

ORACLE

APUNTES



1 de enero de 2023

CAMPUSFP getafe

WALTER ISAMEL SAGASTEGUI LESCANO

INDEX

[1. OPERADORES COMPARACION 3](#_Toc149675491)

[1.1. OPERADORES COMPARACION 3](#_Toc149675492)

[1.2. EJEMPLOS 3](#_Toc149675493)

[2. OPERADORES LOGICOS 3](#_Toc149675494)

[2.1. OPERADORES LOGICOS 3](#_Toc149675496)

[2.2. EJEMPLOS 3](#_Toc149675497)

[3. CLAUSULA LIKE 4](#_Toc149675498)

[3.1. CLAUSULA LIKE 4](#_Toc149675500)

[3.2. EJEMPLOS 5](#_Toc149675501)

[4. OPERADOR CONCATENACION (||) 5](#_Toc149675502)

[4.1. EJEMPLOS 6](#_Toc149675504)

[5. EXPRESION REGULAR 6](#_Toc149675505)

[5.1. EXPRESION REGULAR 6](#_Toc149675507)

[5.2. EJEMPLOS 6](#_Toc149675508)

[6. FUNCIONES FECHA 6](#_Toc149675509)

[6.1. FUNCIONES FECHA 6](#_Toc149675511)

[6.2. EJEMPLOS 7](#_Toc149675512)

[7. OPERADORES ARITMETICOS 9](#_Toc149675513)

[7.1. EJEMPLOS 9](#_Toc149675515)

[8. FUNCIONES MATEMATICAS 9](#_Toc149675516)

[8.1. EJEMPLOS 9](#_Toc149675518)

[9. FUNCIONES DE CADENA 9](#_Toc149675519)

[9.1. FUNCIONES CADENA 9](#_Toc149675521)

[9.2. EJEMPLOS 9](#_Toc149675522)

[10. JOIN 10](#_Toc149675523)

[10.1. JOIN 10](#_Toc149675525)

[10.2. EJEMPLOS 11](#_Toc149675526)

[11. FUNCIONES DE AGREGACION 11](#_Toc149675527)

[11.1. FUNCIONES DE AGREGACION 12](#_Toc149675529)

[11.2. EJEMPLOS 12](#_Toc149675530)

[12. GROUP BY 12](#_Toc149675531)

[12.1. GROUP BY 12](#_Toc149675533)

[12.2. EJEMPLOS 13](#_Toc149675534)

[13. HEAVING 14](#_Toc149675535)

[13.1. HEAVING 14](#_Toc149675537)

[13.2. EJEMPLOS 14](#_Toc149675538)

[14. SUBCONSULTAS 15](#_Toc149675539)

[14.1. SUBCONSULTAS 15](#_Toc149675541)

[14.2. EJEMPLOS 15](#_Toc149675542)

[15. CRUD 17](#_Toc149675543)

[15.1. INSERT 17](#_Toc149675545)

[15.2. SELECT 17](#_Toc149675546)

[15.3. UPDATE 17](#_Toc149675547)

[15.4. DELETE 17](#_Toc149675548)

[16. PL/SQL 17](#_Toc149675549)

[16.1. PL/SQL 17](#_Toc149675551)

[16.2. EJEMPLOS 17](#_Toc149675552)

[17. EJERCICIOS DIVERSOS 18](#_Toc149675553)

[17.1. EJEMPLOS 18](#_Toc149675555)

# OPERADORES COMPARACION

## OPERADORES COMPARACION

= IGUAL

<>, != DISTINTO

> MAYOR

< MENOR

>= MAYOR O IGUAL

<= MENOR O IGUAL

## EJEMPLOS

1. **MOSTRAR TODOS LOS CLIENTES HOMBRES**

SELECT \* FROM CLIENTE WHERE SEXO = 'H';

1. **MOSTRAR TODOS LOS CLIENTES MUJERES**

SELECT \* FROM CLIENTE WHERE SEXO <> 'H';

1. **MOSTRAR TODOS LOS ARTICULOS CUYO PRECIO UNITARIO SEA MAYOR O IGUAL A 2**

SELECT \* FROM ARTICULO WHERE PRECIOUNITARIO >= 2;

# OPERADORES LOGICOS



## OPERADORES LOGICOS

AND CONJUCION

OR DISYUNCION

NOT NEGACION

BETWEEN

IN

NOT IN

IS NULL

IS NOT NULL

## EJEMPLOS

1. **MOSTRAR TODOS LOS CLIENTES HOMBRES Y MAYORES A 40 AÑOS**

SELECT \* FROM CLIENTE WHERE SEXO = 'H' AND EDAD > 40;

1. **MOSTRAR EL PRECIO DE LOS ARTICULOS LAPIZ O CUADERNO**

SELECT \* FROM ARTICULO WHERE DESCRIPCION = 'Lápiz' OR DESCRIPCION = 'Cuaderno';

SELECT \* FROM ARTICULO WHERE DESCRIPCION IN ('Lápiz','Cuaderno');

1. **MOSTRAR EL PRECIO DE LOS ARTICULOS QUE NO SEAN NI LAPIZ NI CUADERNO**

SELECT \* FROM ARTICULO WHERE NOT (DESCRIPCION = 'Lápiz' OR DESCRIPCION = 'Cuaderno');

SELECT \* FROM ARTICULO WHERE DESCRIPCION NOT IN ('Lápiz','Cuaderno');

1. **MOSTRAR TODOS LOS ARTICULOS CUYOS IDS ESTAN EN UNA LISTA**

SELECT \* FROM ARTICULO WHERE IDARTICULO IN ('IDA001', 'IDA002', 'IDA003');

1. **MOSTRAR TODOS LOS VENDEDORES CUYOS IDS NO ESTAN EN UNA LISTA**

SELECT \* FROM VENDEDOR WHERE IDVENDEDOR NOT IN ('IDV001', 'IDV002');

1. **MOSTRAR LOS CLIENTES QUE NO SEAN NI HOMBRES Y TAMPOCO MAYORES A 40 AÑOS**

SELECT \* FROM CLIENTE WHERE NOT (SEXO = 'H' OR EDAD > 40);

SELECT \* FROM CLIENTE WHERE NOT (SEXO = 'H') AND NOT (EDAD > 40);

1. **MOSTRAR LOS CLIENTES CUYA EDAD ESTA ENTRE 23 Y 45 (INCLUYA LOS EXTREMOS)**

SELECT \* FROM CLIENTE WHERE EDAD BETWEEN 23 AND 45;

1. **MOSTRAR LOS CLIENTES CUYA EDAD ESTA ENTRE 23 Y 45 (NO INCLUYA LOS EXTREMOS)**

SELECT \* FROM CLIENTE WHERE EDAD > 23 AND EDAD < 45;

1. **MOSTRAR LOS CLIENTES QUE NO TIENEN DIRECCION**

SELECT \* FROM CLIENTE WHERE DIRECCION IS NULL;

1. **MOSTRAR LOS CLIENTES QUE TIENEN DIRECCION**

SELECT \* FROM CLIENTE WHERE DIRECCION IS NOT NULL;

# CLAUSULA LIKE



## CLAUSULA LIKE

En Oracle, la cláusula **LIKE** se utiliza en una sentencia SQL para realizar búsquedas de patrones en datos de texto. Es útil cuando deseas buscar registros que coincidan con un cierto patrón de caracteres, en lugar de una coincidencia exacta.

La sintaxis básica de una consulta que utiliza **LIKE** es la siguiente:

SELECT columnas

FROM tabla

WHERE columna LIKE 'patrón';

Aquí hay algunas cosas clave a tener en cuenta:

* **columnas**: Representa las columnas que deseas seleccionar.
* **tabla**: Es el nombre de la tabla en la que deseas buscar.
* **columna**: Es la columna específica en la que deseas buscar el patrón.
* **patrón**: Es el patrón de caracteres que estás buscando. Puedes usar comodines para representar caracteres desconocidos o múltiples caracteres.

Los comodines más comunes son:

* **%**: Representa cero, uno o varios caracteres.
* **\_**: Representa un único carácter.

Por ejemplo, si deseas encontrar todos los nombres que comiencen con "J", puedes hacerlo así:

SELECT \*

FROM empleados

WHERE nombre LIKE 'J%';

Esto devolverá todos los registros de la tabla "empleados" donde el nombre comienza con la letra "J".

Otro ejemplo, si deseas encontrar todos los apellidos que terminen con "son", puedes hacerlo así:

SELECT \*

FROM empleados

WHERE apellido LIKE '%son';

Esto devolverá todos los registros de la tabla "empleados" donde el apellido termina con "son".

Recuerda que **LIKE** distingue entre mayúsculas y minúsculas en la mayoría de las bases de datos, a menos que se especifique lo contrario.

Para hacer que una consulta en Oracle no distinga entre mayúsculas y minúsculas, puedes usar la función **UPPER()** o **LOWER()** en combinación con la cláusula **LIKE**. Estas funciones convierten todos los caracteres a mayúsculas o minúsculas, respectivamente, lo que permite realizar la comparación de manera insensible a mayúsculas y minúsculas.

Por ejemplo, si quieres buscar todos los nombres que comienzan con "j" independientemente de si están en mayúsculas o minúsculas, puedes hacer lo siguiente:

SELECT \*

FROM empleados

WHERE UPPER(nombre) LIKE 'J%';

En este caso, la función **UPPER()** convierte todos los caracteres en la columna **nombre** a mayúsculas antes de realizar la comparación.

Si deseas que la comparación sea insensible a mayúsculas y minúsculas en ambos lados (es decir, tanto el patrón como los datos en la columna), debes aplicar **UPPER()** o **LOWER()** en ambas partes de la comparación:

SELECT \*

FROM empleados

WHERE UPPER(nombre) LIKE UPPER('j%');

De esta manera, ambas partes se comparan en mayúsculas, lo que garantiza que no se distinga entre mayúsculas y minúsculas.

Recuerda que al utilizar estas funciones, debes tener en cuenta que podrían afectar el rendimiento de la consulta, especialmente en grandes conjuntos de datos. Sin embargo, en la mayoría de los casos, esto no debería ser un problema significativo.

## EJEMPLOS

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE EMPIECEN CON 'C' EN MAYUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE 'C%';

Tener en cuenta que no mostraría los nombres que empiecen con 'c' en minúscula

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE EMPIECEN CON 'C' EN MAYUSCULA O MINUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE 'c%' OR NOMBRE LIKE 'C%';

O

SELECT \*

FROM CLIENTE

WHERE LOWER(NOMBRE) LIKE 'c%';

O

SELECT \*

FROM CLIENTE

WHERE UPPER(NOMBRE) LIKE 'C%';

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE CONTENGAN LA LETRA 'C' EN MINUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE '%c%';

Tener en cuenta que no mostraría los nombres con 'C' en mayuscula

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE CONTENGAN LA LETRA 'C' EN MINUSCULA O MAYUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE '%c%' OR NOMBRE LIKE '%C%';

1. **MOSTRAR LOS CLIENTE CUYO NOMBRE TERMINA CON 'O' EN MINUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE '%o';

1. **MOSTRAR LOS CLIENTS CUYO NOMBRE TENGA EXACTAMENTE 6 CARACTERES**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE '\_\_\_\_\_\_';

Se utilizan 6 guiones bajos para representar 6 caracteres, un guion por cada caracter

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE EMPIECEN CON 'C' O 'J' AMBOS EN MAYUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE 'C%' OR NOMBRE LIKE 'J%';

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE TENGA LA LETRA 'M' EN MINUSCULA EN LA TERCERA POSICION**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE '\_\_m%';

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE TERMINE CON 'OS' EN MINUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE '%os';

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE TENGA EXACTAMENTE 7 CARACTERES Y TERMINEN EN 'O' EN MINUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE '\_\_\_\_\_\_o';

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE EMPIECEN CON UNA VOCAL EN MAYUSCULA**

SELECT \* FROM CLIENTE WHERE NOMBRE LIKE 'A%' OR NOMBRE LIKE 'E%' OR NOMBRE LIKE 'I%' OR NOMBRE LIKE 'O%' OR NOMBRE LIKE 'U%';

1. **MOSTRAR LOS CLIENTES CUYO NOMBRE EMPIECE CON UNA CONSONANTE EN MAYUSCULA**

SELECT \* FROM CLIENTE WHERE NOT (NOMBRE LIKE 'A%' OR NOMBRE LIKE 'E%' OR NOMBRE LIKE 'I%' OR NOMBRE LIKE 'O%' OR NOMBRE LIKE 'U%');

# OPERADOR CONCATENACION (||)



## EJEMPLOS

1. **CONCATENAR DOS COLUMNAS**

SELECT NOMBRE || ' ' || APELLIDO AS NOMBRE\_COMPLETO FROM CLIENTE;

SELECT NOMBRE || ' - ' || TELEFONO AS DETALLE\_CONTACTO FROM CLIENTE;

1. **CONCATENAR TRES COLUMNAS**

SELECT NOMBRE || ' ' || APELLIDO || ', Tel: ' || TELEFONO AS INFO\_CONTACTO FROM CLIENTE;

1. **CONCATENAR UN TEXTO FIJO A UNA COLUMNA**

SELECT 'EL CLIENTE ' || NOMBRE || ' TIENE ' || EDAD || ' AÑOS.' AS INFO\_CLIENTE FROM CLIENTE;

1. **CONCATENAR UN TEXTO FIJO A DOS COLUMNAS INDEPENDIENTES CADA UNA**

SELECT 'ID: ' || IDCLIENTE || ', NOMBRE: ' || NOMBRE AS INFO\_CLIENTE FROM CLIENTE;

SELECT 'NOMBRE: ' || NOMBRE || ', EDAD: ' || EDAD || ' AÑOS.' AS DETALLE\_CLIENTE FROM CLIENTE;

1. **CONCATENAR TEXTO CON VALORES DE COLUMNAS**

SELECT 'EL CLIENTE ' || NOMBRE || ' TIENE UNA DIRECCIÓN EN ' || DIRECCION AS INFO\_DIRECCION

FROM CLIENTE;

1. **CONCATENAR TEXTO, COLUMNAS Y FUNCIONES**

SELECT 'EL CLIENTE ' || NOMBRE || ' TIENE ' || TO\_CHAR(EDAD) || ' AÑOS.' AS INFO\_CLIENTE FROM CLIENTE;

# EXPRESION REGULAR



## EXPRESION REGULAR

En una expresión regular, el símbolo **^** se utiliza para denotar el comienzo de una cadena. Esto significa que la expresión regular buscará un patrón que esté al principio de la cadena y el simbolo $ se utiliza para denotar el final de una cadena.

La función **REGEXP\_LIKE** devuelve un valor booleano (**TRUE** o **FALSE**) y no se puede usar directamente en la cláusula **SELECT** junto con el nombre.

## EJEMPLOS

1. **MOSTRAR TODOS LOS CLIENTES CUYO NOMBRE EMPIEZAN CON LA LETRA 'C'(MAYUSCULA)**

SELECT \* FROM CLIENTE WHERE REGEXP\_LIKE(NOMBRE, '^C');

1. **MOSTRAR LOS CLIENTES CUYO NUMERO DE TELEFONO TIENE 8 DIGITOS**

SELECT \* FROM CLIENTE WHERE REGEXP\_LIKE(TELEFONO, '^[0-9]{8}$');

1. **MOSTRAR LOS ARTICULOS QUE SON LAPIZ O CUADERNO**

SELECT \* FROM ARTICULO WHERE REGEXP\_LIKE(DESCRIPCION, 'Lápiz|Cuaderno');

1. **MOSTRAR LA DIRECCION DE TODOS LOS CLIENTES DONDE EN LUGAR DE MOSTRAR AV. HAY QUE MOSTRAR AVENIDA**

SELECT NOMBRE, REGEXP\_REPLACE(DIRECCION, 'Av.', 'Avenida') FROM CLIENTE;

1. **MOSTRAR SOLO LOS NUMEROS DE LA DIRECCION DE TODOS LOS CLIENTES**

SELECT NOMBRE, REGEXP\_SUBSTR(DIRECCION, '\d+') FROM CLIENTE;

SELECT NOMBRE, REGEXP\_SUBSTR(DIRECCION, '[0-9]+') FROM CLIENTE;

1. **MOSTRAR LOS CLIENTES CUYO MAIL CUMPLAN CON UNA EXPRESION REGULAR. MINIMO 3 CARACTERES LA EXTENSION**

SELECT NOMBRE, EMAIL FROM CLIENTE

WHERE REGEXP\_LIKE(EMAIL, '^[A-Za-z0-9]+@[A-Za-z0-9]+\.[A-Za-z]{3}$');

# FUNCIONES FECHA



## FUNCIONES FECHA

TO\_CHAR(FECHAFACTURA, 'YYYY') = 1972

EXTRACT(YEAR FROM FECHAFACTURA) = 1972

TO\_CHAR(FECHAFACTURA, 'MM') = 1

EXTRACT(MONTH FROM FECHAFACTURA) = 1

TO\_CHAR(FECHAFACTURA, 'DD') = 1

EXTRACT(DAY FROM FECHAFACTURA) = 1

FECHAFACTURA > TO\_DATE('1972-01-01', 'YYYY-MM-DD')

EXTRACT(YEAR FROM FECHAFACTURA) BETWEEN 1970 AND 1973

## EJEMPLOS

1. **MOSTRAR LAS FACTURAS DISTINTAS AL AÑO 1972**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE NOT (EXTRACT(YEAR FROM FECHAFACTURA) = 1972);

1. **MOSTRAR LAS FACTURAS DISTINTAS AL AÑO 1972**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE EXTRACT(YEAR FROM FECHAFACTURA) <> 1972; -- También funciona '1972'

1. **MOSTRAR LAS FACTURAS ENTRE UN RANGO DE FECHAS: 1 DE ENERO AL 31 DE DICIEMBRE DE 1972**

Esta solución betwen incluye los extremos:

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE FECHAFACTURA BETWEEN TO\_DATE('1972-01-01', 'YYYY-MM-DD') AND TO\_DATE('1972-12-31', 'YYYY-MM-DD');

Esta otra solución no incluye los extremos:

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE FECHAFACTURA > TO\_DATE('1972-01-01', 'YYYY-MM-DD') AND FECHAFACTURA < TO\_DATE('1972-12-31', 'YYYY-MM-DD');

No incluye los extremos

1. **MOSTRAR LAS FACTURAS ENTRE EL AÑO 1970 Y 1973**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE EXTRACT(YEAR FROM FECHAFACTURA) BETWEEN 1970 AND 1973; -- Entre 01/01/1970 hasta 31/12/1973 incluye los extremos

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE FECHAFACTURA BETWEEN TO\_DATE('1972-01-01', 'YYYY-MM-DD') AND TO\_DATE('1973-12-31', 'YYYY-MM-DD'); -- Entre 01/01/1970 hasta 31/12/1973 incluye los extremos

1. **MOSTRAR LAS FACTURAS DE FECHA: 1972-01-01, 1973-01-01, 1975-01-01**

SELECT \*

FROM FACTURA

WHERE FECHAFACTURA IN (TO\_DATE('1972-01-01', 'YYYY-MM-DD'),

TO\_DATE('1973-01-01', 'YYYY-MM-DD'),

TO\_DATE('1975-01-01', 'YYYY-MM-DD'));

1. **MOSTRAR TODAS LAS FACTURAS CON SU FECHA EMITIDA EN FORMATO DD-MM-YYYY Y DD/MM/YYYY**

SELECT IDFACTURA, TO\_CHAR(FECHAFACTURA, 'DD-MM-YYYY') AS FECHA\_FORMATO

FROM FACTURA;

SELECT IDFACTURA, TO\_CHAR(FECHAFACTURA, 'DD/MM/YYYY') AS FECHA\_FORMATO

FROM FACTURA;

1. **MOSTRAR LAS FACTURAS DEL AÑO 1972. IDFACTURA**

SELECT IDFACTURA

FROM FACTURA

WHERE TO\_CHAR(FECHAFACTURA, 'YYYY') = 1972; -- También funciona '1972'

O

SELECT IDFACTURA

FROM FACTURA

WHERE EXTRACT(YEAR FROM FECHAFACTURA) = 1972; -- También funciona '1972'

1. **MOSTRAR LAS FACTURAS DEL MES DE ENERO**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE TO\_CHAR(FECHAFACTURA, 'MM') = 1; -- No funciona '1';

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE EXTRACT(MONTH FROM FECHAFACTURA) = 1; -- También funciona '1'

1. **MOSTRAR LAS FACTURAS DEL DIA 1**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE TO\_CHAR(FECHAFACTURA, 'DD') = 1; -- No funciona '1';

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE EXTRACT(DAY FROM FECHAFACTURA) = 1; -- También funciona '1'

1. **MOSTRAR LAS FACTURAS ENTRE UN RANGO(1 DE ENERO AL 31 DE DICIEMBRE DE 1972) DE FECHAS**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE FECHAFACTURA BETWEEN TO\_DATE('1972-01-01', 'YYYY-MM-DD') AND TO\_DATE('1972-12-31', 'YYYY-MM-DD'); -- BETWEEN incluye los extremos

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE FECHAFACTURA > TO\_DATE('1972-01-01', 'YYYY-MM-DD') AND FECHAFACTURA < TO\_DATE('1972-12-31', 'YYYY-MM-DD'); -- No incluye los extremos incluye los extremos

1. **MOSTRAR LA FECHA ALCTUAL DEL SISTEMA**

SELECT SYSDATE FROM DUAL; -- Dual es una tabla de un solo registro entonces mostrar la fecha actual una sola vez

1. **MOSTRAR TODAS LAS FACTURAS 2 MESES DESPUES DEL PRIMERO DE ENERO DE 1972**

SELECT IDFACTURA, FECHAFACTURA FROM FACTURA WHERE FECHAFACTURA > ADD\_MONTHS('01/01/1972', 2);

1. **MOSTRAR LAS FACTURAS ENTRE EL AÑO 1970 Y 1973**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE EXTRACT(YEAR FROM FECHAFACTURA) BETWEEN 1970 AND 1973; -- Entre 01/01/1970 hasta 31/12/1973 incluye los extremos

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE FECHAFACTURA BETWEEN TO\_DATE('1972-01-01', 'YYYY-MM-DD') AND TO\_DATE('1973-12-31', 'YYYY-MM-DD'); -- Entre 01/01/1970 hasta 31/12/1973 incluye los extremos

1. **MOSTRAR LAS FACTURAS DEL DIA 6 Y AÑO 1975**

SELECT \*

FROM FACTURA

WHERE EXTRACT(YEAR FROM FECHAFACTURA) = 1975 AND EXTRACT(DAY FROM FECHAFACTURA) = 6; -- AND EXTRACT(MONTH FROM FECHAFACTURA) = 1

1. **MOSTRAR LAS FACTURAS DE CUALQUIER AÑO DISTINTO AL AÑO 1972**

SELECT IDFACTURA, FECHAFACTURA

FROM FACTURA

WHERE EXTRACT(YEAR FROM FECHAFACTURA) != 1972;

1. **MOSTRAR LAS FACTURAS DE LA CONSULTA ANTERIOR DONDE LA FECHAFACTURA DE SALIDA TENGA EL FORMATO DD/MM/YYYY**

SELECT IDFACTURA, TO\_CHAR(FECHAFACTURA, 'DD/MM/YYYY') AS FECHAFACTURA

FROM FACTURA

WHERE EXTRACT(YEAR FROM FECHAFACTURA) != 1972;

1. **MOSTRAR LAS FACTURAS DE FECHA: 1972-01-01, 1973-01-01, 1975-01-01**

SELECT \*

FROM FACTURA

WHERE FECHAFACTURA IN (TO\_DATE('1972-01-01', 'YYYY-MM-DD'),

TO\_DATE('1973-01-01', 'YYYY-MM-DD'),

TO\_DATE('1975-01-01', 'YYYY-MM-DD'));

# OPERADORES ARITMETICOS



## EJEMPLOS

SELECT 5 + 2 AS SUMA FROM DUAL; -- 7

SELECT 5 - 2 AS RESTA FROM DUAL; -- 3

SELECT 5 \* 2 AS MULTIPLICACION FROM DUAL; -- 10

SELECT 5 / 2 AS DIVISION FROM DUAL; -- 2.5

SELECT MOD(5, 2) AS RESTO FROM DUAL; -- 1

SELECT MOD((10 \* 2 + 5), 3) FROM DUAL; -- 1

# FUNCIONES MATEMATICAS



## EJEMPLOS

SELECT ABS(-10) FROM DUAL; -- 10

SELECT CEIL(3.14) FROM DUAL; -- 4

SELECT FLOOR(3.14) FROM DUAL; -- 3

SELECT ROUND(3.145, 2) FROM DUAL; -- 3.15

SELECT ROUND(3.145, 1) FROM DUAL; -- 3.1

SELECT ROUND(3.45, 0) FROM DUAL; -- 3

SELECT ROUND(3.56, 0) FROM DUAL; -- 4

SELECT ROUND(3.56, 1) FROM DUAL; -- 3.6

SELECT TRUNC(3.145, 2) FROM DUAL; -- 3.14

SELECT TRUNC(3.155, 1) FROM DUAL; -- 3.1

SELECT TRUNC(3.155, 0) FROM DUAL; -- 3

SELECT SQRT(25) FROM DUAL; -- 5

SELECT SQRT(2) FROM DUAL; -- 1,41421356

SELECT SQRT(5) FROM DUAL; -- 2,23606798

SELECT POWER(2, 3) FROM DUAL; -- 8

SELECT POWER(2, 0.5) FROM DUAL; -- 1,41421356

SELECT POWER(1/4, 0.5) FROM DUAL; -- 0,5

SELECT EXP(1) FROM DUAL; -- 2.71828...

SELECT LOG(10) FROM dual; -- 2.30258...

SELECT LOG10(100) FROM dual; -- 2

# FUNCIONES DE CADENA



## FUNCIONES CADENA

UPPER, LOWER, INITCAP, LENGTH, LENGTHB, SUBSTR, INSTR, CONCAT, TRIM, LTRIM, RTRIM, REMPLACE

## EJEMPLOS

1. **MOSTRAR EL NOMBRE DE TODOS LOS CLIENTES EN MAYUSCULA**

SELECT UPPER(NOMBRE) FROM CLIENTE;

1. **MOSTRAR EL NOMBRE DE TODOS LOS CLIENTES EN MINUSCULAS**

SELECT LOWER(NOMBRE) FROM CLIENTE;

1. **MOSTRAR LA PRIMERA LETRA DE CADA CORREO DE TODOS LOS CLIENTES EN MAYUSCULA**

SELECT INITCAP(EMAIL) FROM CLIENTE;

1. **MOSTRAR LA LONGITUD DEL NOMBRE DE TODOS LOS CLIENTES EN CARACTERES Y BYTES**

SELECT LENGTH(NOMBRE) AS NUMERO\_CARACTERES, LENGTHB(NOMBRE) AS NUMERO\_BYTES FROM CLIENTE;

1. **MOSTRAR EL TEXTO DEL CORREO ANTES DE LA ARROBA DE TODOS LOS CLIENTES**

SELECT SUBSTR(EMAIL,1,LENGTH(NOMBRE)) FROM CLIENTE; -- CADENA, INICIO(PRIMERO POSICION ES 1), LONGITUD

1. **MOSTRAR LA POSICION QUE OCUPA LA PRIMERA LETRA A EN EL NOMBRE DE TODOS LOS CLIENTES**

SELECT INSTR(NOMBRE, 'a') FROM CLIENTE;

1. **MOSTRAR LA CONCATENACION DE LAS COLUMNAS NOMBRE Y APELLIDO DE TODOS LOS CLIENTES**

SELECT CONCAT(NOMBRE,APELLIDO) FROM CLIENTE;

1. **ELIMINAR ESPACIOS EN BLANCO DE UNA CADENA**

SELECT TRIM(' hola ') FROM DUAL; -- Devuelve 'hola'

SELECT LTRIM(' hola ') FROM DUAL; -- Devuelve 'hola '

SELECT RTRIM(' hola ') FROM DUAL; -- Devuelve ' hola'

1. **REMPLAZAR UNA SUBADENA POR OTRA**

SELECT REPLACE('hola mundo', 'mundo', 'amigo') FROM DUAL; -- Devuelve 'hola amigo'

# JOIN



## JOIN

En Oracle (y en la mayoría de las bases de datos relacionales), el **JOIN** es una operación que se utiliza para combinar filas de dos o más tablas basándose en una condición de relación entre ellas. Esto permite extraer información relacionada de diferentes tablas en una sola consulta.

Existen varios tipos de **JOIN** en Oracle:

1. **INNER JOIN**:

* El **INNER JOIN** devuelve solo las filas que tienen correspondencias en ambas tablas.
* La sintaxis básica es:

SELECT columnas

FROM tabla1

INNER JOIN tabla2 ON tabla1.columna = tabla2.columna;

Por ejemplo, si queremos combinar una tabla de empleados con una tabla de departamentos para obtener la información del empleado y su departamento correspondiente, podemos hacerlo así:

SELECT empleados.nombre, departamentos.nombre\_departamento

FROM empleados

INNER JOIN departamentos ON empleados.departamento\_id = departamentos.departamento\_id;

El término "JOIN" por sí solo a menudo se refiere a un "INNER JOIN". En muchos sistemas de gestión de bases de datos, si no se especifica el tipo de join, se asume un "INNER JOIN" de manera implícita.

Por lo tanto, si se habla simplemente de "JOIN" en el contexto de Oracle o en la mayoría de las bases de datos, se está refiriendo a un "INNER JOIN". Si se desea un tipo de join diferente, se debe especificar explícitamente, como en el caso de "LEFT JOIN", "RIGHT JOIN", "FULL JOIN", etc.

**2. LEFT JOIN (o LEFT OUTER JOIN)**:

* El **LEFT JOIN** devuelve todas las filas de la tabla de la izquierda y las filas coincidentes de la tabla de la derecha. Si no hay coincidencias, se devuelve NULL para las columnas de la tabla de la derecha.
* La sintaxis es similar a la del **INNER JOIN**:

SELECT columnas

FROM tabla1

LEFT JOIN tabla2 ON tabla1.columna = tabla2.columna;

Por ejemplo, si queremos listar todos los empleados y sus departamentos, incluso si algunos empleados no tienen un departamento asignado, podemos usar un **LEFT JOIN**:

SELECT empleados.nombre, departamentos.nombre\_departamento

FROM empleados

LEFT JOIN departamentos ON empleados.departamento\_id = departamentos.departamento\_id;

**3. RIGHT JOIN (o RIGHT OUTER JOIN)**:

* El **RIGHT JOIN** es similar al **LEFT JOIN**, pero devuelve todas las filas de la tabla de la derecha y las filas coincidentes de la tabla de la izquierda. Si no hay coincidencias, se devuelve NULL para las columnas de la tabla de la izquierda.
* La sintaxis es similar a la del **INNER JOIN** y **LEFT JOIN**, pero intercambia el orden de las tablas.

**4. FULL JOIN (o FULL OUTER JOIN)**:

* El **FULL JOIN** devuelve filas cuando hay una coincidencia en una de las tablas. Es el resultado de combinar **LEFT JOIN** y **RIGHT JOIN**.
* No es soportado directamente en Oracle. Sin embargo, se puede simular usando **UNION** de **LEFT JOIN** y **RIGHT JOIN**.

**5. CROSS JOIN**:

* El **CROSS JOIN** combina cada fila de la primera tabla con cada fila de la segunda tabla, creando un producto cartesiano.
* La sintaxis es simplemente:

SELECT columnas

FROM tabla1

CROSS JOIN tabla2;

**6. SELF JOIN**:

* Un **SELF JOIN** es cuando una tabla se combina consigo misma. Puede ser útil cuando tienes información relacionada dentro de la misma tabla.

En Oracle, no existe una cláusula de **SELF JOIN** como tal. Para unir una tabla consigo misma en Oracle, debes utilizar alias para referenciar la misma tabla más de una vez en una consulta.

Por ejemplo, si tienes una tabla **Empleados** y quieres encontrar los supervisores de los empleados, puedes hacerlo de la siguiente manera:

SELECT e1.Nombre AS Empleado, e2.Nombre AS Supervisor

FROM Empleados e1

LEFT JOIN Empleados e2 ON e1.IDSupervisor = e2.ID;

En esta consulta, **e1** y **e2** son alias de la tabla **Empleados**. Estamos haciendo un **LEFT JOIN** entre la tabla **Empleados** (aliada como **e1**) y la misma tabla **Empleados** (aliada como **e2**) utilizando la condición **e1.IDSupervisor = e2.ID**. Esto significa que estamos comparando el **IDSupervisor** de un empleado (**e1.IDSupervisor**) con el **ID** de otro empleado (**e2.ID**).

Estos son los tipos básicos de **JOIN** en Oracle.

## EJEMPLOS

1. **MOSTRAR EL NOMBRE DEL CLIENTE POR CADA FACTURA**

SELECT IDFACTURA, FECHAFACTURA, C.IDCLIENTE, C.NOMBRE

FROM FACTURA F

WHERE F.IDCLIENTE = C.IDCLIENTE;

O

SELECT IDFACTURA, FECHAFACTURA, C.IDCLIENTE, C.NOMBRE

FROM FACTURA F

JOIN CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE;

1. **MOSTRAR EL NOMBRE DE LOS ARTICULOS POR CADA FACTURA**

SELECT IDFACTURA, A.IDARTICULO, A.DESCRIPCION

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO;

O

SELECT IDFACTURA, A.IDARTICULO, A.DESCRIPCION

FROM ARTICULO A

JOIN ARTICULOS\_X\_FACTURA AF ON AF.IDARTICULO = A.IDARTICULO;

1. **MOSTRAR TODAS LAS FACTURAS QUE TENGAN EL ARTICULO BORRADOR**

SELECT AF.IDFACTURA, A.DESCRIPCION

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO AND A.DESCRIPCION = 'Borrador'

ORDER BY AF.IDFACTURA;

O

SELECT AF.IDFACTURA, A.DESCRIPCION

FROM ARTICULOS\_X\_FACTURA AF

JOIN ARTICULO A ON AF.IDARTICULO = A.IDARTICULO

WHERE A.DESCRIPCION = 'Borrador'

ORDER BY AF.IDFACTURA;

PUEDE SER RESUELTO TAMBIEN CON UNA SUBCONSULTA

# FUNCIONES DE AGREGACION



## FUNCIONES DE AGREGACION

Las funciones de agregación son operaciones matemáticas que se aplican a un conjunto de valores y devuelven un único resultado resumido. Estas funciones permiten realizar cálculos sobre grupos de filas de una tabla en SQL u otros lenguajes de consulta de bases de datos. Las funciones de agregación más comunes incluyen:

1. **SUM**: Calcula la suma de los valores en una columna.

Ejemplo: **SELECT SUM(precio) FROM productos;**

1. **COUNT**: Cuenta el número de filas en un grupo.

Ejemplo: **SELECT COUNT(\*) FROM clientes;**

1. **AVG**: Calcula el promedio de los valores en una columna numérica.

Ejemplo: **SELECT AVG(edad) FROM empleados;**

1. **MAX**: Encuentra el valor máximo en una columna.

Ejemplo: **SELECT MAX(precio) FROM productos;**

1. **MIN**: Encuentra el valor mínimo en una columna.

Ejemplo: **SELECT MIN(edad) FROM empleados;**

1. **COUNT DISTINCT**: Cuenta el número de valores distintos en una columna.

Ejemplo: **SELECT COUNT(DISTINCT categoria) FROM productos;**

Estas funciones son especialmente útiles cuando se utilizan junto con la cláusula **GROUP BY**. Al aplicar una función de agregación a un grupo de filas, puedes obtener un resumen de los datos en lugar de resultados individuales.

Por ejemplo, si quieres saber el total de ventas por categoría de producto, podrías usar la función **SUM** junto con **GROUP BY**:

SELECT categoria, SUM(ventas)

FROM productos

GROUP BY categoria;

Esto te daría el total de ventas para cada categoría de productos.

Recuerda que las funciones de agregación operan en conjuntos de datos, por lo que a menudo se utilizan en consultas que incluyen la cláusula **GROUP BY**, aunque también pueden usarse sin ella para obtener un resultado agregado de toda la tabla.

## EJEMPLOS

1. **MOSTRAR EL PRECIO PROMEDIO DE LOS ARTICULOS**

SELECT AVG(PRECIOUNITARIO) AS PROMEDIO\_PRECIO FROM ARTICULO;

1. **MOSTRAR LA SUMA DE LOS PRECIOS DE LOS ARTICULOS**

SELECT SUM(PRECIOUNITARIO) AS SUMA\_PRECIO FROM ARTICULO;

1. **MOSTRAR EL TOTAL DE UNIDADES VENDIDAS TOMANDO TODOS LOS ARTICULOS VENDIDOS**

SELECT SUM(CANTIDAD) AS TOTAL\_UNIDADES FROM ARTICULOS\_X\_FACTURA;

1. **MOSTRAR EL NUMERO TOTAL DE CLIENTES**

SELECT COUNT(\*) FROM CLIENTE;

# GROUP BY



## GROUP BY

En Oracle, la cláusula **GROUP BY** es una instrucción que se utiliza junto con las funciones de agregación (como SUM, COUNT, AVG, MAX, MIN) para agrupar filas que tienen valores iguales en una o más columnas y aplicar funciones de agregación a esos grupos.

En otras palabras, **GROUP BY** se utiliza para dividir el conjunto de resultados en grupos basados en los valores de una o más columnas, y luego aplicar una función de agregación a cada uno de esos grupos. Esto es útil cuando quieres realizar cálculos sobre subconjuntos de datos específicos en una tabla.

Aquí hay un ejemplo básico de cómo se usa **GROUP BY** en Oracle:

Supongamos que tienes una tabla llamada **ventas** con las siguientes columnas: **producto**, **cantidad** y **precio\_unitario**.

SELECT producto, SUM(cantidad)

FROM ventas

GROUP BY producto;

En este ejemplo, estamos seleccionando el nombre del producto y la suma de la cantidad vendida para cada producto. La cláusula **GROUP BY producto** agrupa las filas por el nombre del producto, lo que significa que obtendremos una fila para cada producto distinto en la tabla. Luego, la función **SUM(cantidad)** se aplica a cada grupo, lo que nos da la suma de la cantidad vendida para cada producto.

Es importante tener en cuenta que cuando se utiliza **GROUP BY**, solo puedes seleccionar columnas que están incluidas en la cláusula **GROUP BY** o en funciones de agregación. Esto significa que si no se incluye una columna en la cláusula **GROUP BY** o en una función de agregación, no puedes seleccionarla en la consulta.

## EJEMPLOS

1. **MOSTRAR LA CANTIDAD DE ARTICULOS POR CADA FACTURA**

SELECT IDFACTURA, COUNT(IDARTICULO)

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDFACTURA;

1. **MOSTRAR LA CANTIDAD DE FACTURAS POR CLIENTE(IDCLIENTE)**

SELECT IDCLIENTE, COUNT(IDFACTURA)

FROM FACTURA

GROUP BY IDCLIENTE;

1. **MOSTRAR EL TOTAL DE UNIDADES COMPRADAS TOMANDO TODOS LOS ARTICULOS POR CADA FACTURA.**

SELECT IDFACTURA, SUM(CANTIDAD) AS TOTAL\_UNIDADES

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDFACTURA;

1. **MOSTRAR LA CANTIDAD DE FACTURAS POR CLIENTE(NOMBRE)**

SELECT C.NOMBRE, COUNT(IDFACTURA)

FROM FACTURA F, CLIENTE C

WHERE F.IDCLIENTE = C.IDCLIENTE

GROUP BY C.NOMBRE;

O

SELECT C.NOMBRE, COUNT(IDFACTURA)

FROM FACTURA F

JOIN CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE

GROUP BY C.NOMBRE;

1. **MOSTRAR LAS VENTAS TOTALES POR CADA ARTICULO**

SELECT AF.IDARTICULO, SUM(A.PRECIOUNITARIO \* AF.CANTIDAD) AS VENTA\_TOTAL

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO

GROUP BY AF.IDARTICULO

ORDER BY VENTA\_TOTAL;

1. **MOSTRAR EL TOTAL DE UNIDADES COMPRADAS POR CADA ARTICULO**

SELECT IDFACTURA, SUM(CANTIDAD) AS TOTAL\_UNIDADES

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDFACTURA;

1. **MOSTRAR EL TOTAL DE UNIDADES VENDIDAS DE CADA ARTICULO**

SELECT IDARTICULO, SUM(CANTIDAD)

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO;

...

SELECT A.DESCRIPCION, SUM(CANTIDAD)

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO

GROUP BY A.DESCRIPCION;

1. **MOSTRAR TODOS LOS GRUPOS DISTINTOS DE ARTICULO(IDARTICULO)**

SELECT IDARTICULO

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO;

1. **MOSTRAR LA CANTIDAD DE FACTURAS POR CLIENTE**

SELECT IDCLIENTE, COUNT(IDFACTURA) AS CANTIDAD\_FACTURAS

FROM FACTURA

GROUP BY IDCLIENTE

1. **MOSTRAR EL TOTAL EN VENTA POR CADA FACTURA**

SELECT AF.IDFACTURA, SUM(A.PRECIOUNITARIO \* AF.CANTIDAD)

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO

GROUP BY AF.IDFACTURA;

1. **MOSTRAR LAS FACTURAS POR CADA VENDEDOR**

SELECT IDVENDEDOR, IDFACTURA

FROM FACTURA F

ORDER BY IDVENDEDOR;

1. **MOSTRAR LA CANTIDAD DE FACTURAS EMITIDAS POR CADA VENDEDOR(NOMBRE)**

SELECT IDVENDEDOR, IDFACTURA

FROM FACTURA

ORDER BY IDVENDEDOR;

...

SELECT V.NOMBRE, IDFACTURA

FROM FACTURA F, VENDEDOR V

WHERE F.IDVENDEDOR = V.IDVENDEDOR

ORDER BY V.NOMBRE;

...

SELECT V.NOMBRE, COUNT(IDFACTURA)

FROM FACTURA F, VENDEDOR V

WHERE F.IDVENDEDOR = V.IDVENDEDOR

GROUP BY V.NOMBRE

ORDER BY V.NOMBRE;

PUEDE RESOLVERSE CON SUBCONSULTA

# HEAVING



## HEAVING

La cláusula **HAVING** en Oracle se utiliza en conjunción con la cláusula **GROUP BY** para aplicar condiciones a los resultados de grupos de filas, después de que se hayan realizado las operaciones de agregación.

En esencia, **HAVING** filtra los resultados después de que se han aplicado las funciones de agregación y se han agrupado los datos. Esto significa que puedes utilizar **HAVING** para establecer condiciones en los resultados basados en los valores agregados.

Por ejemplo, supongamos que tienes una tabla llamada **ventas** con columnas **producto**, **cantidad** y **precio\_unitario**, y quieres encontrar los productos cuya cantidad total vendida es mayor que 100:

SELECT producto, SUM(cantidad)

FROM ventas

GROUP BY producto

HAVING SUM(cantidad) > 100;

En este ejemplo, primero se agrupan las filas por producto y se calcula la suma de la cantidad vendida para cada producto. Luego, la cláusula **HAVING** filtra esos resultados y solo muestra los productos cuya cantidad total vendida es mayor que 100.

Es importante notar que, a diferencia de **WHERE**, que se aplica antes de la agrupación y se refiere a filas individuales, **HAVING** se aplica después de la agrupación y se refiere a grupos de filas.

En resumen, **HAVING** es útil cuando deseas establecer condiciones en los resultados agregados basados en funciones de agregación como SUM, COUNT, AVG, etc., después de haber agrupado los datos utilizando **GROUP BY**.

## EJEMPLOS

1. **MOSTRAR TODAS LAS FACTURAS CON MAS DE 5 ARTICULOS**

SELECT IDFACTURA, COUNT(IDARTICULO) AS CANTIDAD\_ARTICULOS

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDFACTURA

HAVING COUNT(IDARTICULO) > 5

ORDER BY IDFACTURA;

O

SELECT F.IDFACTURA, COUNT(AXF.IDARTICULO) AS CANTIDAD\_ARTICULOS

FROM FACTURA F

JOIN ARTICULOS\_X\_FACTURA AXF ON F.IDFACTURA = AXF.IDFACTURA

GROUP BY F.IDFACTURA

HAVING COUNT(AXF.IDARTICULO) > 5;

1. **MOSTRAR LOS VENDEDORES QUE HAYAN REALIZADO AL MENOS 8 VENTAS**

SELECT IDVENDEDOR, COUNT(IDFACTURA)

FROM FACTURA F

GROUP BY IDVENDEDOR

HAVING COUNT(IDFACTURA) >= 8;

O

SELECT V.NOMBRE, COUNT(F.IDFACTURA)

FROM FACTURA F

JOIN VENDEDOR V ON F.IDVENDEDOR = V.IDVENDEDOR

GROUP BY V.NOMBRE

HAVING COUNT(F.IDFACTURA) >= 8;

1. **MOSTRAR LOS ARTICULOS CUYA SUMA TOTAL DE UNIDADES VENDIDAS SUPEREN A 40**

SELECT IDARTICULO

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO

HAVING SUM(CANTIDAD) > 40;

# SUBCONSULTAS



## SUBCONSULTAS

Una subconsulta (también conocida como consulta anidada o subquery) es una consulta SQL que se encuentra dentro de otra consulta. La subconsulta se ejecuta primero y su resultado se utiliza como parte de la condición o expresión en la consulta principal.

En esencia, una subconsulta permite realizar una operación basada en los resultados de otra consulta. Esto puede ser útil para filtrar datos, realizar cálculos o hacer comparaciones más complejas.

Hay dos tipos principales de subconsultas:

**1. Subconsultas correlacionadas**: Una subconsulta correlacionada es aquella en la que la subconsulta depende de los resultados de la consulta externa. Es decir, la subconsulta se ejecuta una vez por cada fila procesada por la consulta externa.

Ejemplo de subconsulta correlacionada:

SELECT nombre

FROM empleados

WHERE salario > (SELECT AVG(salario) FROM empleados WHERE departamento = e.departamento);

**2. Subconsultas no correlacionadas**: En una subconsulta no correlacionada, la subconsulta es independiente de la consulta externa y se ejecuta solo una vez.

Ejemplo de subconsulta no correlacionada:

SELECT nombre

FROM empleados

WHERE salario > (SELECT AVG(salario) FROM empleados);

Las subconsultas pueden aparecer en diferentes partes de una consulta SQL, como en la cláusula **WHERE**, **FROM**, **SELECT**, **HAVING**, etc. Su uso es bastante flexible y depende de los requisitos específicos de la consulta que estés realizando.

Las subconsultas son una herramienta poderosa en SQL y te permiten realizar consultas más complejas y sofisticadas al aprovechar los resultados de otras consultas.

## EJEMPLOS

1. **MOSTRAR LOS ARTICULOS CON UN PRECIO SUPERIOR AL PROMEDIO**

SELECT IDARTICULO, DESCRIPCION, PRECIOUNITARIO

FROM ARTICULO

WHERE PRECIOUNITARIO > (SELECT AVG(PRECIOUNITARIO) FROM ARTICULO);

1. **MOSTRAR LOS CLIENTES QUE HAN REALIZADO AL MENOS UNA COMPRA**

SELECT IDCLIENTE, NOMBRE, APELLIDO

FROM CLIENTE

WHERE IDCLIENTE IN (SELECT DISTINCT IDCLIENTE FROM FACTURA);

1. **MOSTRAR LAS FACTURAS DE UN VENDEDOR ESPECIFICOS**

SELECT IDFACTURA, TO\_CHAR(FECHAFACTURA,'DD/MM/YYYY')

FROM FACTURA

WHERE IDVENDEDOR = (SELECT IDVENDEDOR FROM VENDEDOR WHERE NOMBRE = 'Vanessa')

ORDER BY IDFACTURA;

1. **MOSTRAR LOS ARTICULOS COMPRADOS EN UNA FACTURA ESPECIFICA**

SELECT A.IDARTICULO, A.DESCRIPCION

FROM ARTICULO A

WHERE A.IDARTICULO IN (SELECT AF.IDARTICULO FROM ARTICULOS\_X\_FACTURA AF WHERE AF.IDFACTURA = 'IDF001');

1. **MOSTRAR LOS VENDEDORES QUE NO HAN REALIZADO NINGUNA VENTA**

SELECT IDVENDEDOR, NOMBRE

FROM VENDEDOR

WHERE IDVENDEDOR NOT IN (SELECT DISTINCT IDVENDEDOR FROM FACTURA);

1. **MOSTRAR TODAS LAS FACTURAS QUE TENGAN EL ARTICULO BORRADOR**

SELECT IDFACTURA

FROM ARTICULOS\_X\_FACTURA

WHERE IDARTICULO = (SELECT IDARTICULO FROM ARTICULO WHERE DESCRIPCION = 'Borrador')

PUEDE RESOLVERSE CON JOIN

1. **MOSTRAR LAS FACTURAS QUE TENGAN BORRADOR O LAPIZ**

SELECT DISTINCT IDFACTURA

FROM ARTICULOS\_X\_FACTURA

WHERE IDARTICULO IN (SELECT IDARTICULO FROM ARTICULO WHERE DESCRIPCION = 'Borrador' OR DESCRIPCION = 'Lápiz')

ORDER BY IDFACTURA;

1. **MOSTRAR LAS FACTURAS QUE TENGA BORRADOR Y LAPIZ AL MISMO TIEMPO**

SELECT IDFACTURA, COUNT(IDFACTURA)

FROM ARTICULOS\_X\_FACTURA

WHERE IDARTICULO IN (SELECT IDARTICULO FROM ARTICULO WHERE DESCRIPCION = 'Borrador' OR DESCRIPCION = 'Lápiz')

GROUP BY IDFACTURA

HAVING COUNT(IDFACTURA) = 2

ORDER BY IDFACTURA;

1. **MOSTRAR LOS NOMBRES DE LOS VENDEDORES QUE HAYAN REALIZADO AL MENOS UNA VENTANA**

SELECT NOMBRE

FROM VENDEDOR

WHERE IDVENDEDOR IN (SELECT IDVENDEDOR FROM FACTURA);

1. **MOSTRAR LOS NOMBRES DE LOS CLIENTES QUE HAN COMPRADO UN ARTICULO CON UN PRECIO UNITARIO MAYOR A 100**

SELECT NOMBRE

FROM CLIENTE

WHERE IDCLIENTE IN (SELECT IDCLIENTE

FROM FACTURA F

JOIN ARTICULOS\_X\_FACTURA AF ON AF.IDFACTURA = F.IDFACTURA

JOIN ARTICULO A ON AF.IDARTICULO = A.IDARTICULO

WHERE PRECIOUNITARIO > 100);

1. **MOSTRAR LA CANTIDAD DE FACTURAS EMITIDAS POR CADA VENDEDOR(NOMBRE)**

SELECT NOMBRE, (

SELECT COUNT(\*)

FROM FACTURA

WHERE FACTURA.IDVENDEDOR = VENDEDOR.IDVENDEDOR

) AS CANTIDAD\_DE\_FACTURAS

FROM VENDEDOR;

1. **MOSTRAR LOS ARTICULOS QUE SE HAN VENDIDO MAS DE 50 VECES**

SELECT DESCRIPCION

FROM ARTICULO

WHERE IDARTICULO IN (

SELECT IDARTICULO

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO

HAVING SUM(CANTIDAD) > 3

# CRUD



## INSERT

1. **INSERTAR UN CLIENTE**

INSERT INTO CLIENTE(IDCLIENTE,NOMBRE,APELLIDO,TELEFONO,DIRECCION,EDAD,SEXO,EMAIL) VALUES ('IDC006','Angela','Prado',66666666,'C. Orbegozo 394',45,'M','angela@gmail.com');

INSERT INTO CLIENTE(IDCLIENTE,NOMBRE,APELLIDO,TELEFONO,DIRECCION,EDAD,SEXO,EMAIL) VALUES ('IDC007','Casandra','Vazquez',77777777,NULL,45,'M','casandra@hotmail.es');

## SELECT

1. **MOSTRAR LAS FACTURAS DE UN VENDEDOR**

SELECT \* FROM FACTURA WHERE IDVENDEDOR = 'V10001';

## UPDATE

1. **ACTUALIZAR EL PRECIO DE UN ARTICULO**

UPDATE ARTICULO SET PRECIOUNITARIO = 12.0 WHERE IDARTICULO = 'IDA001';

## DELETE

1. **ELIMINAR UN CLIENTE**

DELETE FROM CLIENTE WHERE IDARTICULO = 'IDC006';

DELETE FROM CLIENTE WHERE IDARTICULO = 'IDC007';

# PL/SQL



## PL/SQL

El símbolo **/** al final de tu código PL/SQL en Oracle es un comando especial utilizado en SQLPlus, la interfaz de línea de comandos de Oracle. Este símbolo indica a SQLPlus que ejecute el bloque de código PL/SQL inmediatamente.

En tu caso, después de definir el bloque de código PL/SQL, se incluye el **/** para ejecutarlo. En términos prácticos, esto indica a la interfaz de Oracle que tome el bloque de código y lo ejecute como un solo script.

Entonces, si ejecutas este código en SQLPlus, el bloque PL/SQL será ejecutado automáticamente debido a la presencia del **/** al final del script. Si estás utilizando una interfaz diferente a SQLPlus, es posible que no sea necesario incluir el **/**, ya que otras herramientas pueden manejar los scripts de manera diferente.

## EJEMPLOS

1. **ELIMINAR UNA TABLA SI EXISTE Y NO HACER NADA SI NO EXISTE**

BEGIN

IF EXISTS (SELECT \* FROM USER\_TABLES WHERE TABLE\_NAME = 'nombre\_de\_tu\_tabla') THEN

EXECUTE IMMEDIATE 'DROP TABLE nombre\_de\_tu\_tabla';

END IF;

END;

/

1. **MOSTRAR UN MENSAJE DE VALIDO O NO A LADO DEL NOMBRE DE CADA CLIENTE, SI EL CORREO TINE UNA EXTENSION DE 3 LETRAS**

SELECT

NOMBRE,

CASE

WHEN REGEXP\_LIKE(EMAIL, '^[A-Za-z0-9]+@[A-Za-z0-9]+\.[A-Za-z]{3}$')

THEN 'Válido'

ELSE 'No Válido'

END AS EMAIL\_VALIDO

FROM CLIENTE;

# EJERCICIOS DIVERSOS



## EJEMPLOS

1. **CONCATENAR VALORES DE DIFERENTES TABLAS**

SELECT DISTINCT C.NOMBRE || ' compró ' || A.DESCRIPCION AS COMPRA\_REALIZADA

FROM FACTURA F

JOIN CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE

JOIN ARTICULOS\_X\_FACTURA AF ON F.IDFACTURA = AF.IDFACTURA

JOIN ARTICULO A ON AF.IDARTICULO = A.IDARTICULO

ORDER BY COMPRA\_REALIZADA;

1. **CONCATENAR VALORES DE UNA SUBCONSULTA**

SELECT 'El cliente ' || (SELECT NOMBRE FROM CLIENTE WHERE IDCLIENTE = 'IDC001') || ' tiene una dirección en ' || DIRECCION AS INFO\_DIRECCION

FROM CLIENTE

WHERE IDCLIENTE = 'IDC001';

1. **MOSTRAR EL O LOS ARTICULO MAS VENDIDO. EL QUE TENGA MAS TRANSACCIONES**

(\*) SOLUCION NO OPTIMA

SELECT IDARTICULO, COUNT(IDFACTURA)

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO

ORDER BY IDARTICULO;

...

SELECT IDARTICULO, COUNT(IDFACTURA) AS CANTIDAD\_TRANSACCIONES

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO

ORDER BY CANTIDAD\_TRANSACCIONES DESC

FETCH FIRST 1 ROW ONLY;

(\*) SOLUCION OPTIMA

SELECT IDARTICULO, COUNT(IDFACTURA) AS CANTIDAD\_TRANSACCIONES

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO

HAVING COUNT(IDFACTURA) = (SELECT COUNT(IDFACTURA) AS CANTIDAD\_TRANSACCIONES

FROM ARTICULOS\_X\_FACTURA

GROUP BY IDARTICULO

ORDER BY CANTIDAD\_TRANSACCIONES DESC

FETCH FIRST 1 ROW ONLY);

1. **MOSTRAR EL TOTAL DE VENTAS POR VENDEDOR**

SELECT V.IDVENDEDOR, V.NOMBRE AS NOMBRE\_VENDEDOR,

SUM(AF.CANTIDAD \* AR.PRECIOUNITARIO) AS TOTAL\_VENTAS

FROM VENDEDOR V

JOIN FACTURA F ON V.IDVENDEDOR = F.IDVENDEDOR

JOIN ARTICULOS\_X\_FACTURA AF ON F.IDFACTURA = AF.IDFACTURA

JOIN ARTICULO AR ON AF.IDARTICULO = AR.IDARTICULO

GROUP BY V.IDVENDEDOR, V.NOMBRE;

1. **DIVIDIR UNA CADENA EN PARTES**

SELECT SUBSTR('Juan|Pérez|30', 1, INSTR('Juan|Pérez|30', '|') - 1) AS Nombre,

SUBSTR('Juan|Pérez|30', INSTR('Juan|Pérez|30', '|') + 1,

INSTR('Juan|Pérez|30', '|', 1, 2) - INSTR('Juan|Pérez|30', '|') - 1) AS Apellido,

SUBSTR('Juan|Pérez|30', INSTR('Juan|Pérez|30', '|', 1, 2) + 1) AS Edad

FROM DUAL;

1. **MOSTRAR EL TOTAL DE VENTA POR CADA VENDEDOR**

SELECT F.IDVENDEDOR, SUM(A.PRECIOUNITARIO \* AF.CANTIDAD)

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A, FACTURA F

WHERE AF.IDARTICULO = A.IDARTICULO AND F.IDFACTURA = AF.IDFACTURA

GROUP BY F.IDVENDEDOR;

O

SELECT IDVENDEDOR, SUM(PRECIOUNITARIO \* CANTIDAD) AS TotalVentas

FROM FACTURA F

JOIN ARTICULOS\_X\_FACTURA AF ON F.IDFACTURA = AF.IDFACTURA

JOIN ARTICULO A ON AF.IDARTICULO = A.IDARTICULO

GROUP BY IDVENDEDOR;

1. **ANALIZAR**

SELECT F.IDFACTURA, F.FECHAFACTURA, C.NOMBRE AS NOMBRE\_CLIENTE, V.NOMBRE AS NOMBRE\_VENDEDOR,

A.DESCRIPCION, AXF.CANTIDAD, A.PRECIOUNITARIO, (AXF.CANTIDAD \* A.PRECIOUNITARIO) AS TOTAL

FROM FACTURA F

JOIN CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE

JOIN VENDEDOR V ON F.IDVENDEDOR = V.IDVENDEDOR

JOIN ARTICULOS\_X\_FACTURA AXF ON F.IDFACTURA = AXF.IDFACTURA

JOIN ARTICULO A ON AXF.IDARTICULO = A.IDARTICULO;

1. **MOSTRAR EL NOMBRE DEL CLIENTE POR CADA FACTURA**

SELECT F.IDFACTURA, F.FECHAFACTURA, C.NOMBRE AS NOMBRE\_CLIENTE, V.NOMBRE AS NOMBRE\_VENDEDOR,

A.DESCRIPCION, AXF.CANTIDAD, A.PRECIOUNITARIO, (AXF.CANTIDAD \* A.PRECIOUNITARIO) AS TOTAL

FROM FACTURA F

JOIN CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE

JOIN VENDEDOR V ON F.IDVENDEDOR = V.IDVENDEDOR

JOIN ARTICULOS\_X\_FACTURA AXF ON F.IDFACTURA = AXF.IDFACTURA

JOIN ARTICULO A ON AXF.IDARTICULO = A.IDARTICULO;

1. **MOSTRAR EL NOMBRE DE LOS ARTICULOS POR CADA FACTURA**

SELECT F.IDFACTURA, F.FECHAFACTURA, C.NOMBRE AS NOMBRE\_CLIENTE, V.NOMBRE AS NOMBRE\_VENDEDOR,

A.DESCRIPCION, AXF.CANTIDAD, A.PRECIOUNITARIO, (AXF.CANTIDAD \* A.PRECIOUNITARIO) AS TOTAL

FROM FACTURA F

JOIN CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE

JOIN VENDEDOR V ON F.IDVENDEDOR = V.IDVENDEDOR

JOIN ARTICULOS\_X\_FACTURA AXF ON F.IDFACTURA = AXF.IDFACTURA

JOIN ARTICULO A ON AXF.IDARTICULO = A.IDARTICULO;

1. **MOSTRAR LOS ARTICULOS QUE NO TIENEN NINGUNA VENTA**

Primero obtengo los idarticulo que tiene una venta

SELECT DISTINCT IDARTICULO

FROM ARTICULOS\_X\_FACTURA;

Luego obtengo los articulos que no tienen ninguna venta

SELECT \*

FROM ARTICULO

WHERE IDARTICULO NOT IN (SELECT DISTINCT IDARTICULO

FROM ARTICULOS\_X\_FACTURA);

1. **MOSTRAR LOS ARTICULOS CON SUS VENTAS TOTALES ORDENADO DE MAYOR A MENOR**

SELECT AF.IDARTICULO, SUM(A.PRECIOUNITARIO\*AF.CANTIDAD) AS TOTAL\_VENTA

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO

GROUP BY AF.IDARTICULO

ORDER BY TOTAL\_VENTA DESC;

1. **MOSTRAR EL O LOS ARTICULOS CON LA MAYOR VENTA TOTAL**

SELECT AF.IDARTICULO, SUM(A.PRECIOUNITARIO\*AF.CANTIDAD) AS TOTAL\_VENTA

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO

GROUP BY AF.IDARTICULO

HAVING SUM(A.PRECIOUNITARIO\*AF.CANTIDAD) = (SELECT SUM(A.PRECIOUNITARIO\*AF.CANTIDAD) AS TOTAL\_VENTA

FROM ARTICULOS\_X\_FACTURA AF, ARTICULO A

WHERE AF.IDARTICULO = A.IDARTICULO

GROUP BY AF.IDARTICULO

ORDER BY TOTAL\_VENTA DESC

FETCH FIRST 1 ROW ONLY);

1. **MOSTRAR UNA LISTA QUE SEAN LOS SALARIOS MAXIMO ENTRE LOS VENDEDORES POR DEPARTAMENTO**

SELECT V.IDDEPARTAMENTO, D.NOMBRE, MAX(V.SALARIO)

FROM DEPARTAMENTO D, VENDEDOR V

WHERE V.IDDEPARTAMENTO = D.IDDEPARTAMENTO

GROUP BY V.IDDEPARTAMENTO, D.NOMBRE;