

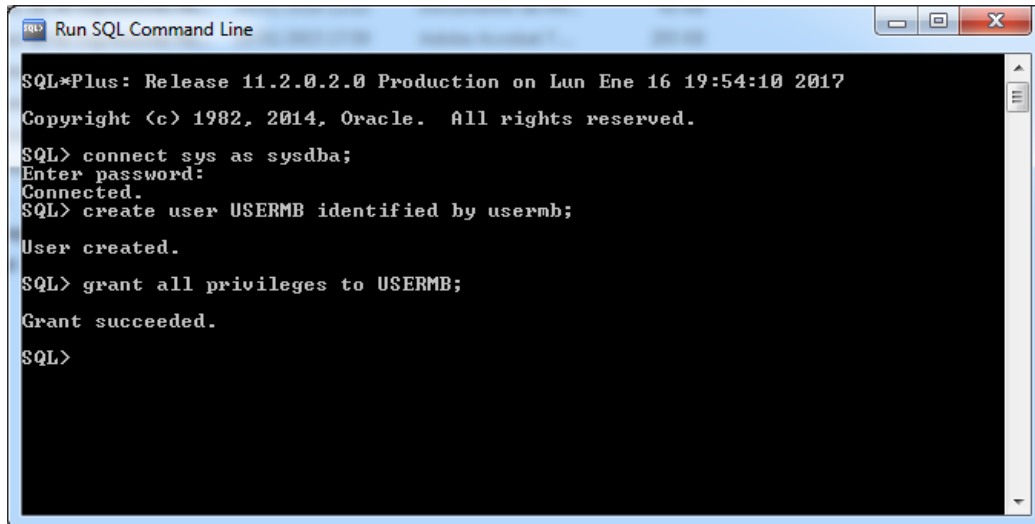
# BASES DE DATOS



# CONSULTAS SIMPLES

## PREPARACIÓN DE LAS TABLAS

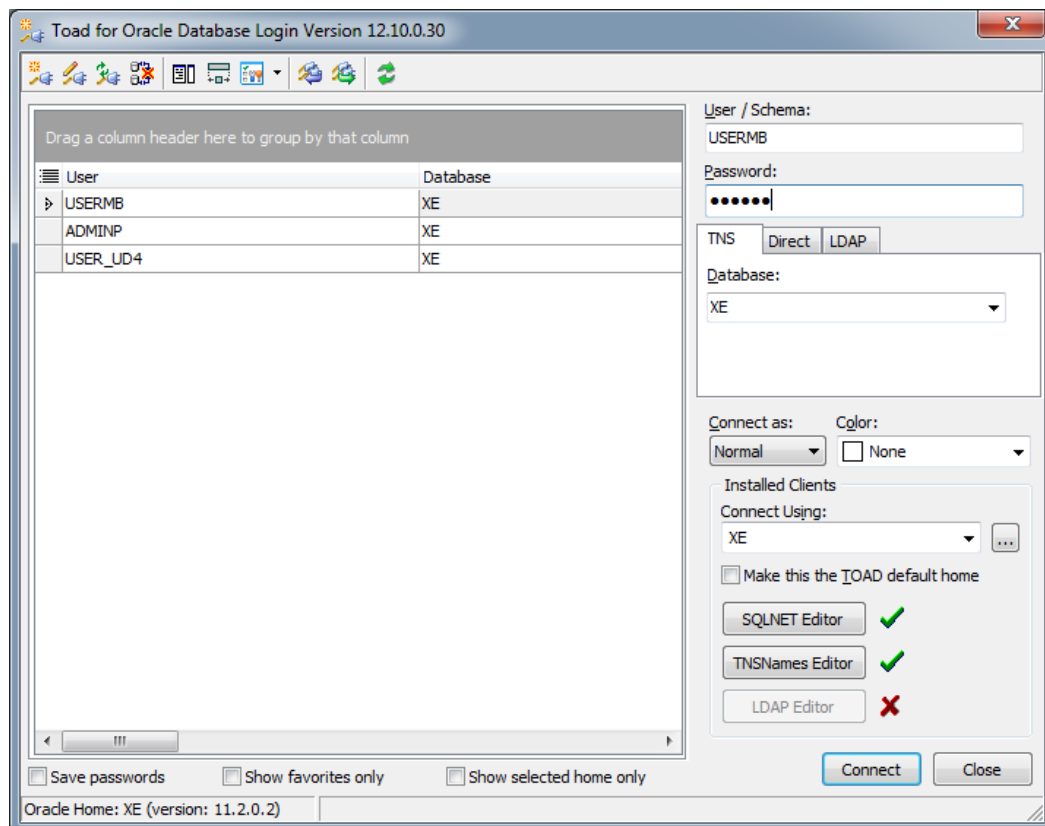
### 1) Crear el usuario USERMB y conectarse desde TOAD con este usuario



```
SQL*Plus: Release 11.2.0.2.0 Production on Lun Ene 16 19:54:10 2017
Copyright (c) 1982, 2014, Oracle. All rights reserved.

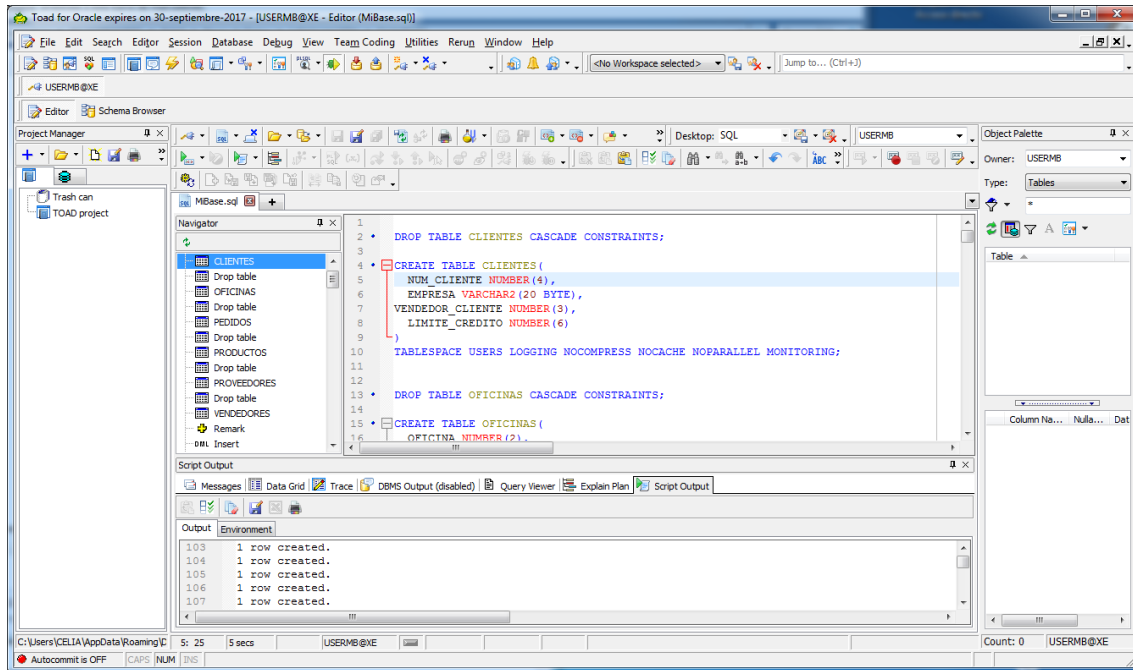
SQL> connect sys as sysdba;
Enter password:
Connected.
SQL> create user USERMB identified by usermb;
User created.
SQL> grant all privileges to USERMB;
Grant succeeded.
SQL>
```

TOAD → Menú Session/New connection



## 2) Lanzar el script MiBase.sql

*Menú File/Open File y después botón Execute as Script*



## 3) Crear las restricciones de clave principal y clave ajena de las tablas según el siguiente modelo relacional.

### Modelo relacional

OFICINAS (oficina, ciudad, región, **director** (FK), objetivo, ventas)

PEDIDOS (num\_pedido, fecha\_pedido, **cliente** (FK) **vendedor** (FK), **fabricante** (FK), **producto** (FK), cantidad, importe, estado)

CLIENTES (num\_cliente, empresa, **vendedor\_cliente** (FK), limite\_credito)

VENDEDORES (num\_emple, nombre, edad, **oficina\_vend** (FK), titulo, fecha\_contrato, **director** (FK), cuota, ventas)

PRODUCTOS (id\_fab, id\_producto, descripción, precio, existencias, **id\_prov** (FK))

PROVEEDORES (id\_prov, empresa, tipo, teléfono, mail)

*ALTER TABLE OFICINAS*

*ADD CONSTRAINT oficinas\_pk PRIMARY KEY (oficina);*

*ALTER TABLE PEDIDOS*

*ADD CONSTRAINT pedidos\_pk PRIMARY KEY (num\_pedido);*

*ALTER TABLE CLIENTES*

*ADD CONSTRAINT clientes\_pk PRIMARY KEY (num\_cliente);*

```
ALTER TABLE VENDEDORES
```

```
ADD CONSTRAINT vendedores_pk PRIMARY KEY (num_emple);
```

```
ALTER TABLE PRODUCTOS
```

```
ADD CONSTRAINT productos_pk PRIMARY KEY (id_fab, id_producto);
```

```
ALTER TABLE PROVEEDORES
```

```
ADD CONSTRAINT proveedores_pk PRIMARY KEY (id_prov);
```

```
ALTER TABLE OFICINAS
```

```
ADD CONSTRAINT oficinas_vendedores_fk FOREIGN KEY (director)  
REFERENCES VENDEDORES(num_emple) ON DELETE SET NULL;
```

```
ALTER TABLE PEDIDOS
```

```
ADD CONSTRAINT pedidos_clientes_fk FOREIGN KEY (cliente)  
REFERENCES CLIENTES(num_cliente) ON DELETE CASCADE;
```

```
ALTER TABLE PEDIDOS
```

```
ADD CONSTRAINT pedidos_vendedores_fk FOREIGN KEY (vendedor)  
REFERENCES VENDEDORES(num_emple) ON DELETE SET NULL;
```

```
ALTER TABLE PEDIDOS
```

```
ADD CONSTRAINT pedidos_productos_fk FOREIGN KEY (fabricante, producto)  
REFERENCES PRODUCTOS(id_fab, id_producto) ON DELETE CASCADE;
```

```
ALTER TABLE CLIENTES
```

```
ADD CONSTRAINT clientes_vendedores_fk FOREIGN KEY (vendedor_cliente)  
REFERENCES VENDEDORES(num_emple) ON DELETE SET NULL;
```

```
ALTER TABLE VENDEDORES
```

```
ADD CONSTRAINT vendedores_oficinas_fk FOREIGN KEY (oficina_vend)  
REFERENCES OFICINAS(oficina) ON DELETE SET NULL;
```

```
ALTER TABLE VENDEDORES
```

```
ADD CONSTRAINT vendedores_vendedores_fk FOREIGN KEY (director)  
REFERENCES VENDEDORES(num_emple) ON DELETE SET NULL;
```

```
ALTER TABLE PRODUCTOS
```

```
ADD CONSTRAINT productos_proveedores_fk FOREIGN KEY (id_prov)  
REFERENCES PROVEEDORES(id_prov) ON DELETE SET NULL;
```

**4) Estas tablas llevan muchos años utilizándose y por ello las fechas tienen valores antiguos (1989 p.ej.) y que al ejecutar el script son transformadas a fechas posteriores al año 2000 (2089 p.ej.).**

**Vamos a ejecutar un par de scripts para adaptar esas fechas a la actualidad (2009 p.ej.). Para ello vamos a restar 80 años a todas las fechas.**

*UPDATE PEDIDOS*

*SET FECHA\_PEDIDO=ADD\_MONTHS(FECHA\_PEDIDO,-(80\*12));*

La sentencia UPDATE actualiza una tabla y SET modifica el valor del campo indicado, en este caso FECHA\_PEDIDO. Ambos se estudiarán más adelante.

ADD\_MONTHS suma al campo fecha indicado el número de meses que se le da como segundo argumento, en nuestro caso con valor negativo porque queremos restarlos.

Ahora hacemos lo mismo con la tabla VENDEDORES y la fecha de contrato:

*UPDATE VENDEDORES*

*SET FECHA\_CONTRATO=ADD\_MONTHS(FECHA\_CONTRATO,-(80\*12));*

NOTA: en la elaboración de las consultas que utilizan fechas, quizás haya que hacer pequeños ajustes por el tiempo transcurrido desde la elaboración de los enunciados.

## REALIZACIÓN DE CONSULTAS SIMPLES EN SQL

### Ejercicio 1

Probar las siguientes sentencias SQL. A la vista de los resultados comentar brevemente que es lo que hacen.

```
SELECT NOMBRE FROM VENDEDORES  
WHERE FECHA_CONTRATO < TO_DATE('01/01/2008','MM/DD/YY');
```

```
SELECT NOMBRE FROM VENDEDORES WHERE VENTAS > CUOTA
```

```
SELECT NOMBRE, VENTAS FROM VENDEDORES  
WHERE DIRECTOR = 104
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE EDAD BETWEEN 50 AND 60
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE EDAD NOT BETWEEN 50 AND 60
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE EDAD >= 50 AND EDAD <= 60
```

```
SELECT NOMBRE FROM VENDEDORES WHERE EDAD IN (50,55,60)
```

```
SELECT NOMBRE FROM VENDEDORES WHERE EDAD NOT IN (50,55,60)
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE EDAD=50 OR EDAD=55 OR EDAD=60
```

```
SELECT EMPRESA FROM CLIENTES WHERE EMPRESA LIKE '%A%'
```

```
SELECT EMPRESA FROM CLIENTES WHERE VENDEDOR_CLIENTE IS NULL
```

```
SELECT NOMBRE, TITULO FROM VENDEDORES  
WHERE DIRECTOR IS NOT NULL AND (EDAD=50 OR EDAD=55)
```

```
SELECT CIUDAD, VENTAS, OBJETIVO FROM OFICINAS  
WHERE VENTAS > OBJETIVO
```

```
SELECT EMPRESA, LIMITE_CREDITO FROM CLIENTES  
WHERE NUM_CLIENTE = 2107
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE VENTAS > CUOTA
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE VENTAS <= CUOTA
```

```
SELECT NUM_PEDIDO, FECHA_PEDIDO, FABRICANTE, PRODUCTO,  
IMPORTE  
FROM PEDIDOS  
WHERE FECHA_PEDIDO BETWEEN TO_DATE('01/10/2009','DD/MM/YY') AND  
TO_DATE('31/12/2009','DD/MM/YY');
```

```
SELECT NOMBRE, VENTAS, CUOTA FROM VENDEDORES  
WHERE VENTAS NOT BETWEEN (0.8 * CUOTA) AND (1.2 * CUOTA)
```

```
SELECT NOMBRE, CUOTA, VENTAS FROM VENDEDORES  
WHERE OFICINA_VEND IN (11, 13, 22)
```

```
SELECT EMPRESA, LIMITE_CREDITO FROM CLIENTES  
WHERE EMPRESA LIKE 'Smith% Corp.'
```

```
SELECT NUM_PEDIDO, PRODUCTO FROM PEDIDOS  
WHERE PRODUCTO LIKE 'A$%BC%' ESCAPE '$'
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE OFICINA_VEND IS NULL
```

```
SELECT NOMBRE FROM VENDEDORES  
WHERE OFICINA_VEND IS NOT NULL
```

```
SELECT NOMBRE, CUOTA, VENTAS FROM VENDEDORES  
WHERE VENTAS < CUOTA OR VENTAS < 300000.00
```

```
SELECT NOMBRE, CUOTA, VENTAS FROM VENDEDORES  
WHERE VENTAS < CUOTA AND VENTAS < 300000.00
```

```
SELECT NOMBRE, CUOTA, VENTAS FROM VENDEDORES  
WHERE VENTAS < CUOTA AND NOT VENTAS < 150000.00
```

## Ejercicio 2

Diseñar tres consultas SQL del tipo de las vistas anteriormente.

## Ejercicio 3

Escribir las SQL's que nos permitan obtener los resultados de las consultas siguientes.

1) Listar el nombre, oficina, y fecha de contrato de todos los empleados.

```
SELECT NOMBRE, OFICINA_VEND, FECHA_CONTRATO  
FROM VENDEDORES;
```

2) Listar una tarifa (ids, descripción y precio) de productos

```
SELECT ID_FAB, ID_PRODUCTO, DESCRIPCION, PRECIO  
FROM PRODUCTOS;
```

3) Repetir la anterior, pero que como título de la primera columna aparezca fabricante en vez de *ID\_FAB*.

```
SELECT ID_FAB AS FABRICANTE, ID_PRODUCTO, DESCRIPCION, PRECIO  
FROM PRODUCTOS;
```

4) Listar la ciudad, región y el superavit (ventas –objetivo) de cada oficina

```
SELECT CIUDAD, REGION, (VENTAS-OBJETIVO) AS SUPERAVIT  
FROM OFICINAS
```

5) De cada producto obtener su fabricante (*ID\_FAB*), *ID\_PRODUCTO*, su descripción y el valor del inventario (*EXISTENCIAS\*PRECIO*)

```
SELECT ID_FAB AS FABRICANTE, ID_PRODUCTO, DESCRIPCION,  
(PRECIO*EXISTENCIAS) AS VALOR  
FROM PRODUCTOS;
```

6) Listar el nombre, mes y año del contrato de cada vendedor.

```
SELECT NOMBRE, EXTRACT(MONTH FROM FECHA_CONTRATO) AS  
MES, EXTRACT(YEAR FROM FECHA_CONTRATO) AS AÑO  
FROM VENDEDORES;
```

Variante mejorada de la anterior mostrando el nombre del mes:



```
SELECT      NOMBRE,      TO_CHAR(FECHA_CONTRATO,'MONTH')      AS
MES,EXTRACT(YEAR FROM FECHA_CONTRATO) AS AÑO
FROM VENDEDORES;
```

**7) Listar las ventas en cada oficina con el formato: “22 tiene ventas de 186,042.00”**

OFICINA 'TIENEVENTASDE'	VENTAS
22 TIENE VENTAS DE	186042
11 TIENE VENTAS DE	692637
12 TIENE VENTAS DE	735042
13 TIENE VENTAS DE	367911
21 TIENE VENTAS DE	835915

```
SELECT OFICINA, 'TIENE VENTAS DE ', VENTAS
FROM OFICINAS;
```

**8) Obtener un listado alfabético de los empleados.**

```
SELECT NOMBRE, NUM_EMPLE, OFICINA_VEND
FROM VENDEDORES
ORDER BY NOMBRE;
```

**9) Obtener un listado de los empleados por orden de antigüedad en la empresa (los de más antigüedad aparecen primero).**

```
SELECT NOMBRE, NUM_EMPLE, FECHA_CONTRATO
FROM VENDEDORES
ORDER BY FECHA_CONTRATO;
```

**10) Obtener un listado de los empleados ordenados por volumen de ventas sacando los de menores ventas primero.**

```
SELECT NOMBRE, NUM_EMPLE, VENTAS
FROM VENDEDORES
ORDER BY VENTAS;
```

**11) Obtener un listado de los empleados por orden de antigüedad en la empresa empezando por los más recientemente incorporados.**

```
SELECT NOMBRE, NUM_EMPLE, FECHA_CONTRATO
FROM VENDEDORES
ORDER BY FECHA_CONTRATO DESC;
```

**12) Obtener un listado de los empleados ordenados por volumen de ventas sacando primero los de mayores ventas.**

```
SELECT NOMBRE, NUM_EMPLE, VENTAS  
FROM VENDEDORES  
ORDER BY VENTAS DESC;
```

**13) Mostrar las ventas de cada oficina, ordenadas por orden alfabético de región y dentro de cada región por ciudad.**

```
SELECT REGION, CIUDAD, VENTAS  
FROM OFICINAS  
ORDER BY REGION, CIUDAD;
```

**14) Listar las oficinas clasificadas por región y dentro de cada región por superávit (ventas-objetivo) de modo que las de mayor superávit aparezcan las primeras.**

```
SELECT REGION, CIUDAD, (VENTAS - OBJETIVO) AS SUPERAVIT  
FROM OFICINAS  
ORDER BY REGION, SUPERAVIT DESC;
```

**15) Listar los códigos de los directores de las oficinas. El director 108 aparece en dos oficinas, por lo tanto aparecerá dos veces en el resultado de la consulta.**

```
SELECT DIRECTOR FROM OFICINAS
```

O

```
SELECT ALL DIRECTOR FROM OFICINAS
```

**16) Repetir el listado anterior, pero en este caso el valor 108 aparecerá una sola vez.**

```
SELECT DISTINCT DIRECTOR FROM OFICINAS
```

**17) Listar el nombre, la oficina y el rendimiento (ventas-cuota) de los empleados.**

```
SELECT NOMBRE, OFICINA_VEND, (VENTAS - CUOTA) FROM VENDEDORES;
```

**18) Repetir la consulta anterior poniendo como nombre RENDIMIENTO al valor calculado.**

```
SELECT NOMBRE, OFICINA_VEND, (VENTAS - CUOTA) AS RENDIMIENTO  
FROM VENDEDORES;
```

**19) Mostrar las ventas de cada vendedor, ordenadas por la oficina en que trabaja y dentro de cada oficina que aparezcan primero los que más venden.**

```
SELECT OFICINA_VEND, NUM_EMPLE, NOMBRE, VENTAS  
FROM VENDEDORES  
ORDER BY OFICINA_VEND, VENTAS DESC;
```

**20) Listar el nombre de los empleados de la oficina 12.**

```
SELECT NOMBRE  
FROM VENDEDORES  
WHERE OFICINA_VEND = 12;
```

**21) Listar el nombre de los empleados de la oficina 12 que tengan más de 30 años.**

```
SELECT NOMBRE, EDAD  
FROM VENDEDORES  
WHERE OFICINA_VEND = 12 AND EDAD > 30;
```

**22) Listar los empleados cuyas ventas superan su cuota**

```
SELECT NUM_EMPLE, NOMBRE, VENTAS, CUOTA  
FROM VENDEDORES  
WHERE VENTAS > CUOTA;
```

**23) Lista los empleados contratados antes del año 2008 (cuya fecha de contrato sea anterior al 1 de enero de 2008).**

```
SELECT NUM_EMPLE, NOMBRE, FECHA_CONTRATO  
FROM VENDEDORES  
WHERE FECHA_CONTRATO < TO_DATE('01-01-2008','DD-MM-YY');
```

**24) Obtener los empleados cuyo año de la fecha de contrato sea menor que 2008.**

```
SELECT NUM_EMPLE, NOMBRE, FECHA_CONTRATO  
FROM VENDEDORES  
WHERE EXTRACT(YEAR FROM FECHA_CONTRATO) < '2008';
```

**25) Listar las oficinas cuyas ventas estén por debajo del 80% de su objetivo.**

```
SELECT OFICINA, CIUDAD, VENTAS, (OBJETIVO*0.8) AS PORCENTJAE  
FROM OFICINAS  
WHERE VENTAS < OBJETIVO*0.8;
```

**26) Listar las oficinas dirigidas por el empleado 108.**

```
SELECT OFICINA  
FROM OFICINAS  
WHERE DIRECTOR = 108;
```

**27) Lista los empleados cuyas ventas estén comprendidas entre 100.000 y 500.000**

```
SELECT NUM_EMPLE, NOMBRE, VENTAS  
FROM VENDEDORES  
WHERE VENTAS BETWEEN 100000 AND 500000;
```

*De otra forma:*

```
SELECT NUM_EMPLE, NOMBRE, VENTAS  
FROM VENDEDORES  
WHERE (VENTAS >= 100000) AND (VENTAS <= 500000);
```

**28) Listar los empleados de las oficinas 11, 12 y 13**

```
SELECT NUM_EMPLE, NOMBRE, OFICINA_VEND  
FROM VENDEDORES  
WHERE OFICINA_VEND IN (11, 12, 13);
```

*De esta forma obtenemos lo mismo que en el ejemplo anterior.*

```
SELECT NUM_EMPLE, NOMBRE, OFICINA_VEND  
FROM VENDEDORES  
WHERE (OFICINA_VEND = 11) OR (OFICINA_VEND = 12) OR (OFICINA_VEND = 13);
```

**29) Listar las oficinas que no tienen director.**

```
SELECT OFICINA, CIUDAD  
FROM OFICINAS  
WHERE DIRECTOR IS NULL;
```

**30) Listar los empleados asignados a alguna oficina (los que tienen un valor en la columna oficina).**

```
SELECT NUM_EMPLE, NOMBRE  
FROM VENDEDORES  
WHERE OFICINA_VEND IS NOT NULL;
```

**31) Listar los empleados cuyo nombre termine por Smith.**

```
SELECT NUM_EMPLE, NOMBRE  
FROM VENDEDORES  
WHERE NOMBRE LIKE '%Smith';
```

**32) Listar los empleados cuyo nombre contiene Rob.**

```
SELECT NUM_EMPLE, NOMBRE  
FROM VENDEDORES  
WHERE NOMBRE LIKE '%Rob%';
```

**33) Listar los empleados cuyo nombre contenga una r como tercera letra (dos caracteres, la letra r, y cero o más caracteres.**

```
SELECT NUM_EMPLE, NOMBRE  
FROM VENDEDORES  
WHERE NOMBRE LIKE '__r%';
```

**34) Obtener una lista de todos los productos mostrando para cada uno su idfab, idproducto, descripción, precio y precio con I.V.A. incluido (es el precio anterior aumentado en un 21%).**

```
SELECT ID_FAB, ID_PRODUCTO, DESCRIPCION, PRECIO, (PRECIO*1.21) AS  
IVA_INCLUIDO  
FROM PRODUCTOS;
```

**35) De cada pedido queremos saber su número de pedido, fabricante, producto, cantidad, precio unitario (importe/cantidad) e importe.**

```
SELECT NUM_PEDIDO, FABRICANTE, PRODUCTO, CANTIDAD,  
IMPORTE/CANTIDAD AS PRECIO_UNITARIO, IMPORTE  
FROM PEDIDOS
```

**36) Listar de cada empleado su nombre, la fecha de su contrato y los años de antigüedad en la empresa.**

```
SELECT NOMBRE, FECHA_CONTRATO,  
EXTRACT(YEAR FROM CURRENT_DATE) -EXTRACT(YEAR FROM  
FECHA_CONTRATO) AS ANTIGUEDAD  
FROM VENDEDORES;
```

*Aquí hemos utilizado la constante CURRENT\_DATE que devuelve la fecha actual, también podemos utilizar SYSDATE.*

**37) Obtener la lista de los clientes ordenados por código de representante asignado. Visualizar todas la columnas de la tabla.**

```
SELECT *  
FROM CLIENTES  
ORDER BY VENDEDOR_CLIENTE;
```

**38) Obtener las oficinas ordenadas por orden alfabético de región y dentro de cada región por ciudad, si hay más de una oficina en la misma ciudad, aparecerá primero la que tenga el número de oficina mayor.**

```
SELECT *  
FROM OFICINAS  
ORDER BY REGION, CIUDAD, OFICINA DESC;
```

**39) Obtener los pedidos ordenados por fecha de pedido.**

```
SELECT *  
FROM PEDIDOS  
ORDER BY FECHA_PEDIDO;
```

**40) Listar las oficinas, clasificadas en orden descendente de ventas, de modo que las oficinas con mayores ventas aparezcan en primer lugar:**

```
SELECT CIUDAD, REGION, VENTAS  
FROM OFICINAS  
ORDER BY VENTAS DESC
```

**41) Listar las oficinas clasificadas en orden descendente de rendimiento de ventas (ventas-objetivo), de modo que las oficinas con mejor rendimiento aparezcan primero.**

```
SELECT CIUDAD, REGION, (VENTAS - OBJETIVO)  
FROM OFICINAS  
ORDER BY 3 DESC, 1 ASC, 2;
```

*Las filas resultantes están ordenadas de mayor a menor (de forma descendente) por la tercera columna, que es la diferencia calculada entre VENTAS y OBJETIVO para cada oficina. Aquellas filas que tengan el mismo valor en esta tercera columna, se ordenan de menor a mayor (de forma ascendente) por la primera columna: CIUDAD. Y, por último, las filas que tengan el mismo valor en la tercera columna y en la primera, salen ordenadas de forma ascendente por la segunda columna (REGION).*

**Repetir la consulta anterior utilizando un alias y nombres de columnas en la ordenación**

```
SELECT CIUDAD, REGION, (VENTAS-OBJETIVO) AS RENDIMIENTO  
FROM OFICINAS  
ORDER BY RENDIMIENTO DESC, CIUDAD ASC, REGION;
```

**42) Listar toda la información de los pedidos de marzo.**

```
SELECT *  
FROM PEDIDOS  
WHERE EXTRACT(MONTH FROM FECHA_PEDIDO)=3;
```

**43) Listar los números de los empleados que tienen una oficina asignada.**

```
SELECT NUM_EMPLE  
FROM VENDEDORES  
WHERE OFICINA_VEND IS NOT NULL;
```

**44) Listar los números de las oficinas que no tienen director.**

```
SELECT OFICINA  
FROM OFICINAS  
WHERE DIRECTOR IS NULL;
```

**45) Listar los datos de las oficinas de las regiones del norte y del este (tienen que aparecer primero las del norte y después las del este).**

```
SELECT *  
FROM OFICINAS  
WHERE REGION IN ('Norte','Este')  
ORDER BY REGION DESC;
```

**46) Listar los empleados de nombre Bob.**

```
SELECT *  
FROM VENDEDORES  
WHERE NOMBRE LIKE 'Bob%';
```

**47) Listar los productos cuyo idproducto acabe en X.**

```
SELECT *  
FROM PRODUCTOS  
WHERE ID_PRODUCTO LIKE '%X';
```

**48) Mostrar el nombre, las ventas y la cuota del empleado número 105.**

```
SELECT NOMBRE, VENTAS, CUOTA  
FROM VENDEDORES  
WHERE NUM_EMPLE=105;
```

**49) Listar las oficinas cuyas ventas están por debajo del 80 por 100 del objetivo. Mostrar la ciudad, las ventas y el objetivo.**

```
SELECT OFICINA, CIUDAD, VENTAS, OBJETIVO, (0.8*OBJETIVO) AS  
PORCENTAJE  
FROM OFICINAS  
WHERE VENTAS<(0.8*OBJETIVO);
```

**50) Listar las oficinas no dirigidas por el empleado número 108. Mostrar la ciudad y número de empleado del director.**

```
SELECT OFICINA, CIUDAD, DIRECTOR  
FROM OFICINAS  
WHERE DIRECTOR<>108;
```

**51) Hallar los pedidos cuyo importe es superior a 20.000 e inferior a 29.999. Mostrar el número de pedido e importe.**

```
SELECT NUM_PEDIDO, IMPORTE  
FROM PEDIDOS  
WHERE IMPORTE>20000 AND IMPORTE <29999;
```

**52) Hallar los pedidos con fecha anterior al 1 de enero de 2010. Mostrar el número de pedido, el importe y la fecha.**

```
SELECT NUM_PEDIDO, IMPORTE, FECHA_PEDIDO  
FROM PEDIDOS  
WHERE FECHA_PEDIDO<TO_DATE('01/01/2010','MM/DD/YYYY');
```

**53) Hallar todos los pedidos obtenidos por cuatro vendedores específicos (elegir 4 que ya existan). Mostrar el número de pedido, representante e importe.**

```
SELECT NUM_PEDIDO, VENDEDOR, IMPORTE  
FROM PEDIDOS  
WHERE VENDEDOR =102 OR VENDEDOR =103 OR VENDEDOR =104 OR  
VENDEDOR =105;
```



**54) Buscar el nombre de las empresas cuyo nombre...**

- ... acaba en 'Corp.'

```
SELECT EMPRESA
FROM CLIENTES
WHERE EMPRESA LIKE '%Corp.';
```

- ... empieza por "Smiths", acaba en "n" y en el medio hay una letra desconocida.

```
SELECT EMPRESA
FROM CLIENTES
WHERE EMPRESA LIKE 'Smiths_n%';
```

- ... empieza por "Smiths", acaba en "n" y en el medio hay una letra desconocida y acaba en "Corp.".

```
SELECT EMPRESA
FROM CLIENTES
WHERE EMPRESA LIKE 'Smiths_n%Corp.';
```

**55) Hallar todos los nombres de vendedores que cumplan:**

- Trabajan en Denver (22), New York (11) o Chicago (12).

```
SELECT NOMBRE, OFICINA_VEND
FROM VENDEDORES
WHERE OFICINA_VEND =22 OR OFICINA_VEND =11 OR OFICINA_VEND =12;
```

- No tienen director

```
SELECT NOMBRE, OFICINA_VEND
FROM VENDEDORES
WHERE DIRECTOR IS NULL;
```

- Fueron contratados a partir de junio de 2008.

```
SELECT NOMBRE, FECHA_CONTRATO
FROM VENDEDORES
WHERE FECHA_CONTRATO >= TO_DATE('01/06/2008','DD/MM/YYYY');
```

- Sus ventas están por encima de la cuota, pero tienen ventas de 600.000 o menos.

```
SELECT NOMBRE, VENTAS, CUOTA
FROM VENDEDORES
WHERE VENTAS > CUOTA AND VENTAS <= 600000;
```

**56) Por cada producto mostrar el identificador del fabricante, el identificador del producto, su descripción y el inventario (existencias por precio).**

```
SELECT ID_FAB, ID_PRODUCTO, DESCRIPCION, EXISTENCIAS, PRECIO,  
(EXISTENCIAS*PRECIO) AS INVENTARIO  
FROM PRODUCTOS;
```

**57) Calcular a cada vendedor una nueva cuota, incrementando su cuota actual en un 3 por 100 de sus ventas anuales. Mostrar el nombre de vendedor, su cuota actual y su nueva cuota.**

```
SELECT NOMBRE, CUOTA, (CUOTA+CUOTA*0.03) AS NUEVACUOTA  
FROM VENDEDORES;
```

**58) Listar las oficinas, clasificadas en orden alfabético por región, y dentro de cada región por orden descendente de rendimiento de ventas (ventas menos objetivo). Por cada oficina se mostrará la ciudad, la región y el rendimiento de ventas.**

```
SELECT CIUDAD, REGION, VENTAS, (VENTAS-OBJETIVO) AS RENDIMIENTO  
FROM OFICINAS  
ORDER BY REGION, RENDIMIENTO DESC;
```

## CONSULTAS RESUMEN

**1) Obtener una sola fila con el resultado de sumar todos los valores de la columna ventas de la tabla oficinas**

```
SELECT SUM(VENTAS)
FROM OFICINAS;
```

**2) ¿Cuántos empleados tenemos?**

```
SELECT COUNT(NUM_EMPLE)
FROM VENDEDORES;
```

*O también:*

```
SELECT COUNT(*)
FROM VENDEDORES;
```

*En este caso las dos sentencias devuelven el mismo resultado ya que la columna NUM\_EMPLE no contiene valores nulos (es la clave principal de la tabla empleados).*

**3) ¿Cuántos empleados tienen una oficina asignada?**

```
SELECT COUNT(OFICINA)
FROM VENDEDORES;
```

*Esta sentencia nos devuelve el número de valores no nulos que se encuentran en la columna oficina de la tabla empleados, por lo tanto nos dice cuántos empleados tienen una oficina asignada.*

**4) Mostrar el acumulado de ventas de los empleados de la oficina 12.**

```
SELECT SUM(VENTAS)
FROM VENDEDORES
WHERE OFICINA_VEND = 12;
```

**5) Obtener:**

**- la suma de las ventas de todos los vendedores**

```
SELECT SUM(VENTAS)
FROM VENDEDORES;
```

- ¿Cuáles son las cuotas mínima y máxima asignadas a los vendedores?

```
SELECT MIN(CUOTA), MAX(CUOTA)
FROM VENDEDORES
```

- ¿Cuántos vendedores superan su cuota?

```
SELECT COUNT(NOMBRE)
FROM VENDEDORES
WHERE VENTAS > CUOTA
```

6) Obtener una lista con la suma de las ventas de los empleados de cada oficina.

```
SELECT OFICINA_VEND, SUM(VENTAS)
FROM VENDEDORES
GROUP BY OFICINA_VEND;
```

7) Obtener la suma de las ventas de las oficinas agrupadas por región y ciudad:

```
SELECT SUM(VENTAS)
FROM OFICINAS
GROUP BY REGION, CIUDAD;
```

8) Obtener la suma de las ventas de los vendedores agrupados por oficina.  
Llamar a la columna suma VENTAS\_TOTALES

```
SELECT OFICINA_VEND, SUM(VENTAS) AS VENTAS_TOTALES
FROM VENDEDORES
GROUP BY OFICINA_VEND;
```

9) Queremos saber las oficinas con un promedio de ventas de sus empleados mayor que 300.000

```
SELECT OFICINA_VEND
FROM VENDEDORES
GROUP BY OFICINA_VEND
HAVING AVG(VENTAS) > 300000;
```

10) ¿Cuál es la cuota media y las ventas medias de todos los empleados?

```
SELECT AVG(CUOTA) AS CUOTA_MEDIA, AVG(VENTAS) AS VENTAS_MEDIA
FROM VENDEDORES;
```

**11) Hallar el importe medio de pedidos, el importe total de pedidos y el precio medio de venta (el precio de venta es el precio unitario en cada pedido).**

```
SELECT ROUND(AVG(IMPORTE),2) AS IMPORTE_MEDIO,  
SUM(IMPORTE) AS IMPORTE_TOTAL,  
ROUND(AVG(IMPORTE/CANTIDAD),2) AS PRECIO_VENTA_MEDIO  
FROM PEDIDOS;
```

Oracle utiliza las funciones **ROUND()** y **TRUNC()** para el manejo de los remanentes en el caso de pérdida de precisión en conversiones de números.

**ROUND** ( n, integer)

ROUND returns n rounded to integer places to the right of the decimal point. If you omit integer, then n is rounded to 0 places.

**TRUNC** ( n1, n2)

The TRUNC (number) function returns n1 truncated to n2 decimal places. If n2 is omitted, then n1 is truncated to 0 places.

**12) Hallar el precio medio de los productos del fabricante ACI.**

```
SELECT ROUND(AVG(PRECIO),2) AS P_MEDIO_ACI  
FROM PRODUCTOS  
WHERE ID_FAB = 'ACI';
```

Ahora no nos interesan todos los productos sino únicamente los del fabricante ACI, por lo que añadimos la cláusula **WHERE** para que antes de calcular la media, elimine del origen de datos los registros que no cumplan la condición.

**13) Hallar en qué fecha se realizó el primer pedido (suponiendo que en la tabla de pedidos tenemos todos los pedidos realizados hasta la fecha).**

```
SELECT MIN(FECHA_PEDIDO) AS PRIMER_PEDIDO  
FROM PEDIDOS
```

La fecha del primer pedido es la fecha más antigua de la tabla de pedidos.

**14) Hallar cuántos pedidos hay de más de 25000.**

```
SELECT COUNT(*) AS CUANTOS_PEDIDOS_MAYORES  
FROM PEDIDOS  
WHERE IMPORTE > 25000;
```

Se podía haber utilizado también **COUNT(NUM\_PEDIDO)** o cualquier nombre de columna que no pueda contener valores nulos, pero **COUNT(\*)** es mejor por ser más rápido (la diferencia se nota con tablas muy voluminosas).

**15) Listar cuántos empleados están asignados a cada oficina, indicar el número de oficina y cuántos hay asignados.**

```
SELECT OFICINA, COUNT(*) AS CUANTOS_EMPLEADOS  
FROM VENDEDORES  
GROUP BY OFICINA;
```

**16) Para cada empleado cuyos pedidos suman más de 30.000, hallar su importe medio de pedidos. En el resultado indicar el número de empleado y su importe medio de pedidos.**

```
SELECT VENDEDOR, ROUND(AVG(IMPORTE),2) AS IMPORTE_MEDIO  
FROM PEDIDOS  
GROUP BY VENDEDOR  
HAVING SUM(IMPORTE) > 30000;
```

*No queremos todos los empleados, únicamente los que tengan un importe total pedido superior a 30.000, luego tenemos que poner la condición `SUM(importe) > 30000`. Como esta condición contiene una función de columna (`SUM()`) se tiene que poner en la cláusula `HAVING` ya que selecciona filas de la tabla resultante no del origen de datos.*