

## Tema 3. CONSULTAS SENCILLAS.

### Consulta de los datos.

Realizar una consulta en SQL consiste en recuperar u obtener aquellos datos que, almacenados en filas y columnas de una o varias tablas de una base de datos, cumplen unas determinadas especificaciones. Para realizar cualquier consulta se utiliza la sentencia **SELECT**.

Las primeras consultas van a ser escritas con un formato inicial de la sentencia **SELECT**, que se irá completando en temas siguientes.

#### Formato inicial de la sentencia **SELECT**

```
SELECT ???ALL  ??????*????????????????????????????>
      ?DISTINCT?  ?lista_de_elementos?

>??FROM lista_de_tablas ??????????????????????????>

>????????????????????????????????????????????????>
      ?WHERE condición_de_selección?

>????????????????????????????????????????????????>;
      ?ORDER BY especificaciones_para_ordenar?
```

#### Tablas utilizadas:

##### TABLA DE EMPLEADOS

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISIÓN	DEP_NO
7499	ALONSO	VENDEDOR	7698	20/02/81	140000	40000	30
7521	LOPEZ	EMPLEADO	7782	08/05/81	135000		10
7654	MARTIN	VENDEDOR	7698	28/09/81	150000	160000	30
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7782	MARTINEZ	DIRECTOR	7839	09/06/81	245000		10
7839	REY	PRESIDENTE		17/11/81	600000		10
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30
7876	GIL	ANALISTA	7782	06/05/82	335000		20
7900	JIMENEZ	EMPLEADO	7782	24/03/83	140000		20

##### TABLA DE DEPARTAMENTOS

DEP_NO	DNOMBRE	LOCALIDAD
10	CONTABILIDAD	BARCELONA
20	INVESTIGACION	VALENCIA

## Lenguaje SQL

30 VENTAS	MADRID
40 PRODUCCION	SEVILLA

### ***Consultas sencillas.***

La consulta más sencilla consiste en recuperar una o varias columnas de una tabla.

```
SELECT ???ALL ?????? * ?????????????????????????????????>
        ?DISTINCT? ?lista_de_elementos?

>??FROM tabla ?????????????????????????????????????????>;
```

- ***lista\_de\_elementos***: nombres de columnas o expresiones obtenidas a partir de ellas, y separadas por comas, que son seleccionadas de cada fila para conocer sus valores.
- **\***: selecciona todas las columnas de la tabla.
- **ALL**: obtiene los valores de todos los elementos seleccionados en todas las filas, aunque sean repetidos. Es la opción por defecto.




En Acces la opción ALL va seguida de \* : `SELECT ALL * FROM EMPLEADOS;`

- **DISTINCT**: obtiene los valores no repetidos de todos los elementos.
- **FROM *tabla***: indica el nombre de la tabla en la que se realiza la consulta. Si la tabla no es de la propiedad del usuario, aunque tenga permiso de acceso, deberá usarse con ***nombre\_propietario.tabla***.

***Alias de tabla.*** SQL permite asignar más de un nombre a la misma tabla, dentro de la misma consulta. Usar alias para una tabla puede ser opcional cuando su finalidad consiste en simplificar su nombre original, y obligatorio en consultas cuya sintaxis lo requiera.

```
SELECT .....
      FROM empleados e
      .....
                        alias de empleados → e
```

## Lenguaje SQL

**Recuerda:** para probar las sentencias SQL en Access, desde la vista de la base de datos, elegir el objeto *Consultas* y doble clic en *Crear una consulta en vista Diseño*. Cerrar la ventana *Mostrar Tabla*, y a continuación abrir la *vista SQL* pulsando al botón  de la barra de herramientas. En Access podremos guardar todas las consultas que hagamos si elegimos la opción *Guardar Como* del menú *Archivo*. Tendremos que dar un nombre a cada consulta. Para ver el resultado de la consulta pulsaremos al botón *Ejecutar consulta*  o al botón *Vista hoja de datos* .

### Ejemplos.

1. Obtener todos los empleados de la tabla *empleados* con todos sus datos.

```
SQL> SELECT * FROM empleados;
```

```
SQL> SELECT ALL FROM empleados;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7499	ALONSO	VENDEDOR	7698	20/02/81	140000	40000	30
7521	LOPEZ	EMPLEADO	7782	08/05/81	135000		10
7654	MARTIN	VENDEDOR	7698	28/09/81	150000	160000	30
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7782	MARTINEZ	DIRECTOR	7839	09/06/81	245000		10
7839	REY	PRESIDENTE		17/11/81	600000		10
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30
7876	GIL	ANALISTA	7782	06/05/82	335000		20
7900	JIMENEZ	EMPLEADO	7782	24/03/83	140000		20

En Acces: 

```
SELECT ALL * FROM EMPLEADOS;  
SELECT * FROM EMPLEADOS;
```

2. Obtener los números de empleados, los apellidos y el número de departamento de todos los empleados de la tabla *empleados*.

```
SQL> SELECT emp_no, apellido, dep_no FROM empleados;
```

EMP_NO	APELLIDO	DEP_NO
7499	ALONSO	30
7521	LOPEZ	10
7654	MARTIN	30
7698	GARRIDO	30
7782	MARTINEZ	10
7839	REY	10
7844	CALVO	30
7876	GIL	20
7900	JIMENEZ	20

**Alias de columna.** Los títulos o cabeceras que muestra la salida de una consulta para las columnas seleccionadas, se corresponden con los nombres de las columnas de las tablas. Para mejorar su legibilidad y estética se utilizan los **alias de columna**. El alias se escribe detrás de la columna, separado de ella al menos por un espacio. Si el alias comprende más de una palabra deberá ir dentro de dobles comillas.

## Lenguaje SQL

---

```
SQL> SELECT emp_no "Nº Empleado", apellido, dep_no Departamento
      FROM empleados;
```

En Acces para poner un alias a una columna utilizamos la palabra **AS** seguida del nombre. El nombre no debe contener espacios:

```
SELECT DISTINCT DEP_NO AS NUM_DEPART FROM EMPLEADOS;
SELECT EMP_NO AS N°EMPLEADO, APELLIDO, DEP_NO AS DEPARTAMENTO
FROM EMPLEADOS;
```

3. Obtener el total a cobrar por cada empleado, suponiendo que se trata de sumar a su salario la correspondiente comisión, si la tuviera.

```
SQL> SELECT apellido,salario+comision "Importe Total"
      FROM empleados;
```

En Acces:

```
SELECT apellido,salario+comision AS Importe_Total FROM empleados;
```

APELLIDO	Importe Total
ALONSO	180000
LOPEZ	
MARTIN	310000
GARRIDO	
MARTINEZ	
REY	
CALVO	180000
GIL	
JIMENEZ	

En esta salida, algunos empleados no llevan importe. El motivo es tener NULL (sin información) en su comisión, y por lo tanto no se realiza la operación suma. Para solucionarlo hemos de recurrir al uso de funciones.

**Sugerencia.- Ver FUNCIONES (En MATERIALES).**

El ejemplo anterior debería resolverse con la función **NVL** que transforma la ausencia de información al valor que se le especifique, tal como sigue:

```
SQL> SELECT apellido,salario+NVL(comision,0) "Importe Total"
      FROM empleados;
```

La nueva salida sería:

APELLIDO	Importe Total
ALONSO	180000
LOPEZ	135000
MARTIN	310000

## Lenguaje SQL

---

GARRIDO	385000
MARTINEZ	245000
REY	600000
CALVO	180000
GIL	335000
JIMENEZ	140000

En Acces:

```
SELECT apellido,salario+NZ(comision,0)AS Importe_Total FROM empleados;
```

4. Concatenar cada empleado con su oficio mediante “es”.

```
SQL> SELECT CONCAT(CONCAT(apellido, ' es '),oficio)
      "Empleado y su oficio"
      FROM empleados;
```

```
Empleado y su oficio
-----
ALONSO es VENDEDOR
LOPEZ es EMPLEADO
MARTIN es VENDEDOR
GARRIDO es DIRECTOR
MARTINEZ es DIRECTOR
REY es PRESIDENTE
CALVO es VENDEDOR
GIL es ANALISTA
JIMENEZ es EMPLEADO
```

En Acces:

```
SELECT APELLIDO + ' ES ' + OFICIO AS EMPLEADO_Y_SU_OFICIO
FROM EMPLEADOS;
```

5. Obtener la fecha de alta de cada empleado con el nombre del mes completo y en castellano.

```
SQL> SELECT TO_CHAR(FECHA_ALTA,'ddmonthyy',
      'NLS_DATE_LANGUAGE=Spanish') "Fecha de alta"
      FROM empleados;
```

```
Fecha de alta
-----
20febrero 81
08mayo 81
28septiembre81
01mayo 81
09junio 81
17noviembre 81
08septiembre81
06mayo 82
24marzo 83
```

En Acces:

```
SELECT Format(FECHA_ALTA, 'DDMMYY') AS Fecha_de_alta
FROM Empleados;
```

### **Tabla DUAL.**

En SQL, toda petición de datos se escribe mediante una consulta, y cualquier consulta debe realizarse sobre una tabla. Cuando la consulta consiste en obtener los valores de una determinada función aplicada a una constante, no a una columna de una tabla, o en acceder a la fecha del sistema, la tabla que se utiliza en la cláusula FROM es la **tabla DUAL**, disponible a todo usuario.

**Ejemplo.** Calcular la quinta potencia de 5.

```
SQL> SELECT POWER(5,5) " 5 Elevado a 5"
      FROM DUAL;
```

```
5 Elevado a 5
-----
          3125
```

En Access no existe la tabla DUAL, sin embargo se pueden realizar operaciones para probar funciones utilizando una tabla.

```
SELECT DISTINCT(5^5) AS 5_Elevado_a_5 FROM EMPLEADOS;
```

### **Condiciones de selección.**

---

Para seleccionar las filas de la tabla sobre las que realizar una consulta, la cláusula **WHERE** permite incorporar una **condición de selección** a la sentencia SELECT.

#### **Formato de consulta con condición de selección**

```
SELECT ???ALL  ?????*????????????????????????>
      ?DISTINCT?  ?lista_de_elementos?

>??FROM lista_de_tablas ??????????????????????>

>????????????????????????????????????????????>;
      ?WHERE condición de selección?
```

**Condición de selección:** expresión formada por columnas de la tabla, constantes, funciones, operadores de comparación y operadores lógicos, que deberá ser cierta para que una fila de la tabla sea seleccionada como parte de la salida obtenida por la consulta.

### . Operadores de comparación.

- aritméticos: = , > , <
- de caracteres: LIKE, máscaras (% , \_)
- lógicos: IN , BETWEEN

**. Operadores lógicos-booleanos:** AND , OR , NOT . Permiten construir condiciones de selección compuestas. El uso de paréntesis ayuda a escribir correctamente, a mejorar la legibilidad de las condiciones compuestas y a establecer prioridades de evaluación para los operadores.

### Ejemplos.

1. Obtener la lista de los empleados vendedores, con su nombre, salario y comisión.

```
SQL> SELECT apellido,salario,comision
      FROM empleados
      WHERE UPPER(oficio)='VENDEDOR' ;
```

APELLIDO	SALARIO	COMISION
ALONSO	140000	40000
MARTIN	150000	160000
CALVO	180000	0

**UPPER(*expresión\_alfabética*)** obtiene la *expresión\_alfabética* en mayúsculas.

En Acces:

```
SELECT APELLIDO,SALARIO,COMISION FROM EMPLEADOS
WHERE UCASE(OFICIO) = 'VENDEDOR' ;
```

2. Seleccionar aquellos empleados cuyo apellido empiece por “M” y su salario esté comprendido entre 100.000 y 200.000 pesetas. Visualizar su número de empleado, apellido y departamento.

```
SQL> SELECT emp_no "Nº Empleado" ,apellido,dep_no Departamento
      FROM empleados
      WHERE (apellido LIKE 'M%') AND
      (salario >=100000 AND salario<= 200000);
```

Nº Empleado	APELLIDO	DEPARTAMENTO
7654	MARTIN	30

## Lenguaje SQL

En Acces:

```
SELECT EMP_NO AS N°EMPLEADO ,APELLIDO, DEP_NO AS DEPARTAMENTO
FROM EMPLEADOS WHERE (APELLIDO LIKE 'M*') AND (SALARIO >=100000 AND
SALARIO<= 200000);
```

El operador **LIKE** usado con % indica que la comparación del apellido se realiza sólo en el primer carácter, que debe ser “M”.El % sustituye al resto de los caracteres.

**En Access NO se utiliza el carácter %, se utilizan los siguientes caracteres comodín:**

- El signo de **interrogación** (?) para sustituir un carácter por cualquiera en esa posición. Por ejemplo ?a busca aquellos valores que empiecen por cualquier letra y la segunda sea una "a".
- El **asterisco** (\*) para representar cualquier número de caracteres situados en la misma posición que el asterisco.

Por ejemplo

Resultado	Criterio
Departamentos cuyo nombre de localidad empieza por B	LIKE 'B*'
Departamentos cuya localidad empieza por M, seguido de 4 letras cualquiera y terminan en D. (MADRID, por ejemplo)	LIKE 'M????D'
Departamentos cuya localidad empieza por cualquier letra, le sigue una A u luego cualquier número de caracteres.	LIKE '?A*'
Departamentos cuya localidad empieza por B y termina en O.	LIKE 'B*O'
Departamentos cuya localidad termina en O.	LIKE '*O'
Todos los departamento que contengan una A en el nombre de la localidad.	LIKE '*A*'

El operador **BETWEEN** comprueba si una expresión toma valores dentro del intervalo que le acompaña.

El mismo ejemplo podría haberse escrito:

```
SQL> SELECT emp_no "N° Empleado" ,apellido,dep_no Departamento
```



## Lenguaje SQL

---

```
FROM empleados
WHERE (apellido LIKE 'M%') AND
salario BETWEEN 100000 AND 200000;
```

En Acces:

```
SELECT EMP_NO AS N°EMPLEADO ,APELLIDO, DEP_NO AS DEPARTAMENTO
FROM EMPLEADOS WHERE (APELLIDO LIKE 'M*') AND
Salario BETWEEN 100000 AND 200000;;
```

3. Seleccionar aquellos empleados cuyo apellido incluya una “z” en el segundo carácter.

```
SQL> SELECT emp_no "N° Empleado" ,apellido,dep_no Departamento
      FROM empleados
      WHERE (apellido LIKE '_Z%') ;
```

ninguna fila seleccionada

El operador **LIKE** usado con `'_'` indica que ocupa la posición de un carácter.

En Acces:

```
SELECT EMP_NO AS N°EMPLEADO ,APELLIDO, DEP_NO AS DEPARTAMENTO
FROM EMPLEADOS WHERE (APELLIDO LIKE '?Z*') ;
```

4. Seleccionar los empleados existentes en los departamentos 10 y 30.

```
SQL>SELECT emp_no "N° Empleado",apellido,dep_no Departamento
      FROM empleados
      WHERE dep_no=10 OR dep_no=30;
```

0

```
SQL>SELECT emp_no "N° Empleado",apellido,dep_no Departamento
      FROM empleados
      WHERE dep_no IN(10,30);
```

N° Empleado	APELLIDO	DEPARTAMENTO
7499	ALONSO	30
7521	LOPEZ	10
7654	MARTIN	30
7698	GARRIDO	30
7782	MARTINEZ	10
7839	REY	10
7844	CALVO	30

El operador **IN** comprueba si una determinada expresión toma alguno de los valores indicados entre paréntesis.

### En Acces:

```
SELECT EMP_NO AS N°EMPLEADO ,APELLIDO, DEP_NO AS DEPARTAMENTO  
FROM EMPLEADOS WHERE Dep_no=10 OR Dep_no=30;
```

```
SELECT EMP_NO AS N°EMPLEADO ,APELLIDO, DEP_NO AS DEPARTAMENTO  
FROM EMPLEADOS WHERE Dep_no IN(10,30);
```

---

## Ordenación.

Para obtener la salida de una consulta clasificada por algún criterio o especificación, la sentencia `SELECT` dispone de la cláusula `ORDER BY` para ordenar.

### Formato de consulta con ordenación

```
SELECT ???ALL  ?????? * ?????????????????????????>  
      ?DISTINCT?  ?lista_de_elementos?  
  
>??FROM lista_de_tablas ?????????????????????>  
  
>????????????????????????????????????????>  
      ?WHERE condición_de_selección?  
  
>????????????????????????????????????????>;  
      ?ORDER BY especificación_para_ordenar?
```

**Especificación\_para\_ordenar** : lista de columnas o expresiones obtenidas a partir de ellas, separadas por comas, y cada una de ellas con indicación del tipo de ordenación.

**Tipo de ordenación:** `ASC` (ascendente) o `DESC` (descendente). Por omisión es `ASC`.

Los nombres de columnas de la *especificación\_para\_ordenar* pueden ser sustituidos por el número de orden que ocupan en la tabla.

Si la *especificación\_para\_ordenar* contiene más de una columna o expresión, el orden en que se realizan las clasificaciones es de izquierda a derecha.

### Ejemplos.

1. Obtener relación alfabética de todos los empleados con todos sus datos.

```
SQL>SELECT * FROM empleados  
      ORDER BY apellido;
```

## Lenguaje SQL

---

### 2. Obtener clasificación alfabética de empleados por departamentos.

```
SQL>SELECT * FROM empleados
      ORDER BY dep_no, apellido;
0
```

```
SQL>SELECT * FROM empleados
      ORDER BY 8,2;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7499	ALONSO	VENDEDOR	7698	20/02/81	140000	40000	30
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7876	GIL	ANALISTA	7782	06/05/82	335000		20
7900	JIMENEZ	EMPLEADO	7782	24/03/83	140000		20
7521	LOPEZ	EMPLEADO	7782	08/05/81	135000		10
7654	MARTIN	VENDEDOR	7698	28/09/81	150000	160000	30

### 3. Obtener los datos de los empleados clasificados por oficios y en orden descendente de salarios.

```
SQL>SELECT * FROM empleados
      ORDER BY oficio,salario DESC;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7876	GIL	ANALISTA	7782	06/05/82	335000		20
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7782	MARTINEZ	DIRECTOR	7839	09/06/81	245000		10
7900	JIMENEZ	EMPLEADO	7782	24/03/83	140000		20
7521	LOPEZ	EMPLEADO	7782	08/05/81	135000		10
7839	REY	PRESIDENTE		17/11/81	600000		10
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30
7654	MARTIN	VENDEDOR	7698	28/09/81	150000	160000	30
7499	ALONSO	VENDEDOR	7698	20/02/81	140000	40000	30