIO
---------- ----------
JIMENEZ         2900
NEGRO           3005
CEREZO          2885
```

ORDENAR:
AL FINAL DEL TODO PODEMOS PONER UN ORDER BY (COLUMNA QUE QUERAMOS ORDENAR POR, ES DECIR -> ATRIBUTO CAMPO DE UNA TABLA.)

EMPLEADOS DEL DEPARTAMENTO 20 ORDENADOS POR APELLIDO, QUIERO NUM, APELLIDO, OFICIO Y DEPARTAMENTO

```
SELECT APELLIDO, OFICIO, EMP_NO, DEPT_NO
FROM EMPLE
WHERE DEPT_NO = 20
ORDER BY APELLIDO;
```

ANALISTAS, ORDENADO POR NÚMERO DE EMPLEADO

```
SELECT APELLIDO, OFICIO, EMP_NO, DEPT_NO
FROM EMPLE
WHERE UPPER(OFICIO) = 'ANALISTA'
ORDER BY EMP_NO;
```

EN LAS CONDICIONES DE STRING SIEMPRE PONER LOS UPPERS

CALCULAR MEDIAS

```
SELECT NOMBRE_ALUMNO, (NOTA1 + NOTA2 + NOTA3) / 3
FROM NOTAS_ALUMNOS;
```

COMMIT

Tras hacer la inserción de datos lo mejor es hacer commits ya que si no los haces se guarda en local y ya; si lo haces se guarda a nivel general , al ser multiusuario todo el mundo lo debe ver.

COMBINACIÓN DE TABLAS CONDICIÓN DE ENLACE

Siempre que relacionemos algo usando 2 tablas tendremos que poner la condición where que te iguale el campo que es igual en ambas tablas; si no generamos productos cartesianos, esto se llama condición de enlace; por ejemplo:

```
SELECT*
from EMPLE, DEPART
where EMPLE.DEPT_NO = DEPART.DEPT_NO     (Condición de enlace)
and UPPER (LOC) = 'BARCELONA';
```

WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO --> ES LA CONDICIÓN DE ENLACE, LO MÁS IMPORTANTE, TE ESTÁ DICIENDO QUE EN UNA RELACIÓN 1-N
AMBAS TABLAS ESTÁN UNIDAS POR EL DEPT_NO, ES DECIR HAY QUE PONERLE ESA CONDICIÓN SIEMPRE CON LO QUE TIENEN EN COMÚN.

N TABLAS = N-1 CONDICIONES DE ENLACE

2 TABLAS ---> 1 CONDICIÓN
3 TABLAS ---> 2 CONDICIONES
4 TABLAS ---> 3 CONDICIONES
5 TABLAS ---> 4 CONDICIONES

ENCIMA SERÍA DINÁMICO, SI LE METEMOS ALGO MÁS LO ACTUALIZA.

OPERADOR DE CONJUNTOS / SUBCONSULTAS

(LAS SUBCONSULTAS SÓLO PARA CUANDO NO HAYA OTRO REMEDIO)

Otra forma de realizar lo que hemos hecho con la condición de enlace:

```
SELECT APELLIDO
FROM EMPLE
WHERE DEPT_NO = (SELECT DEPT_NO FROM DEPART
WHERE UPPER(LOC) = 'BARCELONA');
```

-> Es como una operación matemática , se ejecuta primero lo de dentro del paréntesis y luego el selecto de fuera.

Es como si hicieramos 3*(7+3) - Primero paréntesis y luego multiplicación.

La cosa es que la condición de enlace va más rápido y siempre será mejor utilizar las condiciones de enlace; no obstante, cuando no se puedan utilizar las subconsultas habrá que usar esto también.

EJEMPLO 1: Habrá casos en los que necesitaré si o si una subconsulta para recuperar datos, por ejemplo cuando quiera saber algo que se basa en una información que puede cambiar a lo largo del tiempo, por ejemplo los apellidos de los empleados que tienen el mismo oficio que 'Gil'. Si por lo que sea mañana 'Gil' cambia de oficio la consulta debe seguir funcionando. Para eso la basaré en una subconsulta que me dará el oficio de gil en el momento de ejecución y de este modo la consulta funcionará siempre aunque cambiemos en el futuro de oficio a Gil.

Le meteremos otro select incrustado para que funcione; da igual que haya muchos select en un mismo query -> así funcionará siempre con cualquier dato que haya; las querys funcionarán siempre.

Composición:

```
Select apellido from emple
Where upper(oficio) = (Select Upper(Oficio) From emple
where upper(apellido)= 'GIL');
```

APELLIDO

GIL
FERNANDEZ

= Aquí es dónde vendría nuestra incertidumbre y metemos otro select.

Pero…

Atención: Si añado a un segundo empleado que se apellida ‘Gil’ y además tiene un oficio diferente al primer ‘Gil’...

Probamos.

Insert into EMPLE values (8000, ‘GIL’, ‘VENDEDOR’, NULL, SYSDATE, 1600, 100, 10);

```
SQL> SELECT *
  2  FROM EMPLE
  3  WHERE APELLIDO = 'GIL';

    EMP_NO APELLIDO   OFICIO            DIR FECHA_ALT     SALARIO   COMISION
---------- ---------- ---------- ---------- ---------- ---------- ----------
   DEPT_NO
----------
      7788 GIL        ANALISTA         7566 09/11/1991       3000
        20

      8000 GIL        VENDEDOR              03/02/2025       1600        100
        10
```

Ahora tenemos dos Gil con dos oficios distintos; uno analista y otro vendedor, **entonces si probamos la consulta de antes**:

Select apellido from emple
Where upper(oficio) **=** (Select Upper(Oficio) From emple
where upper(apellido)= ‘GIL’);

```
ERROR en lÝnea 2:
ORA-01427: la subconsulta de una sola fila devuelve mßs de una fila
```

Error: La subconsulta de una sola fila devuelve más de una fila.

Dará error porque + cómo solucionarlo…

-Si la subconsulta está bien y la consulta también está bien lo que falla es el ‘=’ ; el tema es que el ‘=’ es un operador binario !!!, es decir sólo funciona poniendo un único valor a cada lado; es decir solo puedes igualar un valor , si tenemos un conjunto de valores para comparar el ‘=’ no sabe hacerlo; entonces tendremos que usar el **IN**; es decir, la subconsulta devuelve dos valores y el ‘=’ no sabe compararlo porque solo funciona con un dato a cada

lado, es un comparador binario, usa un operador de conjuntos un comparador de conjuntos sería el **IN**.

Entonces la solución sería la implementación del IN.

SOLUCIÓN:

```
Select apellido from emple
Where upper(oficio) IN (Select Upper(Oficio) From emple
where upper(apellido)= 'GIL');
```

```
APELLIDO
----------
GIL
FERNANDEZ
ARROYO
SALA
MARTIN
TOVAR
GIL
```

-**COROLARIO** (Enseñanza): Si no estoy seguro de que la subconsulta devuelva un único valor (y sólo lo devolverá cuando la condición sea sobre una PK!!!)... deberé utilizar operadores de conjuntos como 'IN' o 'NOT IN'; si hacemos referencia a la primary key podremos usar un '=' ya que estamos comparando de forma binaria, únicamente con uno ; pero , si comparamos con varios deberemos usar un 'IN', por eso lo mejor siempre será curarnos en salud, es decir prevenir y usar un 'IN'

EJEMPLO 2: Empleados que cobren más que Gil… pero atención, porque puede que haya dos empleados que se llamen 'GIL' y el comparador '>' no funcione porque también es un operador binario.

PLANTEAMIENTO A PRIMERA VISTA:

```
SELECT APELLIDO FROM EMPLE
WHERE SALARIO > (SELECT SALARIO FROM EMPLE
WHERE UPPER(APELLIDO) = 'GIL');
```

-> ASÍ DARÁ ERROR, EL PROBLEMA ES EL MISMO, ESTAMOS UTILIZANDO UN OPERADOR DE COMPARACIÓN BINARIO Y NO UNO DE CONJUNTOS; EL > ES BINARIO.

SOLUCIÓN 1: Utilizar la palabra/s clave o reservada/s 'ANY' O 'ALL'.

ANY: Que cualquiera.
ALL: Que todos.

```
SELECT APELLIDO FROM EMPLE
WHERE SALARIO > ALL (SELECT SALARIO FROM EMPLE
WHERE UPPER(APELLIDO) = 'GIL');

APELLIDO
----------
NEGRO
REY
```

SOLUCIÓN 2: Adelantándonos al tema 4… podría utilizar una función… en concreto 'MAX' que da el máximo valor.

```
SELECT APELLIDO FROM EMPLE
WHERE SALARIO > ALL (SELECT MAX (SALARIO) FROM EMPLE
WHERE UPPER(APELLIDO) = 'GIL');
```

-La solución ya iría metida en el propio select.

```
APELLIDO
----------
NEGRO
REY
```

EJERCICIOS PRÁCTICA CONDICIÓN DE ENLACE, SUBCONSULTAS, ETC…

Ejercicios de Condición de Enlace

1. **Muestra todos los empleados junto con el nombre del departamento en el que trabajan.**
   Resultado esperado: Listado de empleados con su apellido, oficio y nombre del departamento.
2. **Muestra el nombre del departamento y el apellido de los empleados que trabajan en la ciudad de 'Madrid'.**
   Resultado esperado: Los apellidos de los empleados que trabajan en el departamento cuya localización es Madrid.
3. **Muestra los empleados que tienen el mismo oficio que 'JIMÉNEZ'.**
   Resultado esperado: Listado de empleados con el mismo oficio que JIMÉNEZ.
4. **Lista los empleados cuyo salario sea mayor al salario promedio de todos los empleados.**
   Resultado esperado: Apellidos de empleados cuyo salario es superior a la media.
5. **Encuentra los empleados cuyo salario sea mayor que el salario del empleado 'MARTIN'.**
   Resultado esperado: Apellidos de empleados con salario mayor al de MARTÍN.

Ejercicios de Subconsultas con Operadores Binarios e IN

6. **Muestra los empleados que trabajan en el mismo departamento que ‘FERNÁNDEZ’.**
   Resultado esperado: Listado de apellidos de empleados que trabajan en el mismo departamento que FERNÁNDEZ.
7. **Lista los empleados cuyo salario sea mayor que el salario mínimo de todos los empleados.**
   Resultado esperado: Apellidos de empleados con un salario mayor al salario mínimo registrado.
8. **Encuentra los empleados cuyo salario es mayor que cualquier salario de los empleados con el oficio de ‘VENDEDOR’.**
   Resultado esperado: Empleados cuyo salario es mayor que al menos un vendedor.
9. **Muestra los empleados cuyo salario es menor que todos los empleados con el oficio de ‘DIRECTOR’.**
   Resultado esperado: Empleados que cobran menos que cualquier director.
10. **Encuentra los empleados que trabajan en un departamento que no tiene empleados en ‘Barcelona’.**
    Resultado esperado: Apellidos de empleados que no están en un departamento ubicado en Barcelona.

Ejercicios con IN, NOT IN, ANY, ALL y Funciones Agregadas

11. Lista los empleados que ganan más que todos los empleados cuyo oficio sea ‘ANALISTA’.
Resultado esperado: Apellidos de empleados con salario superior al máximo salario de los analistas.
12. Encuentra los empleados cuyo salario está en el mismo rango de los salarios de los empleados con el oficio de ‘EMPLEADO’.
Resultado esperado: Empleados cuyo salario es igual a alguno de los empleados con
13. Muestra los empleados cuyo salario sea superior al de ‘TOVAR’, pero inferior al de ‘NEGRO’.
Resultado esperado: Apellidos de empleados cuyo salario esté entre los salarios de TOVAR y NEGRO.
14. Muestra los empleados que no trabajan en ninguno de los departamentos que tienen empleados con oficio ‘VENDEDOR’.
Resultado esperado: Apellidos de empleados que trabajan en departamentos sin vendedores.
15. Lista los empleados que NO tienen comisión.
Resultado esperado: Apellidos de empleados cuyo campo de comisión sea NULL.

SOLUCIONES:

1. SELECT APELLIDO, DNOMBRE
   FROM EMPLE, DEPART
   WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO;

2. SELECT APELLIDO, DNOMBRE
   FROM EMPLE, DEPART
   WHERE DEPART.DEPT_NO = EMPLE.DEPT_NO
   AND UPPER(LOC) = 'MADRID';

3. SELECT APELLIDO, OFICIO
   FROM EMPLE
   WHERE OFICIO IN (SELECT OFICIO
   FROM EMPLE
   WHERE UPPER(APELLIDO) = 'JIMENEZ');

4. SELECT APELLIDO, OFICIO, SALARIO
   FROM EMPLE
   WHERE SALARIO > (SELECT AVG(SALARIO)
   FROM EMPLE);

5. SELECT APELLIDO, OFICIO, SALARIO
   FROM EMPLE
   WHERE SALARIO > ALL (SELECT SALARIO
   FROM EMPLE
   WHERE UPPER(APELLIDO) = 'MARTIN');

6. SELECT APELLIDO, OFICIO, DEPT_NO
   FROM EMPLE
   WHERE DEPT_NO IN (SELECT DEPT_NO
   FROM EMPLE
   WHERE UPPER(APELLIDO) = 'FERNANDEZ');

7. SELECT APELLIDO, OFICIO, SALARIO
   FROM EMPLE
   WHERE SALARIO > (SELECT MIN (SALARIO)
   FROM EMPLE);

8. SELECT APELLIDO, OFICIO, SALARIO
   FROM EMPLE
   WHERE SALARIO >ANY (SELECT SALARIO
   FROM EMPLE
   WHERE UPPER(OFICIO) = 'VENDEDOR');

9. SELECT APELLIDO, OFICIO, SALARIO,
   FROM EMPLE
   WHERE SALARIO <ALL (SELECT SALARIO

```
FROM EMPLE
WHERE UPPER(OFICIO) = 'DIRECTOR');
```

10.
```
SELECT APELLIDO, OFICIO, LOC
FROM EMPLE, DEPART
WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO
AND EMPLE.DEPT_NO NOT IN (
SELECT DISTINCT DEPT_NO
FROM EMPLE, DEPART
WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO
AND UPPER(LOC) = 'BARCELONA');
```

11.
```
SELECT APELLIDO, OFICIO, SALARIO
FROM EMPLE
WHERE SALARIO > ALL (SELECT SALARIO
FROM EMPLE
WHERE UPPER(OFICIO) = 'ANALISTA');
```

12.
```
SELECT APELLIDO, OFICIO, SALARIO
FROM EMPLE
WHERE SALARIO IN ( SELECT SALARIO
FROM EMPLE
WHERE UPPER(OFICIO) = 'EMPLEADO');
```

13.
```
SELECT APELLIDO, OFICIO, SALARIO
FROM EMPLE
WHERE SALARIO > (SELECT SALARIO
FROM EMPLE
WHERE UPPER(APELLIDO) = 'TOVAR')
AND SALARIO < (SELECT SALARIO
FROM EMPLE
WHERE UPPER(APELLIDO) = 'NEGRO');
```

14.
```
SELECT APELLIDO, OFICIO, DEPT_NO
FROM EMPLE
WHERE DEPT_NO NOT IN (SELECT DEPT_NO
FROM EMPLE
WHERE UPPER(OFICIO) = 'VENDEDOR');
```

15.
```
SELECT APELLIDO, OFICIO, DEPT_NO, COMISION
FROM EMPLE
WHERE COMISION IS NULL OR COMISION = 0;
```

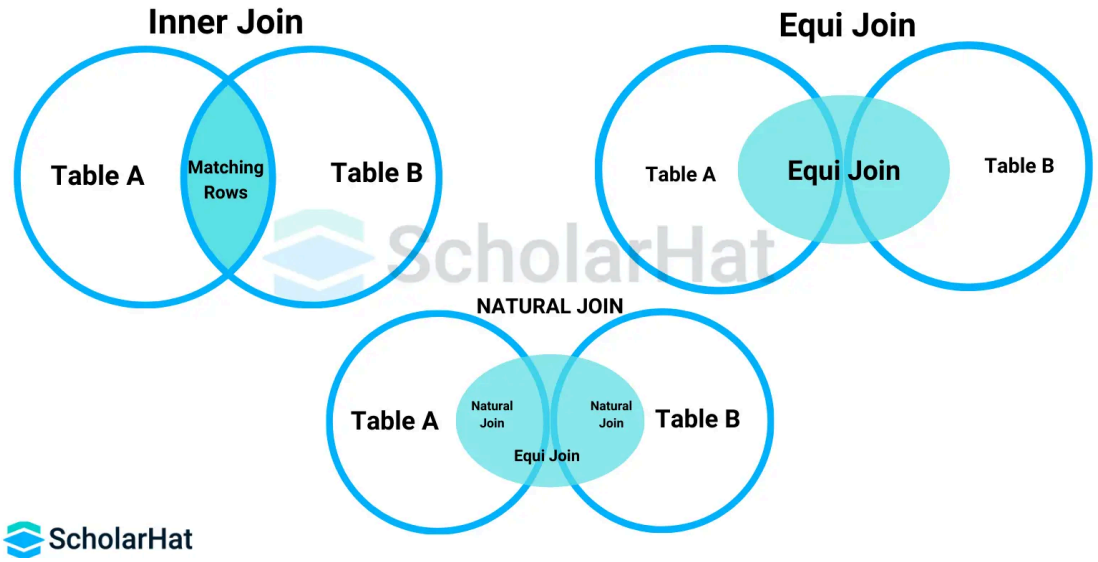
JOIN (= REJUNTAR)

Lo que conocíamos hasta ahora:

Select NOMBRE
FROM CLIENTE, COCHE
WHERE cliente.ID_CLIENTE = coche.ID_CLIENTE
and upper(MARCA) = ‘OPEL’;

-> se puede seguir trabajando así con condiciones de enlace pero hay una serie de joins que también se emplean y hacen lo mismo que la condición de enlace.

OPERACIONES DE JOIN



- **NATURAL Join**: Une dos tablas de forma natural, uniéndose por la condición de enlace, ejemplo: Dime los apellidos de los empleados de los departamentos de ventas.

  Hasta ahora…

  SELECT APELLIDO
  FROM EMPLE, DEPART
  WHERE EMPLE.Dept_no = DEPART.Dept_no
  AND UPPER(DEPART.Dnombre)='VENTAS';

-Si se nos olvida poner el enlace saldrá un producto cartesiano y mezclará todos los empleados con todos los departamentos -> resultados falsos.

— LA CONSULTA EQUIVALENTE CON JOIN SERÍA—

  SELECT APELLIDO
  FROM EMPLE **JOIN** DEPART
  ON EMPLE.Dept_no = DEPART.Dept_no
  WHERE UPPER(DEPART.Dnombre) = ‘VENTAS’;

Te ahorras el where y pones la condición de enlace después del ON, entonces te ahorras la coma también; y el join lo tienes que poner al definir las tablas que vas a usar en el from.

CON LA NUEVA TABLA CLIENTE:

CONSULTA: *Quiero saber los nombres de los clientes atendidos por empleados de departamentos de 'VENTAS'.

-CON 2 ENLACES DE CONDICION (VERSIÓN ORIGINAL)

```
SELECT CLIENTE.NOMBRE, DEPART.DNOMBRE, EMPLE.APELLIDO
FROM CLIENTE, EMPLE, DEPART
WHERE DEPART.DEPT_NO = EMPLE.DEPT_NO
AND CLIENTE.EMP_NO = EMPLE.EMP_NO
AND UPPER(DNOMBRE) = 'VENTAS';
```

-CON JOIN

```
SELECT CLIENTE.NOMBRE, DEPART.DNOMBRE, EMPLE.APELLIDO
FROM CLIENTE
JOIN EMPLE
ON CLIENTE.EMP_NO = EMPLE.EMP_NO
JOIN DEPART
ON EMPLE.DEPT_NO = DEPART.DEPT_NO
WHERE UPPER(DEPART.DNOMBRE) = 'VENTAS';
```

EJERCICIOS EDITORIAL HECHOS CON JOIN:

TABLAS: LIBRO, EDITOR, IMPRENTA, AUTOR.

- Nombre de todos los autores que tenemos
- Nombre de los editores
- Nombre de los editores que trabajan con el libro "El Sol Brilla"
- Nombre de los libros que ha editado "Lalo Lález"
- Los libros que aún están en revisión
- Los nombres de imprenta que me han impreso algún libro
- Los nombres de imprenta que me han impreso algún libro del autor "Lolo Lólez"

- **Nombre de todos los autores que tenemos**

```
SELECT NOMBRE_AUTOR
FROM AUTOR;
```

- **Nombre de los editores**

```
SELECT NOMBRE_EDITOR
FROM EDITOR;
```

- **Nombre de los editores que trabajan con el libro "El Sol Brilla"**

CLÁSICO

```
SELECT TITULO_LIBRO, NOMBRE_EDITOR
FROM LIBRO, EDITOR
WHERE LIBRO.ID_EDITOR = EDITOR.ID_EDITOR
AND UPPER (LIBRO.TITULO_LIBRO) = ‘EL SOL BRILLA’;
```

CON JOIN:

```
SELECT TITULO_LIBRO, NOMBRE_EDITOR
FROM LIBRO JOIN EDITOR
ON LIBRO.ID_EDITOR = EDITOR.ID_EDITOR
AND UPPER (LIBRO.TITULO_LIBRO) = ‘EL SOL BRILLA’;
```

- **Nombre de los libros que ha editado "Lalo Lález"**

CLÁSICO

```
SELECT TITULO_LIBRO, NOMBRE_EDITOR
FROM LIBRO, EDITOR
WHERE LIBRO.ID_EDITOR = EDITOR.ID_EDITOR
AND UPPER(NOMBRE_EDITOR) = ‘LALO LÁLEZ’;
```

CON JOIN

```
SELECT TITULO_LIBRO, NOMBRE_EDITOR
FROM LIBRO JOIN EDITOR
ON LIBRO.ID_EDITOR = EDITOR.ID_EDITOR
AND UPPER(NOMBRE_EDITOR) = ‘LALO LÁLEZ’;
```

- **Los libros que aún están en revisión**

```
SELECT TITULO_LIBRO, ESTADO_LIBRO
FROM LIBRO
WHERE ESTADO_LIBRO = ‘En revisión’;
```

- **Los nombres de imprenta que me han impreso algún libro**

CLÁSICA

```
SELECT TITULO_LIBRO, NOMBRE_IMPRENTA, ESTADO_LIBRO
FROM IMPRENTA,LIBRO
WHERE LIBRO.ID_IMPRENTA = IMPRENTA.ID_IMPRENTA
AND
UPPER(ESTADO_LIBRO) = ‘PUBLICADO’;
```

CON JOIN

```
SELECT TITULO_LIBRO, NOMBRE_IMPRENTA, ESTADO_LIBRO
FROM IMPRENTA JOIN LIBRO
ON LIBRO.ID_IMPRENTA = IMPRENTA.ID_IMPRENTA
AND
UPPER(ESTADO_LIBRO) = ‘PUBLICADO’;
```

- **Los nombres de imprenta que me han impreso algún libro del autor "Lolo Lólez"**

CLÁSICO

```
SELECT NOMBRE_IMPRENTA
FROM IMPRENTA, LIBRO, AUTOR
WHERE IMPRENTA.ID_IMPRENTA = LIBRO.ID_IMPRENTA
AND LIBRO.ID_AUTOR = AUTOR.ID_AUTOR
AND AUTOR.NOMBRE_AUTOR = 'Lolo Lólez'
AND LIBRO.ESTADO_LIBRO = ‘Publicado’;
```

CON JOIN

```
SELECT NOMBRE_IMPRENTA
FROM LIBRO JOIN IMPRENTA
ON IMPRENTA.ID_IMPRENTA = LIBRO.ID_IMPRENTA
JOIN AUTOR
ON LIBRO.ID_AUTOR = AUTOR.ID_AUTOR
AND AUTOR.NOMBRE_AUTOR = ‘Lolo Lólez’
AND LIBRO.ESTADO_LIBRO = ‘Publicado’;
```

NOT IN:

PARA CUANDO UTILIZARÉ ENTONCES LAS SUBCONSULTAS ????
ÚNICAMENTE PARA AQUELLOS CASOS EN LOS QUE NO SE PUEDE
REALIZAR LAS COSAS DE OTRA MANERA. POR EJEMPLO:

LÓGICA INVERSA: NO PUEDO ENLAZAR TABLAS CON DATOS QUE NO EXISTEN
EJ: DAME LOS ARTÍCULOS NO VENDIDOS PARA PODERLOS PROMOCIONAR
AQUÍ SÍ QUE NECESITO SABER LOS ARTÍCULOS QUE ESTÁN EN LA TABLA
DE VENTAS PARA CON EL OPERADOR NOT IN LLEGAR A ENCONTRAR LOS
QUE ESTÁN EN LA TABLA DE ARTÍCULOS QUE NO SE CORRESPONDEN CON ESOS

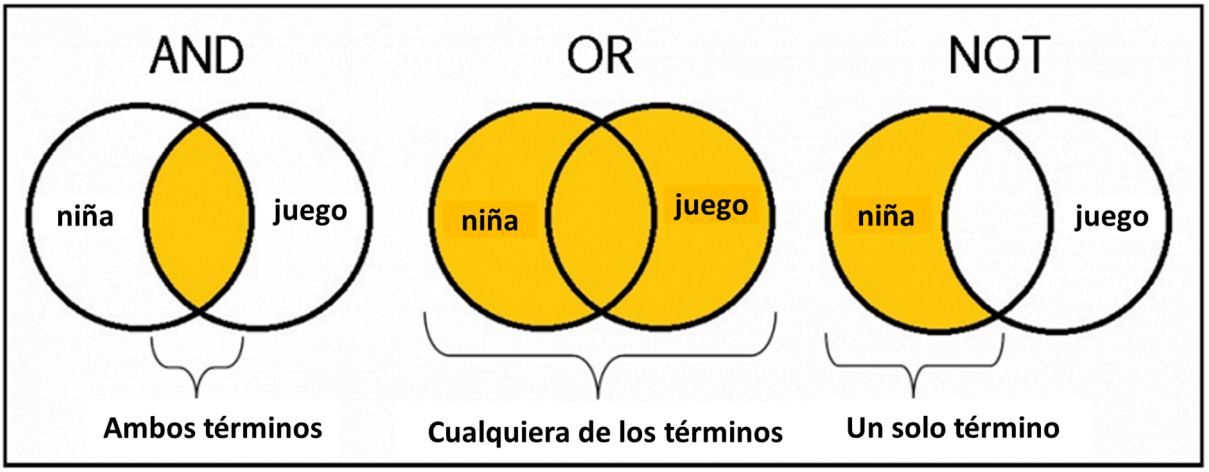
```
SELECT DISTINCT ARTÍCULO, CÓD FABRICANTE
FROM ARTICULOS
WHERE ARTICULO NOT IN (SELECT ARTICULO FROM VENTAS);
```

EN REALIDAD ES UNA RESTA DE CONJUNTOS !!!

ARTICULO COD_FABRICANTE
---------- ----------
Leche entera 30
Leche desnat. 30
Mejillones 10

SI LO QUIERO CON MÁXIMA SEGURIDAD TRABAJO CON FUNCIONES DE CADENA

```
SELECT DISTINCT ARTICULO, COD_FABRICANTE
FROM ARTICULOS
WHERE UPPER(ARTICULO) NOT IN (SELECT UPPER(ARTICULO) FROM VENTAS);
```



```
SELECT       # Seleccionar datos de la base de datos
FROM         # Especificar la tabla de la que seleccionar datos
WHERE        # Filtrar filas basadas en una condición
AS           # Renombrar una columna o tabla con un alias
JOIN         # Combinar filas de dos o más tablas
AND          # Combinar múltiples condiciones, todas deben ser verdaderas
OR           # Combinar múltiples condiciones, al menos una debe ser verdadera
LIMIT        # Restringir el número de filas devueltas
IN           # Especificar múltiples valores en una cláusula WHERE
CASE         # Crear lógica condicional dentro de una sentencia SQL
IS NULL      # Verificar valores vacíos
LIKE         # Buscar un patrón específico en una columna
COMMIT       # Guardar cambios realizados en la transacción
ROLLBACK     # Deshacer los cambios realizados en la transacción
ALTER TABLE  # Modificar la estructura de una tabla existente
UPDATE       # Modificar filas existentes en una tabla
CREATE       # Crear una nueva tabla, vista u otro objeto en la base de datos
DELETE       # Eliminar filas de una tabla
INSERT INTO  # Agregar nuevas filas a una tabla
DROP         # Eliminar una tabla u otro objeto en la base de datos
GROUP BY     # Agrupa filas que tienen los mismos valores en columnas
ORDER BY     # Ordenar el conjunto de resultados de una consulta
HAVING       # Filtrar grupos basados en una condición
COUNT        # Devuelve el número de filas que coinciden con un criterio
SUM          # Calcular el total de una columna numérica
AVG          # Calcular el valor promedio de una columna numérica
MIN          # Devolver el valor más pequeño de la columna seleccionada
MAX          # Devolver el valor más grande de la columna seleccionada
```

FUNCIONES

CONSULTA: **Quiero ver el número de empleados y su apellido para todos aquellos que tengan el mayor salario de la empresa**.

MAX

--MÁXIMO SALARIO DE LA EMPRESA—

```
SELECT MAX(SALARIO)
FROM EMPLE;
```

```
MAX(SALARIO)
------------
        4100
```

-> ESTO YA LO METES EN UNA SUBCONSULTA…

```
SELECT EMP_NO, APELLIDO, SALARIO
FROM EMPLE
WHERE SALARIO IN (SELECT MAX(SALARIO)
FROM EMPLE);
```

```
  EMP_NO APELLIDO      SALARIO
---------- ---------- ----------
      7839 REY              4100
```

CONSULTA: **Quiero saber cuántos empleados cobran el máximo salario de la empresa.**

COUNT

-> COUNT (CAMPO TABLA) “ALIAS SI SE DESEA”

```
SELECT COUNT(APELLIDO)  “NÚMERO EMPLEADOS CON EL SALARIO MAX”
FROM EMPLE
WHERE SALARIO IN (SELECT MAX(SALARIO)
FROM EMPLE);
```

```
NÚMERO EMPLEADOS CON EL SALARIO MAX
-----------------------------------
                                  1
```

SUBCONSULTAS CO - RRELACIONADAS

-Son consultas que contienen una subconsulta que hace referencia a la propia consulta exterior !!

Por ejemplo: Quiero saber el número y apellido de los empleados que cobran el máximo salario de su propio departamento.

V 1.0 (LA MALA) CON DOS O INCLUSO TRES CONSULTAS.

1) Saco el salario máximo de un departamento, por ejemplo el 20.

```
SELECT MAX(SALARIO) FROM EMPLE
WHERE DEPT_NO = 20;
```

```
MAX(SALARIO)
------------
        3000
```

2) Ahora quiero saber quién es el que tiene ese salario y…

```
SELECT EMP_NO, APELLIDO, DEPT_NO
FROM EMPLE
WHERE SALARIO = 3000
AND DEPT_NO = 20;
```

(Esta forma está muy mal)

El problema es que si nosotros actualizamos el salario de un empleado ya no funcionaría

```
Update EMPLE SET SALARIO = 3000
WHERE EMP_NO = 7934;
```

VERSIÓN 2.0 CON SUBCONSULTA -> CORRECTA

La subconsulta sería esta del paso 1

```
SELECT MAX(SALARIO) FROM EMPLE
WHERE DEPT_NO = 30
```

-> Esto te estaría devolviendo el salario máximo del departamento 20.

Y mi consulta “Grande” sería esta:

```
SELECT EMP_NO, APELLIDO, DEPT_NO
FROM EMPLE
WHERE SALARIO = (SELECT MAX(SALARIO) FROM EMPLE
WHERE DEPT_NO =20) AND DEPT_NO = 20;
```

Es decir tenemos un where en la subconsulta pero también fuera de la subconsulta, dos condiciones.

Lógica de porque ponemos la condición otra vez fuera de la consulta (es decir, la ponemos en la subconsulta interna y en la consulta externa):

Cuando usamos una subconsulta para obtener el salario máximo de un departamento específico, necesitamos asegurarnos de que los empleados seleccionados pertenecen a ese mismo departamento. Esto se logra repitiendo la condición en la consulta exterior.

Siempre que usemos una subconsulta que obtiene valores de un subconjunto de datos, debemos asegurarnos de que la consulta exterior filtra correctamente los registros que queremos mostrar porque por ejemplo si nosotros pusiéramos que queremos también que nos muestren los empleados del departamento 30 en la consulta exterior (la grande) nos estaría mostrando los empleados del departamento 30 con el salario máximo del departamento 20 ; no del departamento 30, deberíamos poner entonces el salario máximo del departamento 30 por ejemplo y fuera que queremos los del departamento 30.

CONSULTA: **Quiero que salgan los empleados que cobren el máximo salario de su departamento independientemente de cual sea el departamento, es decir, quiero que salgan los que cobran más de todos los departamentos pero comparándose con su propio departamento.**

VERSIÓN 3.1 (UNA SOLUCIÓN VÁLIDA)

```
SELECT EMPLE.EMP_NO, EMPLE.APELLIDO, EMPLE.DEPT_NO, EMPLE.SALARIO
FROM EMPLE, DEPART
WHERE EMPLE.SALARIO = (
SELECT MAX(SALARIO)
FROM EMPLE
WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO)
AND EMPLE.DEPT_NO = DEPART.DEPT_NO;
```

EMP_NO	APELLIDO	DEPT_NO	SALARIO
7839	REY	10	4100
7902	FERNANDEZ	20	3000
7788	GIL	20	3000
7698	NEGRO	30	3005

Esto funciona porque lo está haciendo con la condición de enlace dentro y fuera de la subconsulta.

Fíjate que lo que conseguimos es que el DEPT_NO de la subconsulta (QUE CORRESPONDE A EMPLE) siendo EMPLE la tabla de la subconsulta) sea igual al DEPT_NO de la consulta principal (DEPART.DEPT_NO) , es decir lo que está haciendo es que si el tío de dentro es del departamento 10, afuera (lo que se muestra) tiene que coger a un tío del departamento 10, es una correlación; la consulta exterior está relacionada con la consulta interior.

VERSIÓN 4.0 (AÚN MÁS MEJORADA)

Puedo conseguir lo mismo pero sin necesidad de utilizar dos tablas, por lo que mi consulta irá más rápida.

```
SELECT EMP_NO, APELLIDO, DEPT_NO
FROM EMPLE E
WHERE SALARIO = (SELECT MAX(SALARIO)
FROM EMPLE
WHERE EMPLE.DEPT_NO = E.DEPT_NO);
```

EMP_NO	APELLIDO	DEPT_NO
7698	NEGRO	30
7788	GIL	20
7839	REY	10
7902	FERNANDEZ	20

Ahora al quitar la tabla depart ya no puedo usar la condición de enlace ni el join; ya no tiene ningún tipo de sentido. Necesito Emple.DEPT_NO = DEPT_NO el segundo DEPT_NO haga referencia a la consulta exterior, pero no la puedo llamar emple !!! porque se confunde con la de la subconsulta, así que le ponemos un alias a la tabla emple para referenciar correctamente porque sino se iguala consigo mismo en la subconsulta, con la misma, es decir no sabe si esta comparando los datos con la tabla emple de la subconsulta o con la tabla emple de la consulta principal así que poco a poco lo vamos desglosando.

APUNTES 25/02/2025

FUNCIONES QUE DEVUELVEN VALORES NUMÉRICOS

TIPOS DE FUNCIONES (ESTÁN EN EL LIBRO págs. 75-84):

-ARITMÉTICAS

-SIMPLES (Funciones que devuelven valores simples).

```
SELECT CEIL(12.58) FROM DUAL;
```

REDONDEA 12.58 HACIA ARRIBA (CEIL=TECHO) DEVUELVE 13

```
SELECT FLOOR(12.58) FROM DUAL;
```

REDONDEA 12.58 HACIA ARRIBA DEVUELVE 12

```
SELECT ROUND(12.58) FROM DUAL;
```

REDONDEO NATURAL, DEVUELVE 13

```
SELECT ROUND(12.38) FROM DUAL;
```

REDONDEO NATURAL, DEVUELVE 12

-SELECT POWER(12,2) FROM DUAL;
ELEVA AL CUADRADO (12x12) DEVUELVE 144

-SELECT MOD(12,2) FROM DUAL;
MÓDULO (EL RESTO) DEVUELVE 0

-SELECT MOD(13,2) FROM DUAL;
MÓDULO (EL RESTO) DEVUELVE 1

-SELECT SIGN(0) FROM DUAL;
DEVUELVE EL SIGNO,
EN 0 ES 0 EN NEGATIVOS ES (-1)
Y EN POSITIVOS (1)

-SELECT SQRT(4) FROM DUAL;
RAÍZ CUADRADA DE UN NÚMERO (DEVUELVE 2)

-SELECT SQRT(13) FROM DUAL;
RAÍZ CUADRADA DE UN NÚMERO (DEVUELVE 3.61254242)

-SELECT ROUND(SQRT(13),2) FROM DUAL;
RAÍZ CUADRADA DE UN NÚMERO CON REDONDEO DE (2) (DEVUELVE 3.61)

-SELECT ROUND(SQRT(13),-1) FROM DUAL;
RAÍZ CUADRADA DE UN NÚMERO CON REDONDEO A UN NÚMERO MENOS (DEVUELVE 0)

SELECT ROUND(1222,-2) FROM DUAL;
//DEVUELVE 1200

-SELECT TRUNC(1299.256,2) FROM DUAL;
TRUNCA EL NÚMERO CON EL SEGUNDO NÚMERO
DEVUELVE 1200

SELECT APELLIDO, SALARIO, COMISION, SALARIO+NVL(COMISIÓN,0) "TOTAL"
FROM EMPLE
WHERE DEPT_NO = 30;
pág 76 DEL LIBRO
INTERCAMBIA UN VALOR "NULL" POR EL QUE TU QUIERAS PARA LA CONSULTA
EN ESTE CASO 0, PARA PODER SACAR EL TOTAL CORRECTO DE SALARIO + COMISIÓN

APUNTES 03/02/2025

FUNCIONES QUE TRABAJAN CON GRUPOS DE VALORES.

Ejemplo: Coger del salario y sacar el mínimo de un conjunto

MÍNIMO:

Ej: SELECT MIN (SALARIO) FROM EMPLE
WHERE DEPT_NO = 10;

MÁXIMO:

SELECT MAX (SALARIO) FROM EMPLEw
WHERE DEPT_NO = 10;

MEDIA (AVERAGE)

SELECT AVG(SALARIO)
FROM EMPLE
WHERE DEPT_NO = 10

SE PUEDEN COMBINAR TAMBIÉN -> FUNCIÓN SUM (SUMAR LOS VALORES)

SELECT SUM(SALARIO + NVL(COMISION, 0)) FROM EMPLE
WHERE DEPT_NO = 10

->Funcionaría de dentro hacia afuera, con el **NVL** -> función Null value lo que hace es que todos los valores que una lista de valores tiene a null los pone en algo, lo que definas a continuación
NVL(Tabla, valor a setear de los nulls)

COUNT

SELECT COUNT(APELLIDO) FROM EMPLE
WHERE DEPT_NO = 10;

->Cuenta todos los valores

TODOS LOS EMPLEADOS DE TODA LA EMPRESA QUE COBREN POR ENCIMA DE 2500

SELECT COUNT (*) FROM EMPLE
WHERE (SALARIO + NVL(COMISION,0))>2500;

DEL DEPARTAMENTO 30 QUIERO SABER LOS OFICIOS QUE HAY (CUÁNTOS HAY)

SELECT **COUNT(DISTINCT OFICIO)** AS OFICIOS_DEPARTAMENTO_30
FROM EMPLE
WHERE DEPT_NO = 30;

-> EL SELECT COUNT TE CUENTA TODOS AUNQUE ESTÉ REPETIDO, ENTONCES TIENES QUE
HACERLO CON EL SELECT DISTINCT PARA EVITAR QUE TE CUENTE VARIAS VECES LA MISMA COLUMNA / FILA.

TENGO UNA TABLA QUE SE LLAMA NOTAS_ALUMNOS (TABLA N-M)

NOTA MÁXIMA DE LA PRIMERA EVALUACIÓN

SELECT **MAX**(NOTA1)
FROM NOTAS_ALUMNOS;

PERO Y SI QUIERO LA MEJOR NOTA DE UN ALUMNO ENTONCES TENEMOS QUE TENER EN CUENTA VARIOS CAMPOS ES DECIR (NOTA 1, NOTA 2, NOTA 3).... ESO CON EL MAX / MIN NO SE PUEDE HACER YA QUE SOLO PODEMOS REFERENCIAR A UNA TABLA, ENTONCES VAMOS A HACERLO CON OTRA FUNCIÓN.

SELECT **GREATEST**(NOTA1, NOTA2, NOTA3) -> PUEDES PONER VARIOS CAMPOS.
FROM NOTAS_ALUMNOS
WHERE NOMBRE_ALUMNO = ‘Benito Martín, Luis’;

DIME LA **NOTA MINIMA** DE Benito Martín, Luis

SELECT **LEAST**(NOTA1, NOTA2, NOTA3) -> PUEDES PONER VARIOS CAMPOS.
FROM NOTAS_ALUMNOS
WHERE NOMBRE_ALUMNO = ‘Benito Martín, Luis’;

FUNCIONES DE CARÁCTERES

SELECT CHR(65) “LETRA”
FROM DUAL;

FUNCIÓN CONCAT -> CONCATENA CADENAS EJEMPLO:

SELECT CONCAT (‘HOLA’ , ‘QUE HACE’) “SALUDO”
FROM DUAL;

FUNCIÓN INITCAP -> TE PASA LA PRIMERA A MAYÚSCULAS:

SELECT INITCAP(LOWER(‘HOLA, QUE HACE’)) “SALUDO”

RELLENAR CON ALGO:

SELECT RPAD(APELLIDO,10,’*’),SALARIO,COMISIÓN

QUITAR ESPACIOS -> RTRIM, LTRIM (DERECHA, IZQUIERDA)

SELECT APELLIDO FROM EMPLE
WHERE UPPER(RTRIM(OFICIO))=’DIRECTOR’

ANIDAR LTRIM Y RTRIM -> SE PUEDE HACER EN EL MISMO

SELECT RTRIM(LTRIM(‘SALUDOS’,’AS’),’AS’)FROM DUAL;

Y SE PUEDEN ANIDAR AÚN MÁS COSAS

SELECT RPAD(RTRIM(LTRIM(‘SALUDOS’,’AS’),’AS’)’15’, ‘*-’)FROM DUAL;

TAMBIÉN PUEDE RECORTAR EL CARÁCTER QUE TÚ LE DIGAS.

EJ: SELECT LTRIM(‘SALUDOS’ , ‘S’) FROM DUAL;

NO SE REFIERE A LA CADENA SINO AL CONJUNTO, LO QUE SE CARGA ES EL CONJUNTO.

REPLACE:

SELECT REPLACE (‘SALUDO’, ‘UD’, ‘ud’) FROM DUAL;
(‘palabra’ , ‘lo que quitas’, ‘por lo que lo cambias’)

SUBSTRING -> DEVUELVE PARTE DE UNA CADENA

SELECT ‘HOLA, QUE HACHE’ “SALUDO” FROM DUAL;

IMAGINATE QUE QUIERO DEVOLVER LO QUE HAY DESPUÉS DE LA COMA, ES DECIR EL QUE HACHE

SELECT SUBSTR(‘HOLA, QUE HACHE’, 6) “SALUDOS” FROM DUAL;

Y LO PODEMOS MEZCLAR CON EL LTRIM PARA ELIMINAR ESPACIOS

SELECT LTRIM(SUBSTR(‘HOLA, QUE HACHE’, 6)) “SALUDOS” FROM DUAL;

TRANSLATE

->CAMBIA UNA SERIE DE CADENAS POR OTRA PERO NO LOS ELIMINA SINO LOS CAMBIA EN EL MOMENT

SELECT TRANSLATE(‘SALUDO POR LA MAÑANA’, ‘UAO’, ‘uao’) “SALUDO”
(la cadena, lo que buscas en ella, por lo que cambias lo que lo buscas)
Es un plan de traducción