TABLA RESUMEN DE OPERADORES

	ARITMÉTICOS	IGUALDAD	LÓGICOS	CONCATENACIÓN
Unarios	+, - (signo)			
Binarios	+, -, *, /	<, >, <=, >=, ¡=, =	AND, NOT, OR	||
OPERADORES DE COMPARACIÓN DE CADENAS DE CARACTERES:				
LIKE	Compara cadenas con un patrón de comparación. Este patrón tiene unos comodines: **%** (cualquier cadena de caracteres de longitud 0 o más) _ (sustituye a cualquier carácter, pero sólo uno)			
OPERADORES ESPECÍFICOS:				
[NOT] IN	Operador que permite comprobar si un valor está contenido en una lista de valores literales. Admite el operador [Not] Exp **IN** (x,x,....,x) x **NOT IN** (a,b,c) A nivel de ejecución: (x <> a AND x <> b AND x <> c)			
BETWEEN	Exp **[NOT] BETWEEN** Val_inicial **AND** Val_final Devuelve verdadero si Exp está contenida. Los valores inicial y final inclusive.			
IS NULL	Exp **IS [NOT] NULL** Devuelve verdadero si Exp contiene un nulo.			

PREFERENCIA DE OPERADORES	
UNARIOS	+ -
	* /
BINARIOS	+ - ||
	= < > ¡= <> <= >= IS NULL LIKE BETWEEN IN

OPERADORES DE CONJUNTO	
UNION	Devuelve las filas de la unión de dos consultas eliminando las filas duplicadas. SELECT a, b FROM T1 **UNION** SELECT x, y FROM T2
UNION ALL	Une las filas de la primera consulta con todas las filas de la segunda consulta; esto es, no elimina filas repetidas. SELECT a, b FROM T1 **UNION ALL** SELECT x, y FROM T2
INTERSECT (Intersección conjuntista)	Devuelve las filas que se encuentran tanto en la primera consulta como en la segunda; esto es, las filas repetidas. SELECT a, b FROM T1 **INTERSECT** SELECT x, y FROM T2
MINUS (Resta conjuntista)	Devuelve todas las filas de la primera consulta que no estén en la segunda consulta. SELECT a, b FROM T1 **MINUS** SELECT x, y FROM T2

Funciones SQL

- **Numéricas**

ABS

Descripción Retorna el valor absoluto de *n*.

Ejemplo
```
SELECT ABS(-15) "Absolute" FROM DUAL;

  Absolute
----------
        15
```

CEIL

Descripción Retorna el entero más pequeño mayor o igual a *n*.

Ejemplo
```
SELECT CEIL(15.7) "Ceiling" FROM DUAL;

   Ceiling
----------
        16
```

FLOOR

Descripción Retorna el entero más grande que es menor o igual que *n*.

Ejemplo
```
SELECT FLOOR(15.7) "Floor" FROM DUAL;

     Floor
----------
        15
```

MOD

Sintaxis `MOD(m,n)`

Descripción Retorna el resto de dividir *m* entre *n*. Retorna *m* si *n* es 0.

Ejemplo
```
SELECT MOD(11,4) "Modulus" FROM DUAL;

   Modulus
----------
         3
```

POWER

Sintaxis `POWER(m,n)`

Descripción Retorna *m* elevado a la enésima potencia.

Ejemplo
```
SELECT POWER(3,2) "Raised" FROM DUAL;

    Raised
----------
         9
```

ROUND

Sintaxis `ROUND(n[,m])`

Descripción Retorna *n* redondeado *m* posiciones a la derecha del punto decimal; si *m* se omite, se redondea a las unidades. *m* puede ser negativo, en este caso se redondea la parte entera del número, tantas posiciones a la izquierda del punto decimal como indique *m*.

Ejemplo 1
```
SELECT ROUND(15.193,1) "Round" FROM DUAL;
     Round
----------
      15.2
```

Ejemplo 2
```
SELECT ROUND(15.193,-1) "Round" FROM DUAL;
     Round
----------
        20
```

SIGN

Sintaxis `SIGN(n)`

Descripción Si *n*<0, retorna -1; si *n*=0, devuelve 0; si *n*>0, devuelve 1.

Ejemplo
```
SELECT SIGN(-15) "Sign" FROM DUAL;
      Sign
----------
        -1
```

SQRT

Descripción Devuelve la raíz cuadrada de *n*. *n* no puede ser negativo.

Ejemplo
```
SELECT SQRT(26) "Square root" FROM DUAL;
Square root
-----------
5.09901951
```

TRUNC

Descripción Devuelve *n* truncado *m* posiciones decimales; si *m* se omite, trunca a las unidades; *m* puede ser negativo, en este caso, trunca de la parte entera de *n*.

Ejemplos
```
SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;
  Truncate
----------
      15.7
SELECT TRUNC(15.79,-1) "Truncate" FROM DUAL;
  Truncate
----------
        10
```

- **De cadena de caracteres (varchar2)**

CHR

Sintaxis `CHR(n)`

Descripción Devuelve el carácter equivalente al número *n* de la tabla de idioma de la base de datos.

Ejemplo
```
SELECT CHR(67)||CHR(65)||CHR(84) "Dog" FROM DUAL;
Dog
---
CAT
```

CONCAT

Sintaxis
```
CONCAT(char1, char2)
```

Descripción Devuelve el string *char1* concatenado con *char2*. Esta función es equivalente al operador (||).

Ejemplo
```
SELECT CONCAT( CONCAT(ename, ' is a '), job) "Job"
FROM emp WHERE empno = 7900;
Job
------------------
JAMES is a CLERK
```

INITCAP

Descripción Devuelve un string donde la primera letra de cada palabra está en mayúscula.

Ejemplo
```
SELECT INITCAP('the soap') "Capitals" FROM DUAL;
Capitals
---------
The Soap
```

LOWER

Descripción Devuelve un string donde todas las letras están en minúsculas.

Ejemplo
```
SELECT LOWER('MR. SCOTT MCMILLAN') "Lowercase" FROM DUAL;
Lowercase
--------------------
mr. scott mcmillan
```

LPAD

Descripción Devuelve *char1*, rellenado por la izquierda hasta *n* caracteres con el string *char2*; *char2* puede ser un espacio en blanco. Si la longitud de *char1* es menor que *n*, LPAD devuelve el substring de *char1* con *n* caracteres.

Ejemplo
```
SELECT LPAD('Page 1',15,'*.') "LPAD Ejemplo" FROM DUAL;
LPAD Ejemplo
---------------
*.*.*.*.*Page 1
```

LTRIM

Sintaxis
```
LTRIM(char [,set])
```

Descripción Elimina los caracteres que aparecen en *set* de la izquierda de *char*, hasta que encuentre uno que no esté en *set*.

Si no se indica [set] LTRIM elimina los espacios en blanco que haya a la izquierda en el char o string .

Ejemplo
```
SELECT LTRIM('xyxXxyLAST WORD','xy') "LTRIM Ejempl" FROM DUAL;
LTRIM exampl
------------
XxyLAST WORD
```

REPLACE

Sintaxis
```
REPLACE(char,search_string[,replacement_string])
```

Descripción Devuelve *char* donde cada ocurrencia de *search_string* es reemplazada con *replacement_string*. Si *replacement_string* se omite o es null, todas las ocurrencias de *search_string* son eliminadas.

Ejemplo
```
SELECT REPLACE('JACK and JUE','J','BL') "Changes" FROM DUAL;
Changes
--------------
BLACK and BLUE
```

RPAD

Sintaxis
```
RPAD(char1, n [,char2])
```

Descripción Es similar a LPAD pero por la derecha.

Ejemplo
```
SELECT RPAD('MORRISON',12,'ab') "RPAD Ejemplo" FROM DUAL;
RPAD Ejemplo
-----------------
MORRISONabab
```

RTRIM

Sintaxis
```
RTRIM(char [,set]
```

Descripción Es similar a LTRIM pero por la derecha.

Ejemplo
```
SELECT RTRIM('BROWNINGyxXxy','xy') "RTRIM e.g." FROM DUAL;
RTRIM e.g
-------------
BROWNINGyxX
```

SUBSTR

Sintaxis
```
SUBSTR(char, m [,n])
```

Descripción Devuelve un substring de *char*, empezando en la posición *m* y cogiendo *n* caracteres. Si *m* es negativo empieza por –m posiciones empezando por la derecha . Si *n* es 0 devuelve la cadena vacia.

Ejemplo
```
SELECT SUBSTR('EnUnLugarDeLaMancha',3,4)  FROM DUAL;
SELECT SUBSTR('EnUnLugarDeLaMancha',-8,8) FROM DUAL;
SELECT SUBSTR('EnUnLugarDeLaMancha',-8,3) FROM DUAL;
```

```
SELECT SUBSTR('EnUnLugarDeLaMancha',3,0) FROM DUAL;
```

UPPER

Sintaxis `UPPER(char)`

Descripción Devuelve *char*, pero con todas las letras en mayúscula.

Ejemplo
```
SELECT UPPER('Large') "Uppercase"  FROM DUAL;
Upper
-----
LARGE
```

- **Que devuelven un dato de tipo NUMBER.**

ASCII

Sintaxis `ASCII(char)`

Descripción Devuelve la representación numérica de *char (tabla ASCII)*.

Ejemplo
```
SELECT ASCII('Q') FROM DUAL;
 ASCII('Q')
----------
        81
```

INSTR

Sintaxis `INSTR (char1,char2 [,n[,m]])`

Descripción Devuelve la posición donde se encuentra *emésima* ocurrencia de *char2* dentro de char1 empezando por el carácter con posición *n*. Si n es negativa se busca desde la derecha.

Ejemplo 1
```
SELECT INSTR('CORPORATE FLOOR','OR',3,2)"Instring" FROM DUAL;
  Instring
----------
        14
```

Ejemplo 2
```
SELECT INSTR('CORPORATE FLOOR','OR',-3,2) "Reversed" FROM DUAL;
Reversed Instring
-----------------
                2
```

LENGTH

Sintaxis `LENGTH(char)`

Descripción Devuelve la longitud de *char*. Si *char* es nulo, devuelve null.

Ejemplo
```
SELECT LENGTH('CANDIDE') "Length" FROM DUAL;
Length
------
     7
```

- **De Fecha.**

Todas devuelven un dato de tipo Date, excepto MONTHS_BETWEEN que devuelve NUMBER.

ADD_MONTHS

Sintaxis `ADD_MONTHS(d,n)`

Descripción Devuelve la fecha *d* más *n* meses.

Ejemplo
```
SELECT TO_CHAR(ADD_MONTHS(hiredate,1),'DD-MON-YYYY') "Next
Month" FROM emp WHERE ename = 'SMITH';
Next Month
-----------
17-JAN-1981
```

LAST_DAY

Sintaxis `LAST_DAY(d)`

Descripción Devuelve el último día del mes de *d*.

Ejemplo 1
```
SELECT SYSDATE,LAST_DAY(SYSDATE) "Last",
LAST_DAY(SYSDATE) - SYSDATE "Days Left" FROM DUAL;
SYSDATE   Last       Days Left
--------- --------- ----------
23-OCT-97 31-OCT-97          8
```

MONTHS_BETWEEN

Sintaxis `MONTHS_BETWEEN(d1, d2)`

Descripción Devuelve el número de meses entre las fechas *d1* y *d2*.

Ejemplo
```
SELECT MONTHS_BETWEEN (TO_DATE('02-02-1995','MM-DD-YYYY'),
TO_DATE('01-01-1995','MM-DD-YYYY') ) "Months" FROM DUAL;
    Months
----------
1.03225806
```

NEXT_DAY

Sintaxis `NEXT_DAY(d, char)`

Descripción Devuelve la fecha que coincide con el primer dia de la semana indicado en *char*.

Ejemplo
```
SELECT NEXT_DAY('16-ENE-2009','LUNES') "Sig. Lunes" FROM DUAL;
Sig. Lunes
--------------------
19/01/09
```

ROUND

Sintaxis `ROUND(d[,fmt])`

Descripción Devuelve la fecha *d* redondeada a la unidad indicada en el formato *fmt*. Si omitimos *fmt*, *d* es redondeada al siguiente dia. *Ver tabla 1*.

Ejemplo
```
SELECT ROUND(TO_DATE('27-OCT-92'),'YEAR') "New Year" FROM
DUAL;
New Year
---------
01-JAN-93
```

SYSDATE

Descripción Devuelve fecha y hora actual.

Ejemplo
```
SELECT TO_CHAR(SYSDATE,'MM-DD-YYYY HH24:MI:SS')"NOW" FROM DUAL;
NOW
-------------------
01-01-2009 20:27:11
```

TRUNC

Sintaxis `TRUNC(d,[fmt])`

Descripción Devuelve la fecha *d* truncada a la unidad indicada en el formato *fmt*. Si omitimos *fmt*, *d* es redondeada al siguiente dia. *Ver tabla 1*.

Ejemplo
```
SELECT TRUNC(TO_DATE('01-OCT-09','DD-MON-YY'), 'YEAR')
  "New Year" FROM DUAL;
 New Year
---------
01-JAN-92
```

Tabla 1. ROUND y TRUNC para fechas.

Formato	Rounding or Truncating Unit
CC SCC	One greater than the first two digits of a four-digit year.
YYYY YEAR YYY YY Y	Year (rounds up on July 1)
Q	Quarter (rounds up on the sixteenth day of the second month of the quarter)
MONTH MON MM RM	Month (rounds up on the sixteenth day)
WW	Same day of the week as the first day of the year.
W	Same day of the week as the first day of the month.

`DDD`
`DD`
`J`
	Day

`DAY`
`DY`
`D`
	Starting day of the week

`HH`
`HH12`
`HH24`
	Hour

`MI`	Minute

- **Funciones de Conversión.**

TO_CHAR, conversión de fechas

Sintaxis `TO_CHAR(d [, fmt [, 'nlsparams'] ])`

Descripción Convierte un DATE *d* en un VARCHAR2, usando el formato *fmt*. . ***Ver tabla 1***.

Ejemplo
```
SELECT TO_CHAR(SYSDATE, 'DD/MM/YYYY') FROM DUAL;
```

TO_CHAR, conversión de números

Sintaxis `TO_CHAR(n [, fmt [, 'nlsparams'] ])`

Descripción Convierte un NUMBER *n* en un VARCHAR2, usando el formato *fmt*. . ***Ver tabla 2***.

Ejemplo
```
SELECT TO_CHAR(-1234,'L99G999D99MI') "Cuenta" FROM DUAL;
```

TO_DATE

Sintaxis `TO_DATE(char [, fmt [, 'nlsparams'] ])`

Descripción Convierte un VARCHAR2 o CHAR en un DATE, usando el formato *fmt*.. ***Ver tabla 3***.

Ejemplo
```
SELECT TO_DATE('01/01/2009','DD MM YY') "Cuenta" FROM DUAL;
```

TO_NUMBER

Sintaxis `TO_NUMBER(char [,fmt [, 'nlsparams'] ])`

Descripción Convierte un VARCHAR2 o CHAR en un NUMBER, usando el formato *fmt*.

Hay que tener en cuenta los símbolos utilizados por el sistema para distinguir los decimales de los miles y/o millones.

Ejemplo
```
SELECT TO_NUMBER('1.234,25', '9G999D99') "Numero" FROM DUAL;
```

Tabla 2 - Number Format Elements

Element	Example	Description
9	`9999`	Return value with the specified number of digits with a leading space if positive. Return value with the specified number of digits with a leading minus if negative. Leading zeros are blank, except for a zero value, which returns a zero for the integer part of the fixed-point number.
0	`0999` `9990`	Return leading zeros. Return trailing zeros.
$	`$9999`	Return value with a leading dollar sign.
B	`B9999`	Return blanks for the integer part of a fixed-point number when the integer part is zero (regardless of "0's in the format model).
MI	`9999MI`	Return negative value with a trailing minus sign "-". Return positive value with a trailing blank.

S	`S9999`
	
	`9999S`
		Return negative value with a leading minus sign "-".
		Return positive value with a leading plus sign "+".
		
		Return negative value with a trailing minus sign "-".
		Return positive value with a trailing plus sign "+".

PR	`9999PR`	Return negative value in <angle brackets>. Return positive value with a leading and trailing blank.
D	`99D99`	Return a decimal character (that is, a period ".") in the specified position.
G	`9G999`	Return a group separator in the position specified.
C	`C999`	Return the ISO currency symbol in the specified position.
L	`L999`	Return the local currency symbol in the specified position.
, (coma)	`9,999`	Return a comma in the specified position.
. (period)	`99.99`	Return a decimal point (that is, a period ".") in the specified position.
V	`999V99`	Return a value multiplied by 10^n (and if necessary, round it up), where *n* is the number of 9's after the "V".
EEEE	`9.9EEEE`	Return a value using in scientific notation.
RN rn	`RN`	Return a value as Roman numerals in uppercase. Return a value as Roman numerals in lowercase. Value can be an integer between 1 and 3999.
FM	`FM90.9`	Return a value with no leading or trailing blanks.

Ejemplos

number	'fmt'	Result
-1234567890	9999999999S	'1234567890-'
0	99.99	' .00'
+0.1	99.99	' 0.10'
-0.2	99.99	' -.20'
0	90.99	' 0.00'
+0.1	90.99	' 0.10'
-0.2	90.99	' -0.20'
0	9999	' 0'
1	9999	' 1'
0	B9999	' '
1	B9999	' 1'
0	B90.99	' '
+123.456	999.999	' 123.456'
-123.456	999.999	'-123.456'
+123.456	FM999.009	'123.456'
+123.456	9.9EEEE	' 1.2E+02'
+1E+123	9.9EEEE	' 1.0E+123'
+123.456	FM9.9EEEE	'1.23E+02'
+123.45	FM999.009	'123.45'
+123.0	FM999.009	'123.00'
+123.45	L999.99	' $123.45'
+123.45	FML99.99	'$123.45'
+1234567890	9999999999S	'1234567890+'

Tabla 3 Date Format Elements

Element	Specify in TO_DATE?	Meaning
- / , . ; : 'text'	Yes	Punctuation and quoted text is reproduced in the result.
AD A.D.	Yes	AD indicator with or without periods.
AM A.M.	Yes	Meridian indicator with or without periods.
BC B.C.	Yes	BC indicator with or without periods.
D	Yes	Day of week (1-7).
DAY	Yes	Name of day, padded with blanks to length of 9 characters.
DD	Yes	Day of month (1-31).
DDD	Yes	Day of year (1-366).
DY	Yes	Abbreviated name of day.
HH	Yes	Hour of day (1-12).
HH12	No	Hour of day (1-12).
HH24	Yes	Hour of day (0-23).
IW	No	Week of year (1-52 or 1-53) based on the ISO standard.
IYY IY I	No	Last 3, 2, or 1 digit(s) of ISO year.
IYYY	No	4-digit year based on the ISO standard.
J	Yes	Julian day; the number of days since January 1, 4712 BC. Number specified with 'J' must be integers.

MI	Yes	Minute (0-59).
MM	Yes	Month (01-12; JAN = 01)
MON	Yes	Abbreviated name of month.
MONTH	Yes	Name of month, padded with blanks to length of 9 characters.
PM P.M.	No	Meridian indicator with or without periods.
Q	No	Quarter of year (1, 2, 3, 4; JAN-MAR = 1)
RM	Yes	Roman numeral month (I-XII; JAN = I).
RR	Yes	Given a year with 2 digits, returns a year in the next century if the year is <50 and the last 2 digits of the current year are >=50; returns a year in the preceding century if the year is >=50 and the last 2 digits of the current year are <50.
RRRR	Yes	Round year. Accepts either 4-digit or 2-digit input. If 2-digit, provides the same return as RR. If you don't want this functionality, simply enter the 4-digit year.
SS	Yes	Second (0-59).
SSSSS	Yes	Seconds past midnight (0-86399).
WW	No	Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
W	No	Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
Y,YYY	Yes	Year with comma in this position.
YEAR SYEAR	No	Year, spelled out; "S" prefixes BC dates with "-".
YYYY SYYYY	Yes	4-digit year; "S" prefixes BC dates with "-".
YYY YY Y	Yes	Last 3, 2, or 1 digit(s) of year.

- **Otras Funciones**

DECODE

Sintaxis `DECODE(expr,valor1,rsdo1,valor2,rsdo2, ..., rsdo_else)`

Descripción Evalua una expresión `expr` y dependiendo del valor de `expr` nos devuelve un resultado.

Ejemplo
```
SELECT DECODE(TO_CHAR(SYSDATE,'Q'),'1','PRIMER','2','SEGUNDO',
'3','TERCER','CUARTO') || ' TRIMESTRE' AS Trimestre FROM DUAL;
```

GREATEST

Sintaxis `GREATEST(expr [,expr] ...)`

Descripción Devuelve el valor mayor de una lista de valores.

Ejemplo
```
SELECT GREATEST ('HARRY', 'HARRIOT', 'HAROLD') FROM DUAL;

Great
-----
HARRY
```

LEAST

Sintaxis `LEAST(expr [,expr] ...)`

Descripción Devuelve el valor menor de una lista de valores.

Ejemplo
```
SELECT LEAST (123,25,356,28) FROM DUAL;
```

NVL

Sintaxis `NVL(expr1, expr2)`

Descripción Si *expr1* es NULL, devuelve *expr2*; en caso contrario devuelve *expr*.

USER

Sintaxis `USER`

Descripción Devuelve el nombre del usuario actual.
```
SELECT USER, UID FROM DUAL;
```

- **Funciones de Grupo**

AVG

Sintaxis `AVG([DISTINCT|ALL] n)`

Descripción Devuelve la media de *n*.

COUNT

Sintaxis `COUNT({* | [DISTINCT|ALL] expr})`

Descripción Devuelve el número de filas de una consulta

MAX

Sintaxis `MAX([DISTINCT|ALL] expr)`

Descripción Devuelve el valor máximo de *expr*.

MIN

Sintaxis `MIN([DISTINCT|ALL] expr)`

Descripción Devuelve el valor mínimo de *expr*.

STDDEV

Sintaxis `STDDEV([DISTINCT|ALL] x)`

Descripción Devuelve la desviación standard de *x*.

SUM

Sintaxis `SUM([DISTINCT|ALL] n)`

Descripción Devuelve la suma de los valores de *n*.

Ejemplos:

```
CONNECT SYSTEM/MANAGER;

/* Creamos un usuario nuevo al que le asignaremos los objetos de este ejercicio
*/
CREATE USER BD_100 IDENTIFIED BY BD_100
DEFAULT TABLESPACE USER_DATA;

/* Le asignamos permisos de administrador */

GRANT DBA TO BD_100;

/* Nos conectamos con este usuario */

CONNECT BD_100/BD_100;

/* Creacion de las tablas */

CREATE TABLE Articulos (
       cArtFml varchar2 (4) NOT NULL,
       cArtCdg varchar2 (4) NOT NULL,
       cArtDsc varchar2 (30),
       nArtPrc number(10) NULL ,
       nArtExs number(10,2) NULL ,
CONSTRAINT PK_ARTICULOS PRIMARY KEY(cArtFml,cArtCdg) )
TABLESPACE USER_DATA;

CREATE TABLE Compras (
       dCmpFch date NOT NULL ,
        cArtFml varchar2 (10),
       cArtCdg varchar2 (10),
       nCmpUnd number(10) NOT NULL,
       nCmpPrc number(10, 2)NOT NULL)
TABLESPACE USER_DATA;

ALTER TABLE COMPRAS ADD
CONSTRAINT FK_COMPRAS FOREIGN KEY  ( cArtFml, cArtCdg )
REFERENCES ARTICULOS ( cArtFml, cArtCdg );

CREATE TABLE Familias (
       cArtFml varchar (10) NOT NULL,
       cFmlDsc varchar (30),
CONSTRAINT PK_FAMILIAS PRIMARY KEY (cArtFml))
TABLESPACE USER_DATA;

CREATE TABLE ROTURAS (
       dRtrFch date NOT NULL ,
        cArtFml varchar2 (10),
       cArtCdg varchar2 (10),
       nRtrNmr number(10) NOT NULL)
TABLESPACE USER_DATA;

ALTER TABLE ROTURAS ADD
CONSTRAINT FK_ROTURAS FOREIGN KEY  ( cArtFml, cArtCdg )
REFERENCES ARTICULOS ( cArtFml, cArtCdg );

CREATE TABLE VENTAS (
       dVntFch date NOT NULL ,
        cArtFml varchar2 (10),
       cArtCdg varchar2 (10),
       nVntUnd number(10) NOT NULL,
       nVntPrc number(10, 2)NOT NULL)
TABLESPACE USER_DATA;
```

```
ALTER TABLE VENTAS ADD
CONSTRAINT FK_VENTAS FOREIGN KEY  ( cArtFml, cArtCdg )
REFERENCES ARTICULOS ( cArtFml, cArtCdg );


ALTER TABLE Articulos ADD
       CONSTRAINT FK_Articulos_Familias
       FOREIGN KEY (cArtFml) REFERENCES Familias (cArtFml);


/* Insertamos registros */

Insert Into Familias Values ('F1','Discos Duros');
Insert Into Familias Values ('F2','Procesadores');
Insert Into Familias Values ('F3','Pantallas');

Insert Into Articulos Values ('F1','SEA1','HD Seagate 40 GB', 1, 1.11);
Insert Into Articulos Values ('F1','IBM1','HD IBM 80 GB', 2, 2.22);
Insert Into Articulos Values ('F1','IBM2','HD IBM 120 GB', 3, 3.33);
Insert Into Articulos Values ('F2','P4-1','Pentium 4 2400 Mh', 4, 4.44);
Insert Into Articulos Values ('F2','P4-2','Pentium 4 3000 Mh' ,5, 5);
Insert Into Articulos Values ('F2','P3-X','Pentium 3 Xeon', 6, 6);
Insert Into Articulos Values ('F3','TFT1','TFT LG 5100', 7,7);

Insert Into Compras   Values (SYSDATE,'F1','SEA1', 1, 1);
Insert Into Compras   Values (SYSDATE,'F1','IBM1',2,2);
Insert Into Compras   Values (SYSDATE,'F1','IBM2',3,3);
Insert Into Compras   Values (SYSDATE,'F2','P4-1',4,4);
Insert Into Compras   Values (SYSDATE,'F2','P4-2',5,5);
Insert Into Compras   Values (SYSDATE,'F3','TFT1',6,6);

Insert Into Ventas    Values (SYSDATE,'F1','IBM1',1,4);
Insert Into Ventas    Values (SYSDATE,'F2','P4-1',1,5);
Insert Into Ventas    Values (SYSDATE,'F2','P4-2',3,6);

Insert Into Roturas   Values (SYSDATE,'F3','TFT1',2);
```

Using SQL Arithmetic Operators

```
SELECT cArtDsc,nArtPrc * nArtExs AS Valor FROM Articulos;

SELECT cArtDsc,nArtPrc * nArtExs AS Valor FROM Articulos WHERE cArtFml = 'F1';
```

Using SQL Numeric Functions

```
SELECT cArtDsc, ROUND(nArtPrc * nArtExs ,1 ) AS Valor_1 FROM Articulos WHERE
cArtFml = 'F1';
SELECT cArtDsc, ROUND(nArtPrc * nArtExs ,-1 ) AS Valor_1 FROM Articulos WHERE
cArtFml = 'F1';
SELECT cArtDsc, TRUNC(nArtPrc * nArtExs ,0 ) AS Valor_1 FROM Articulos WHERE
cArtFml = 'F1';


SELECT cArtDsc, MOD(nArtPrc * nArtExs ,2) AS Valor_1 FROM Articulos WHERE cArtFml
= 'F1';
```

Using SQL Character Functions

```
SELECT cArtDsc, UPPER(cArtDsc),LOWER(cArtDsc),INITCAP(cArtDsc) FROM Articulos;

SELECT cArtDsc, RTRIM(cArtDsc,'GB') FROM Articulos;
```

```
SELECT cArtDsc, RPAD(cArtDsc,30,'-') || '*' FROM Articulos;

SELECT cArtDsc, LENGTH(cArtDsc) FROM Articulos;

SELECT cArtDsc, SUBSTR(cArtDsc,5,3) FROM Articulos;

SELECT cArtDsc, SUBSTR(cArtDsc,-5,3) FROM Articulos;

SELECT cArtDsc, SUBSTR(cArtDsc,-5, LENGTH(cArtDsc)  ) FROM Articulos;

SELECT cArtDsc, REPLACE(cArtDsc,'GB','TERABYTES') FROM Articulos;
```

Using SQL Date Functions

```
SELECT SYSDATE FROM DUAL;

SELECT CARTCDG,SYSDATE,DCMPFCH,TRUNC(MONTHS_BETWEEN(SYSDATE, DCMPFCH)) "Meses
Transcurridos" FROM Compras;

SELECT CARTCDG,SYSDATE,DCMPFCH,ADD_MONTHS(DCMPFCH,5) " Mas 5 Meses" FROM Compras;

SELECT CARTCDG,SYSDATE,DCMPFCH,LAST_DAY(DCMPFCH) FROM Compras;
```

Using the SQL Character Conversion Function

```
SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY AD') "Today" FROM DUAL;

SELECT TO_CHAR(SYSDATE, 'FMMonth DD YYYY') "Today" FROM DUAL;

SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "Now" FROM DUAL;

SELECT TO_CHAR(NARTPRC * NARTEXS,'$99,999.99') Stock FROM Articulos;
```

Using the SQL Number Conversion Function

```
SELECT TO_NUMBER('1234,99') + 500 FROM DUAL;

SELECT TO_NUMBER('11.200,34', '99G999D99') + 500 FROM DUAL;
```

Using SQL Date Conversion Functions

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY HH24:MI:SS';

SELECT TO_DATE('ENERO 15, 2006, 11:00 PM', 'Month dd, YYYY, HH:MI AM') FROM DUAL;

SELECT TO_DATE('27-OCT-98', 'DD-MON-RR') FROM DUAL;

SELECT TO_DATE('28-Nov-05 14:10:10', 'DD-Mon-YY HH24:MI:SS') FROM DUAL;

SELECT TO_DATE('ENERO 15, 2006, 11:00 AM', 'Month dd, YYYY, HH:MI AM')
FROM DUAL;
```

Using SQL Aggregate Functions

```
SELECT COUNT(*) FROM ARTICULOS;

SELECT COUNT(CARTFML) FROM ARTICULOS;

SELECT COUNT(DISTINCT CARTFML) FROM ARTICULOS;

SELECT COUNT(DISTINCT CARTFML) FROM ARTICULOS where cArtCdg LIKE 'S%';

SELECT CARTFML,COUNT(*) FROM ARTICULOS
GROUP BY CARTFML
ORDER BY CARTFML;
```

```
SELECT MIN(NARTPRC),MAX(NARTPRC),AVG(NARTPRC) FROM ARTICULOS;

SELECT MIN(NARTPRC),MAX(NARTPRC),AVG(NARTPRC) FROM ARTICULOS
GROUP BY CARTFML
ORDER BY CARTFML;

SELECT CARTFML,MIN(NARTPRC),MAX(NARTPRC),AVG(NARTPRC) FROM ARTICULOS
GROUP BY CARTFML
ORDER BY CARTFML;

SELECT CARTFML,MIN(NARTPRC),MAX(NARTPRC),AVG(NARTPRC) FROM ARTICULOS
GROUP BY CARTFML
HAVING MIN(NARTPRC) <= 4
ORDER BY MIN(NARTPRC) DESC;
```

Using the SQL NVL Function

```
SELECT CARTFML, NVL(NARTPRC,0) FROM ARTICULOS;

SELECT CARTFML, NVL2(NARTPRC, NARTPRC * 2, 1) FROM ARTICULOS;
```

Ejemplos

```
-- Granada, a xx de xxxxxxxx del xxxx, son las xx horas y xx minutos

SELECT 'GRANADA, '||TO_CHAR(SYSDATE,'DD')||' DE '|| TO_CHAR(SYSDATE,'MONTH')||
' DEL '|| TO_CHAR(SYSDATE,'YYYY')||', SON LAS '||TO_CHAR(SYSDATE,'HH24')||' HORAS
Y '||TO_CHAR(SYSDATE,'MI')||
' MINUTOS' AS FECHA
FROM DUAL;

-- Estamos en el xx Trimestre

SELECT ' Estamos en el ' ||
DECODE(TO_NUMBER(TO_CHAR(SYSDATE,'Q')),1,'PRIMER ',2,'SEGUNDO ',3,'TERCER
',4,'CUARTO ') || 'TRIMESTRE' AS TRIMESTRE
FROM DUAL;
```