

Índice

1. Expresiones aritméticas (* + - /)	2
2. Función NVL (para cálculos nulos)	5
3. Funciones aritméticas	6
ABS()	6
CEIL()	7
ROUND()	8
FLOOR()	10
MOD()	11
SIGN()	12
POWER()	13
SQRT()	14
TRUNC ()	15
4. Funciones alfanuméricas	17
ASCII ()	17
CHR ()	18
INITCAP ()	19
LENGTH ()	20
INSTR ()	21
LOWER ()	24
UPPER()	25
LTRIM ()	26
RTRIM ()	27
LPAD, RPAD, SUBSTR ...	28
5. Ejercicios propuestos	37

1. EXPRESIONES ARITMETICAS

Las operaciones aritméticas se realizan incluyendo en las sentencias SQL expresiones de este tipo compuestas por operadores aritméticos, constantes numéricas y columnas de la base de datos del mismo tipo. Los operadores que se pueden emplear son:

- Suma +
- Resta -
- Multiplicación *
- División /

Cuando se utilicen varios operadores aritméticos en una misma expresión se debe tener en cuenta lo siguiente:

- . La multiplicación y división tienen la misma prioridad.
- . La suma y la resta tienen la misma prioridad.
- . Cuando los operadores tienen la misma prioridad en una expresión, ésta se evalúa de izquierda a derecha.
- . La multiplicación y división tienen mayor prioridad que la suma y la resta, por lo que si una expresión está compuesta por varios operadores se ejecutarán antes las multiplicaciones y divisiones.
- . Se puede cambiar el nivel de prioridad de los operadores con el uso de paréntesis.
- . Se pueden seleccionar constantes de cualquier tipo en todas las consultas.

1.1 Calcular el salario anual de todos los empleados. Se supone que el valor de la columna salary es el salario mensual, y el número de pagas es 15.

```
SELECT last_name, salary, salary * 15, job_id  
FROM employee;
```

	LAST_NAME	SALARY	SALARY*15	JOB_ID
1	SMITH	800	12000	667
2	ALLEN	1600	24000	670
3	DOYLE	2850	42750	671
4	DENNIS	2750	41250	671
5	BAKER	2200	33000	671
6	WARD	1250	18750	670
7	PETERS	1250	18750	670
8	SHAW	1250	18750	670
9	DUNCAN	1250	18750	670
10	LANGE	1250	18750	670
11	JONES	2975	44625	671
12	ALBERTS	3000	45000	671
13	PORTER	1250	18750	670
14	LEWIS	1800	27000	668
15	MARTIN	1250	18750	670
16	SOMMERS	1850	27750	668
17	BLAKE	2850	42750	671
18	CLARK	2450	36750	671
19	SCOTT	3000	45000	669
20	WEST	1500	22500	670
21	FISHER	3000	45000	669
22	ROSS	1300	19500	670
23	KING	5000	75000	672
24	TURNER	1500	22500	670
25	ADAMS	1100	16500	667
26	JAMES	950	14250	667

Las expresiones aritméticas pueden aparecer seleccionadas en el **SELECT** o bien en la cláusula **WHERE** o en la cláusula **ORDER BY**.

1.2 Seleccionar los empleados que tengan un salario anual superior a 25000 dólares.

```
SELECT last_name, salary, salary * 15, job_id
FROM employee
WHERE salary * 15 > 25000;
```

	LAST_NAME	SALARY	SALARY*15	JOB_ID
1	DOYLE	2850	42750	671
2	DENNIS	2750	41250	671
3	BAKER	2200	33000	671
4	JONES	2975	44625	671
5	ALBERTS	3000	45000	671
6	LEWIS	1800	27000	668
7	SOMMERS	1850	27750	668
8	BLAKE	2850	42750	671
9	CLARK	2450	36750	671
10	SCOTT	3000	45000	669
11	FISHER	3000	45000	669
12	KING	5000	75000	672
13	FORD	3000	45000	669
14	ROBERTS	2875	43125	669

1.3 Calcular el total mensual a percibir por cada empleado, siendo éste la suma del salario y la comisión.

```
SELECT last_name, salary, commission, salary + commission
FROM employee;
```

	LAST_NAME	SALARY	COMMISSION	SALARY+COMMISSION
1	SMITH	800	(null)	(null)
2	ALLEN	1600	300	1900
3	DOYLE	2850	(null)	(null)
4	DENNIS	2750	(null)	(null)
5	BAKER	2200	(null)	(null)
6	WARD	1250	500	1750
7	PETERS	1250	300	1550
8	SHAW	1250	1200	2450
9	DUNCAN	1250	(null)	(null)
10	LANGE	1250	300	1550
11	JONES	2975	(null)	(null)
12	ALBERTS	3000	(null)	(null)
13	PORTER	1250	900	2150
14	LEWIS	1800	(null)	(null)
15	MARTIN	1250	1400	2650
16	SOMMERS	1850	(null)	(null)
17	BLAKE	2850	(null)	(null)
18	CLARK	2450	(null)	(null)
19	SCOTT	3000	(null)	(null)
20	WEST	1500	1000	2500
21	FISHER	3000	(null)	(null)
22	ROSS	1300	800	2100
23	KING	5000	(null)	(null)
24	TURNER	1500	0	1500
25	ADAMS	1100	(null)	(null)
26	JAMES	950	(null)	(null)

Se observa que aquellos empleados que no tienen comisión, es decir que es nula, reciben un total también nulo. Esto es debido a que el **resultado de cualquier expresión aritmética que contenga operandos nulos, es nulo**. Por lo tanto, al sumar el salario con una comisión nula, el resultado es nulo también.

2. FUNCIÓN NVL

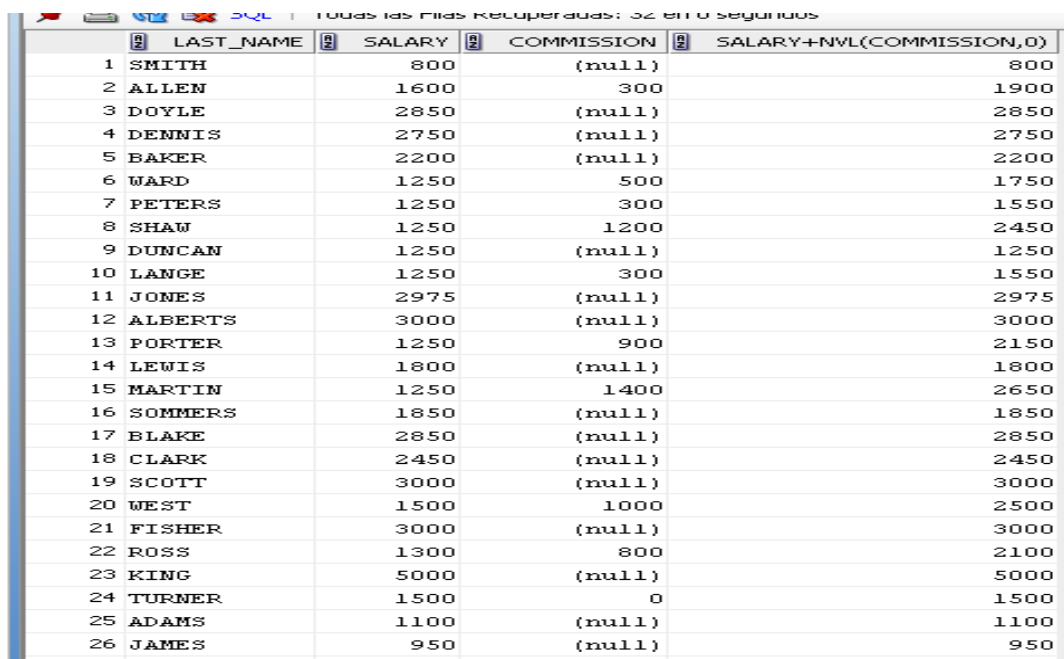
Para solucionar este problema SQL*PLUS dispone de la función NVL la cual permite sustituir temporalmente el valor nulo por otro cualquiera deseado. Su formato es:

NVL(columna,valor)

Cuando 'columna' tome el valor nulo, éste se sustituirá por 'valor'.

2.1 Calcular el total mensual a percibir por cada empleado, siendo éste la suma del salario y la comisión

```
SELECT last_name, salary, commission, salary + NVL(commission,0)
FROM employee;
```



SQL*PLUS window showing the results of the query: `SELECT last_name, salary, commission, salary + NVL(commission,0) FROM employee;`

	LAST_NAME	SALARY	COMMISSION	SALARY+NVL(COMMISSION,0)
1	SMITH	800	(null)	800
2	ALLEN	1600	300	1900
3	DOYLE	2850	(null)	2850
4	DENNIS	2750	(null)	2750
5	BAKER	2200	(null)	2200
6	WARD	1250	500	1750
7	PETERS	1250	300	1550
8	SHAW	1250	1200	2450
9	DUNCAN	1250	(null)	1250
10	LANGE	1250	300	1550
11	JONES	2975	(null)	2975
12	ALBERTS	3000	(null)	3000
13	PORTER	1250	900	2150
14	LEWIS	1800	(null)	1800
15	MARTIN	1250	1400	2650
16	SOMMERS	1850	(null)	1850
17	BLAKE	2850	(null)	2850
18	CLARK	2450	(null)	2450
19	SCOTT	3000	(null)	3000
20	WEST	1500	1000	2500
21	FISHER	3000	(null)	3000
22	ROSS	1300	800	2100
23	KING	5000	(null)	5000
24	TURNER	1500	0	1500
25	ADAMS	1100	(null)	1100
26	JAMES	950	(null)	950

Cuando la comisión sea nula la función NVL tomará el valor cero, y cuando sea distinta de nulo tomará ese mismo valor.

3. FUNCIONES ARITMETICAS

Los cálculos numéricos son fácilmente realizados utilizando las columnas de la base de datos con las expresiones y funciones aritméticas. Las funciones, al igual que las expresiones, pueden formar parte de la sentencia **SELECT**, bien como columna a seleccionar o como parte de las cláusulas **WHERE** y **ORDER BY**.

Los argumentos de estas funciones pueden ser tanto constantes como columnas de la tabla siempre que sean numéricas. Aunque a continuación especificamos 'columna' y 'número' como argumentos de las funciones, estos pueden sustituirse en cualquier momento por expresión, constante o columna.

Las funciones aritméticas, aplicables a columnas o expresiones, son:

ABS()

Esta función devuelve el valor absoluto del argumento.

```
SELECT ABS (salary-1500), salary, last_name
FROM employee;
```

	ABS(SALARY-1500)	SALARY	LAST_NAME
1	700	800	SMITH
2	100	1600	ALLEN
3	1350	2850	DOYLE
4	1250	2750	DENNIS
5	700	2200	BAKER
6	250	1250	WARD
7	250	1250	PETERS
8	250	1250	SHAW
9	250	1250	DUNCAN
10	250	1250	LANGE
11	1475	2975	JONES
12	1500	3000	ALBERTS
13	250	1250	PORTER
14	300	1800	LEWIS
15	250	1250	MARTIN
16	350	1850	SOMMERS
17	1350	2850	BLAKE
18	950	2450	CLARK
19	1500	3000	SCOTT
20	0	1500	WEST
21	1500	3000	FISHER
22	200	1300	ROSS
23	3500	5000	KING
24	0	1500	TURNER
25	400	1100	ADAMS
26	550	950	JAMES

CEIL()

Esta función devuelve el valor del entero más pequeño igual o mayor que el valor del argumento (redondeo por exceso)

```
SELECT CEIL (salary/30), salary, last_name  
FROM employee;
```

Todas las Filas Recuperadas: 32 en 0 s

	CEIL(SALARY/30)	SALARY	LAST_NAME
1	27	800	SMITH
2	54	1600	ALLEN
3	95	2850	DOYLE
4	92	2750	DENNIS
5	74	2200	BAKER
6	42	1250	WARD
7	42	1250	PETERS
8	42	1250	SHAW
9	42	1250	DUNCAN
10	42	1250	LANGE
11	100	2975	JONES
12	100	3000	ALBERTS
13	42	1250	PORTER
14	60	1800	LEWIS
15	42	1250	MARTIN
16	62	1850	SOMMERS
17	95	2850	BLAKE
18	82	2450	CLARK
19	100	3000	SCOTT
20	50	1500	WEST
21	100	3000	FISHER
22	44	1300	ROSS
23	167	5000	KING
24	50	1500	TURNER
25	37	1100	ADAMS
26	32	950	JAMES

ROUND ()

Redondea el valor del argumento a cero dígitos decimales.

```
SELECT ROUND (salary/30), salary, last_name  
FROM employee;
```

	ROUND(SALARY/30)	SALARY	LAST_NAME
1	27	800	SMITH
2	53	1600	ALLEN
3	95	2850	DOYLE
4	92	2750	DENNIS
5	73	2200	BAKER
6	42	1250	WARD
7	42	1250	PETERS
8	42	1250	SHAW
9	42	1250	DUNCAN
10	42	1250	LANGE
11	99	2975	JONES
12	100	3000	ALBERTS
13	42	1250	PORTER
14	60	1800	LEWIS
15	42	1250	MARTIN
16	62	1850	SOMMERS
17	95	2850	BLAKE
18	82	2450	CLARK
19	100	3000	SCOTT
20	50	1500	WEST
21	100	3000	FISHER
22	43	1300	ROSS
23	167	5000	KING
24	50	1500	TURNER
25	37	1100	ADAMS
26	32	950	JAMES

ROUND (,)

Redondea el valor del primer argumento a tantos dígitos decimales como exprese el segundo argumento. Si éste es negativo se redondea hacia la izquierda.

```
SELECT ROUND (salary/30,2), salary, last_name
FROM employee;
```

	ROUND(SALARY/30,2)	SALARY	LAST_NAME
1	26,67	800	SMITH
2	53,33	1600	ALLEN
3	95	2850	DOYLE
4	91,67	2750	DENNIS
5	73,33	2200	BAKER
6	41,67	1250	WARD
7	41,67	1250	PETERS
8	41,67	1250	SHAW
9	41,67	1250	DUNCAN
10	41,67	1250	LANGE
11	99,17	2975	JONES
12	100	3000	ALBERTS
13	41,67	1250	PORTER
14	60	1800	LEWIS
15	41,67	1250	MARTIN
16	61,67	1850	SOMMERS
17	95	2850	BLAKE
18	81,67	2450	CLARK
19	100	3000	SCOTT
20	50	1500	WEST
21	100	3000	FISHER
22	43,33	1300	ROSS
23	166,67	5000	KING
24	50	1500	TURNER
25	36,67	1100	ADAMS
26	31,67	950	JAMES

FLOOR ()

Esta función devuelve el valor del entero más grande igual o menor que el valor del argumento (redondeo por defecto).

```
SELECT FLOOR (salary/30), salary, last_name  
FROM employee;
```

	FLOOR(SALARY/30)	SALARY	LAST_NAME
1	26	800	SMITH
2	53	1600	ALLEN
3	95	2850	DOYLE
4	91	2750	DENNIS
5	73	2200	BAKER
6	41	1250	WARD
7	41	1250	PETERS
8	41	1250	SHAW
9	41	1250	DUNCAN
10	41	1250	LANGE
11	99	2975	JONES
12	100	3000	ALBERTS
13	41	1250	PORTER
14	60	1800	LEWIS
15	41	1250	MARTIN
16	61	1850	SOMMERS
17	95	2850	BLAKE
18	81	2450	CLARK
19	100	3000	SCOTT
20	50	1500	WEST
21	100	3000	FISHER
22	43	1300	ROSS
23	166	5000	KING
24	50	1500	TURNER
25	36	1100	ADAMS
26	31	950	JAMES

MOD (,)

Esta función devuelve el resto de dividir el primer argumento por el segundo.

```
SELECT MOD (salary,30), salary, last_name  
FROM employee;
```

	MOD(SALARY,30)	SALARY	LAST_NAME
1	20	800	SMITH
2	10	1600	ALLEN
3	0	2850	DOYLE
4	20	2750	DENNIS
5	10	2200	BAKER
6	20	1250	WARD
7	20	1250	PETERS
8	20	1250	SHAW
9	20	1250	DUNCAN
10	20	1250	LANGE
11	5	2975	JONES
12	0	3000	ALBERTS
13	20	1250	PORTER
14	0	1800	LEWIS
15	20	1250	MARTIN
16	20	1850	SOMMERS
17	0	2850	BLAKE
18	20	2450	CLARK
19	0	3000	SCOTT
20	0	1500	WEST
21	0	3000	FISHER
22	10	1300	ROSS
23	20	5000	KING
24	0	1500	TURNER
25	20	1100	ADAMS
26	20	950	JAMES

SIGN ()

Esta función devuelve 1 si el valor del argumento es mayor que cero, 0 si es cero, y -1 si es menor que cero.

```
SELECT SIGN (salary-1500), salary, last_name
FROM employee;
```

	SIGN(SALARY-1500)	SALARY	LAST_NAME
1	-1	800	SMITH
2	1	1600	ALLEN
3	1	2850	DOYLE
4	1	2750	DENNIS
5	1	2200	BAKER
6	-1	1250	WARD
7	-1	1250	PETERS
8	-1	1250	SHAW
9	-1	1250	DUNCAN
10	-1	1250	LANGE
11	1	2975	JONES
12	1	3000	ALBERTS
13	-1	1250	PORTER
14	1	1800	LEWIS
15	-1	1250	MARTIN
16	1	1850	SOMMERS
17	1	2850	BLAKE
18	1	2450	CLARK
19	1	3000	SCOTT
20	0	1500	WEST
21	1	3000	FISHER
22	-1	1300	ROSS
23	1	5000	KING
24	0	1500	TURNER
25	-1	1100	ADAMS
26	-1	950	JAMES

POWER ()

Esta función eleva el valor del primer argumento al exponente expresado por el segundo.

```
SELECT POWER (salary,2), salary, last_name
FROM employee;
```

	POWER(SALARY,2)	SALARY	LAST_NAME
1	640000	800	SMITH
2	2560000	1600	ALLEN
3	8122500	2850	DOYLE
4	7562500	2750	DENNIS
5	4840000	2200	BAKER
6	1562500	1250	WARD
7	1562500	1250	PETERS
8	1562500	1250	SHAW
9	1562500	1250	DUNCAN
10	1562500	1250	LANGE
11	8850625	2975	JONES
12	9000000	3000	ALBERTS
13	1562500	1250	PORTER
14	3240000	1800	LEWIS
15	1562500	1250	MARTIN
16	3422500	1850	SOMMERS
17	8122500	2850	BLAKE
18	6002500	2450	CLARK
19	9000000	3000	SCOTT
20	2250000	1500	WEST
21	9000000	3000	FISHER
22	1690000	1300	ROSS
23	25000000	5000	KING
24	2250000	1500	TURNER
25	1210000	1100	ADAMS
26	902500	950	JAMES

SQRT ()

Esta función devuelve el valor positivo de la raíz cuadrada del valor del argumento.

```
SELECT SQRT (salary), salary, last_name
FROM employee;
```

	SQRT(SALARY)	SALARY	LAST_NAME
1	28,28427124746190097603377448419396157139	800	SMITH
2	40	1600	ALLEN
3	53,38539126015655605405761982797855313141	2850	DOYLE
4	52,44044240850757734957267568399687992376	2750	DENNIS
5	46,90415759823429554565630113544466280588	2200	BAKER
6	35,35533905932737622004221810524245196424	1250	WARD
7	35,35533905932737622004221810524245196424	1250	PETERS
8	35,35533905932737622004221810524245196424	1250	SHAW
9	35,35533905932737622004221810524245196424	1250	DUNCAN
10	35,35533905932737622004221810524245196424	1250	LANGE
11	54,54356057317857205751077243686450936402	2975	JONES
12	54,77225575051661134569697828008021339527	3000	ALBERTS
13	35,35533905932737622004221810524245196424	1250	PORTER
14	42,42640687119285146405066172629094235709	1800	LEWIS
15	35,35533905932737622004221810524245196424	1250	MARTIN
16	43,01162633521313385864736767524856816013	1850	SOMMERS
17	53,38539126015655605405761982797855313141	2850	BLAKE
18	49,49747468305832670805910534733943274993	2450	CLARK
19	54,77225575051661134569697828008021339527	3000	SCOTT
20	38,72983346207416885179265399782399610832	1500	WEST
21	54,77225575051661134569697828008021339527	3000	FISHER
22	36,05551275463989293119221267470495946251	1300	ROSS
23	70,71067811865475244008443621048490392848	5000	KING
24	38,72983346207416885179265399782399610832	1500	TURNER
25	33,16624790355399849114932736670686683927	1100	ADAMS
26	30,82207001484488225125096190727122112617	950	JAMES

TRUNC ()

Trunca el valor del argumento a cero dígitos decimales.

```
SELECT TRUNC (salary/30), salary, last_name  
FROM employee;
```

	TRUNC(SALARY/30)	SALARY	LAST_NAME
1	26	800	SMITH
2	53	1600	ALLEN
3	95	2850	DOYLE
4	91	2750	DENNIS
5	73	2200	BAKER
6	41	1250	WARD
7	41	1250	PETERS
8	41	1250	SHAW
9	41	1250	DUNCAN
10	41	1250	LANGE
11	99	2975	JONES
12	100	3000	ALBERTS
13	41	1250	PORTER
14	60	1800	LEWIS
15	41	1250	MARTIN
16	61	1850	SOMMERS
17	95	2850	BLAKE
18	81	2450	CLARK
19	100	3000	SCOTT
20	50	1500	WEST
21	100	3000	FISHER
22	43	1300	ROSS
23	166	5000	KING
24	50	1500	TURNER
25	36	1100	ADAMS
26	31	950	JAMES

TRUNC (,)

Trunca el primer argumento a tantos dígitos decimales como exprese el segundo. Si éste es negativo se trunca hacia la izquierda.

```
SELECT TRUNC (salary/30,2), salary, last_name
FROM employee;
```

	TRUNC(SALARY/30,2)	SALARY	LAST_NAME
1	26,66	800	SMITH
2	53,33	1600	ALLEN
3	95	2850	DOYLE
4	91,66	2750	DENNIS
5	73,33	2200	BAKER
6	41,66	1250	WARD
7	41,66	1250	PETERS
8	41,66	1250	SHAW
9	41,66	1250	DUNCAN
10	41,66	1250	LANGE
11	99,16	2975	JONES
12	100	3000	ALBERTS
13	41,66	1250	PORTER
14	60	1800	LEWIS
15	41,66	1250	MARTIN
16	61,66	1850	SOMMERS
17	95	2850	BLAKE
18	81,66	2450	CLARK
19	100	3000	SCOTT
20	50	1500	WEST
21	100	3000	FISHER
22	43,33	1300	ROSS
23	166,66	5000	KING
24	50	1500	TURNER
25	36,66	1100	ADAMS
26	31,66	950	JAMES

4. FUNCIONES ALFANUMERICAS

También se dispone de una serie de funciones tipo carácter para facilitar el uso de las columnas con este tipo de datos. Al igual que las numéricas pueden constar de columnas o constantes tipo carácter.

A estas funciones se les pueden pasar argumentos tanto numéricos como carácter. Cualquier argumento carácter puede ser o bien una columna de tipo carácter de la tabla, o bien una cadena de caracteres.

Las funciones de tipo carácter que podemos utilizar son:

ASCII ()

Esta función devuelve el número ASCII que representa el primer carácter del valor del argumento (como cadena de caracteres).

```
SELECT last_name, ASCII (last_name), job_id
FROM employee;
```

	LAST_NAME	ASCII(LAST_NAME)	JOB_ID
1	SMITH	83	667
2	ALLEN	65	670
3	DOYLE	68	671
4	DENNIS	68	671
5	BAKER	66	671
6	WARD	87	670
7	PETERS	80	670
8	SHAW	83	670
9	DUNCAN	68	670
10	LANGE	76	670
11	JONES	74	671
12	ALBERTS	65	671
13	PORTER	80	670
14	LEWIS	76	668
15	MARTIN	77	670
16	SOMMERS	83	668
17	BLAKE	66	671
18	CLARK	67	671
19	SCOTT	83	669
20	WEST	87	670
21	FISHER	70	669
22	ROSS	82	670
23	KING	75	672
24	TURNER	84	670
25	ADAMS	65	667
26	JAMES	74	667

CHR ()

Esta función devuelve el carácter ASCII que representa el argumento numérico.

```
SELECT last_name, CHR(department_id), job_id
FROM employee;
```

	LAST_NAME	CHR(DEPARTMENT_ID)	DEPARTMENT_ID	JOB_ID
1	SMITH	□	20	667
2	ALLEN	□	30	670
3	DOYLE		13	671
4	DENNIS	□	23	671
5	BAKER	□	14	671
6	WARD	□	30	670
7	PETERS		13	670
8	SHAW		13	670
9	DUNCAN	□	23	670
10	LANGE	□	23	670
11	JONES	□	20	671
12	ALBERTS	□	12	671
13	PORTER		13	670
14	LEWIS	□	24	668
15	MARTIN	□	30	670
16	SOMMERS	"	34	668
17	BLAKE	□	30	671
18	CLARK		10	671
19	SCOTT	□	20	669
20	WEST	□	23	670
21	FISHER	□	12	669
22	ROSS	+	43	670
23	KING		10	672
24	TURNER	□	30	670
25	ADAMS	□	20	667
26	JAMES	□	30	667

INITCAP ()

Esta función transforma el argumento, una cadena de caracteres, en la misma cadena empezando con mayúsculas todas las palabras.

```
SELECT last_name, INITCAP (last_name), job_id
FROM employee;
```

	LAST_NAME	INITCAP(LAST_NAME)	JOB_ID
1	SMITH	Smith	667
2	ALLEN	Allen	670
3	DOYLE	Doyle	671
4	DENNIS	Dennis	671
5	BAKER	Baker	671
6	WARD	Ward	670
7	PETERS	Peters	670
8	SHAW	Shaw	670
9	DUNCAN	Duncan	670
10	LANGE	Lange	670
11	JONES	Jones	671
12	ALBERTS	Alberts	671
13	PORTER	Porter	670
14	LEWIS	Lewis	668
15	MARTIN	Martin	670
16	SOMMERS	Sommers	668
17	BLAKE	Blake	671
18	CLARK	Clark	671
19	SCOTT	Scott	669
20	WEST	West	670
21	FISHER	Fisher	669
22	ROSS	Ross	670
23	KING	King	672
24	TURNER	Turner	670
25	ADAMS	Adams	667
26	JAMES	James	667

LENGTH ()

Esta función devuelve el número de caracteres que contiene el argumento (cadena de caracteres).

```
SELECT last_name, LENGTH (last_name), job_id  
FROM employee;
```

	LAST_NAME	LENGTH(LAST_NAME)	JOB_ID
1	SMITH	5	667
2	ALLEN	5	670
3	DOYLE	5	671
4	DENNIS	6	671
5	BAKER	5	671
6	WARD	4	670
7	PETERS	6	670
8	SHAW	4	670
9	DUNCAN	6	670
10	LANGE	5	670
11	JONES	5	671
12	ALBERTS	7	671
13	PORTER	6	670
14	LEWIS	5	668
15	MARTIN	6	670
16	SOMMERS	7	668
17	BLAKE	5	671
18	CLARK	5	671
19	SCOTT	5	669
20	WEST	4	670
21	FISHER	6	669
22	ROSS	4	670
23	KING	4	672
24	TURNER	6	670
25	ADAMS	5	667
26	JAMES	5	667

INSTR (,,)

Esta función devuelve la posición que ocupa la primera ocurrencia del segundo argumento en el primero. Si no se encuentra, la función retorna un cero. Si se especifica un tercer argumento éste determina la posición a partir de la cual se busca en el primer argumento. Si se especifica un cuarto argumento devuelve la posición que indica la ocurrencia que indica dicho argumento.

Ejemplos:

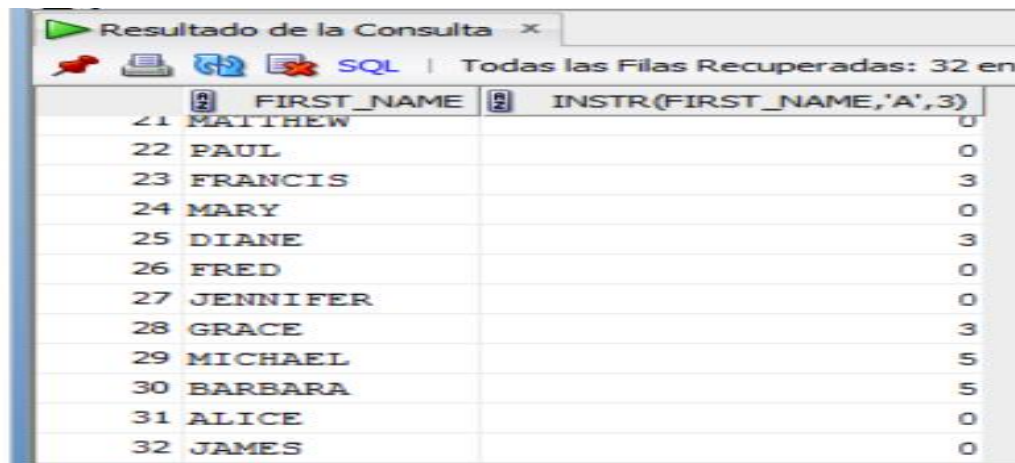
Visualizar la posición que ocupa la primera A de cada uno de los nombres de los empleados

```
SELECT first_name, INSTR (first_name, 'A')  
FROM employee;
```

	FIRST_NAME	INSTR(FIRST_NAME,'A')
21	MATTHEW	2
22	PAUL	2
23	FRANCIS	3
24	MARY	2
25	DIANE	3
26	FRED	0
27	JENNIFER	0
28	GRACE	3
29	MICHAEL	5
30	BARBARA	2
31	ALICE	1
32	JAMES	2

Visualizar la posición que ocupa la primera A de cada uno de los nombres de los empleados a partir de la 3 posición

```
SELECT first_name, INSTR (first_name, 'A',3)  
FROM employee;
```



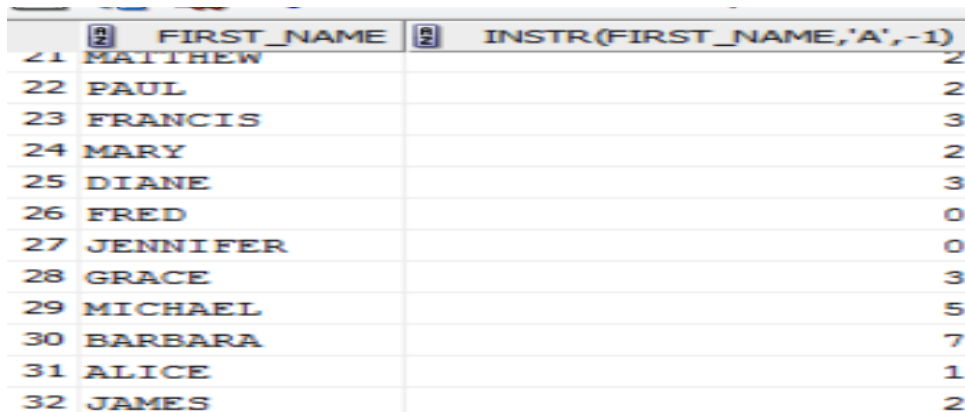
Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 32 en

	FIRST_NAME	INSTR(FIRST_NAME,'A',3)
21	MATTHEW	0
22	PAUL	0
23	FRANCIS	3
24	MARY	0
25	DIANE	3
26	FRED	0
27	JENNIFER	0
28	GRACE	3
29	MICHAEL	5
30	BARBARA	5
31	ALICE	0
32	JAMES	0

Visualizar la posición que ocupa la primera A de cada uno de los nombres de los empleados de derecha a izquierda

```
SELECT first_name, INSTR (first_name, 'A',-1)  
FROM employee;
```



	FIRST_NAME	INSTR(FIRST_NAME,'A',-1)
21	MATTHEW	2
22	PAUL	2
23	FRANCIS	3
24	MARY	2
25	DIANE	3
26	FRED	0
27	JENNIFER	0
28	GRACE	3
29	MICHAEL	5
30	BARBARA	7
31	ALICE	1
32	JAMES	2

Visualizar la posición que ocupa la primera A de cada uno de los nombres de los empleados de derecha a izquierda empezando por la 2ª posición

```
SELECT first_name, INSTR (first_name, 'A',-2)  
FROM employee;
```

	FIRST_NAME	INSTR(FIRST_NAME,'A',-2)
21	MATTHEW	2
22	PAUL	2
23	FRANCIS	3
24	MARY	2
25	DIANE	3
26	FRED	0
27	JENNIFER	0
28	GRACE	3
29	MICHAEL	5
30	BARBARA	5
31	ALICE	1
32	JAMES	2

Visualizar la posición que ocupa la segunda A de cada uno de los nombres de los empleados empezando de izquierda a derecha.

```
SELECT first_name, INSTR (first_name, 'A',1,2)  
FROM employee;
```

	FIRST_NAME	INSTR(FIRST_NAME,'A',1,2)
21	MATTHEW	0
22	PAUL	0
23	FRANCIS	0
24	MARY	0
25	DIANE	0
26	FRED	0
27	JENNIFER	0
28	GRACE	0
29	MICHAEL	0
30	BARBARA	5
31	ALICE	0
32	JAMES	0

LOWER ()

Esta función convierte el argumento en el mismo pero con letras minúsculas.

```
SELECT last_name, LOWER (last_name), job_id
FROM employee;
```

	R2 LAST_NAME	R2 LOWER(LAST_NAME)	R2 JOB_ID
1	SMITH	smith	667
2	ALLEN	allen	670
3	DOYLE	doyle	671
4	DENNIS	dennis	671
5	BAKER	baker	671
6	WARD	ward	670
7	PETERS	peters	670
8	SHAW	shaw	670
9	DUNCAN	duncan	670
10	LANGE	lange	670
11	JONES	jones	671
12	ALBERTS	alberts	671
13	PORTER	porter	670
14	LEWIS	lewis	668
15	MARTIN	martin	670
16	SOMMERS	sommers	668
17	BLAKE	blake	671
18	CLARK	clark	671
19	SCOTT	scott	669
20	WEST	west	670
21	FISHER	fisher	669
22	ROSS	ross	670
23	KING	king	672
24	TURNER	turner	670
25	ADAMS	adams	667
26	JAMES	james	667

UPPER ()

Esta función convierte el argumento en el mismo pero con letras mayúsculas.

```
SELECT last_name, UPPER (last_name), job_id
FROM employee;
```

	LAST_NAME	UPPER(LAST_NAME)	JOB_ID
1	SMITH	SMITH	667
2	ALLEN	ALLEN	670
3	DOYLE	DOYLE	671
4	DENNIS	DENNIS	671
5	BAKER	BAKER	671
6	WARD	WARD	670
7	PETERS	PETERS	670
8	SHAW	SHAW	670
9	DUNCAN	DUNCAN	670
10	LANGE	LANGE	670
11	JONES	JONES	671
12	ALBERTS	ALBERTS	671
13	PORTER	PORTER	670
14	LEWIS	LEWIS	668
15	MARTIN	MARTIN	670
16	SOMMERS	SOMMERS	668
17	BLAKE	BLAKE	671
18	CLARK	CLARK	671
19	SCOTT	SCOTT	669
20	WEST	WEST	670
21	FISHER	FISHER	669
22	ROSS	ROSS	670
23	KING	KING	672
24	TURNER	TURNER	670
25	ADAMS	ADAMS	667
26	JAMES	JAMES	667

LTRIM (,)

Esta función borra por la izquierda en el primer argumento todos los caracteres hasta encontrar uno que no esté en el conjunto de caracteres formado por el segundo argumento.

```
SELECT last_name, job_id, LTRIM (job_id, 67)
FROM employee;
```

	LAST_NAME	JOB_ID	LTRIM(JOB_ID,67)
7	PETERS	670 0	
8	SHAW	670 0	
9	DUNCAN	670 0	
10	LANGE	670 0	
11	JONES	671 1	
12	ALBERTS	671 1	
13	PORTER	670 0	
14	LEWIS	668 8	
15	MARTIN	670 0	
16	SOMMERS	668 8	
17	BLAKE	671 1	
18	CLARK	671 1	
19	SCOTT	669 9	
20	WEST	670 0	
21	FISHER	669 9	
22	ROSS	670 0	
23	KING	672 2	
24	TURNER	670 0	
25	ADAMS	667 (null)	
26	JAMES	667 (null)	
27	FORD	669 9	
28	ROBERTS	669 9	
29	DOUGLAS	667 (null)	
30	MILLER	667 (null)	
31	JENSEN	667 (null)	
32	MURRAY	667 (null)	

RTRIM (,)

Esta función borra por la derecha en el primer argumento todos los caracteres hasta encontrar uno que no esté en el conjunto de caracteres formado por el segundo argumento.

```
SELECT last_name, job_id, RTRIM (last_name, 'TH')  
FROM employee;
```

	LAST_NAME	JOB_ID	RTRIM(LAST_NAME, 'TH')
1	SMITH	667	SMI
2	ALLEN	670	ALLEN
3	DOYLE	671	DOYLE
4	DENNIS	671	DENNIS
5	BAKER	671	BAKER
6	WARD	670	WARD
7	PETERS	670	PETERS
8	SHAW	670	SHAW
9	DUNCAN	670	DUNCAN
10	LANGE	670	LANGE
11	JONES	671	JONES
12	ALBERTS	671	ALBERTS
13	PORTER	670	PORTER
14	LEWIS	668	LEWIS
15	MARTIN	670	MARTIN
16	SOMMERS	668	SOMMERS
17	BLAKE	671	BLAKE
18	CLARK	671	CLARK
19	SCOTT	669	SCO
20	WEST	670	WES
21	FISHER	669	FISHER
22	ROSS	670	ROSS
23	KING	672	KING
24	TURNER	670	TURNER
25	ADAMS	667	ADAMS
26	JAMES	667	JAMES

LPAD (,,)

Esta función añade en el primer argumento por la izquierda todos los caracteres blancos necesarios para que la longitud resultante del mismo sea la especificada como segundo argumento. Si se especifica un tercer argumento entonces en lugar de rellenar con blancos se rellenará con los caracteres de ese argumento.

```
SELECT last_name, job_id, LPAD (job_id, 25)
FROM employee;
```

	R2	LAST_NAME	R2	JOB_ID	R2	LPAD(JOB_ID,25)
1		SMITH		667		667
2		ALLEN		670		670
3		DOYLE		671		671
4		DENNIS		671		671
5		BAKER		671		671
6		WARD		670		670
7		PETERS		670		670
8		SHAW		670		670
9		DUNCAN		670		670
10		LANGE		670		670
11		JONES		671		671
12		ALBERTS		671		671
13		PORTER		670		670
14		LEWIS		668		668
15		MARTIN		670		670
16		SOMMERS		668		668
17		BLAKE		671		671
18		CLARK		671		671
19		SCOTT		669		669
20		WEST		670		670
21		FISHER		669		669
22		ROSS		670		670
23		KING		672		672
24		TURNER		670		670
25		ADAMS		667		667
26		JAMES		667		667

RPAD (,,)

Esta función añade en el primer argumento por la derecha todos los caracteres blancos necesarios para que la longitud resultante del mismo sea la especificada como segundo argumento. Si se especifica un tercer argumento entonces en lugar de rellenar con blancos se rellenará con los caracteres de ese argumento.

```
SELECT last_name, job_id, RPAD (job_id, 25)
FROM employee;
```

	LAST_NAME	JOB_ID	RPAD(JOB_ID,25)
1	SMITH	667	667
2	ALLEN	670	670
3	DOYLE	671	671
4	DENNIS	671	671
5	BAKER	671	671
6	WARD	670	670
7	PETERS	670	670
8	SHAW	670	670
9	DUNCAN	670	670
10	LANGE	670	670
11	JONES	671	671
12	ALBERTS	671	671
13	PORTER	670	670
14	LEWIS	668	668
15	MARTIN	670	670
16	SOMMERS	668	668
17	BLAKE	671	671
18	CLARK	671	671
19	SCOTT	669	669
20	WEST	670	670
21	FISHER	669	669
22	ROSS	670	670
23	KING	672	672
24	TURNER	670	670
25	ADAMS	667	667
26	JAMES	667	667

SUBSTR (,,)

Esta función devuelve la subcadena del primer argumento, desde la posición que indica el segundo argumento, tantas posiciones como indica el tercer argumento. Si no se especifica un tercer argumento se entenderá que se desea la subcadena hasta el final.

Ejemplos:

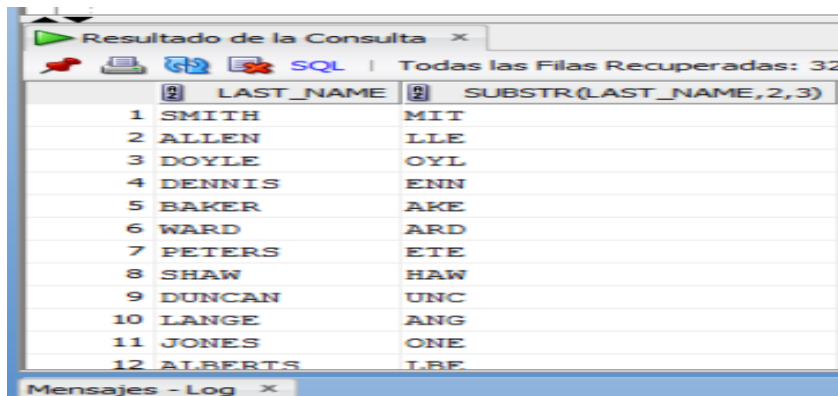
Visualizar los apellidos desde la 3ª posición hasta el final

```
SELECT last_name, job_id, SUBSTR(last_name, 3)
from employee;
```

	R 2	LAST_NAME	R 2	JOB_ID	R 2	SUBSTR(LAST_NAME,3)
1		SMITH		667		ITH
2		ALLEN		670		LEN
3		DOYLE		671		YLE
4		DENNIS		671		NNIS
5		BAKER		671		KER
6		WARD		670		RD
7		PETERS		670		TERS
8		SHAW		670		AW
9		DUNCAN		670		NCAN
10		LANGE		670		NGE
11		JONES		671		NES
12		ALBERTS		671		BERTS
13		PORTER		670		RTER
14		LEWIS		668		WIS
15		MARTIN		670		RTIN
16		SOMMERS		668		MMERS
17		BLAKE		671		AKE
18		CLARK		671		ARK
19		SCOTT		669		OTT
20		WEST		670		ST
21		FISHER		669		SHER
22		ROSS		670		SS
23		KING		672		NG
24		TURNER		670		RNER
25		ADAMS		667		AMS
26		JAMES		667		MES

Visualizar los caracteres del 2 al 4 de los apellidos de los empleados

```
SELECT last_name, SUBSTR(last_name, 2,3)
from employee
```



Resultado de la Consulta x

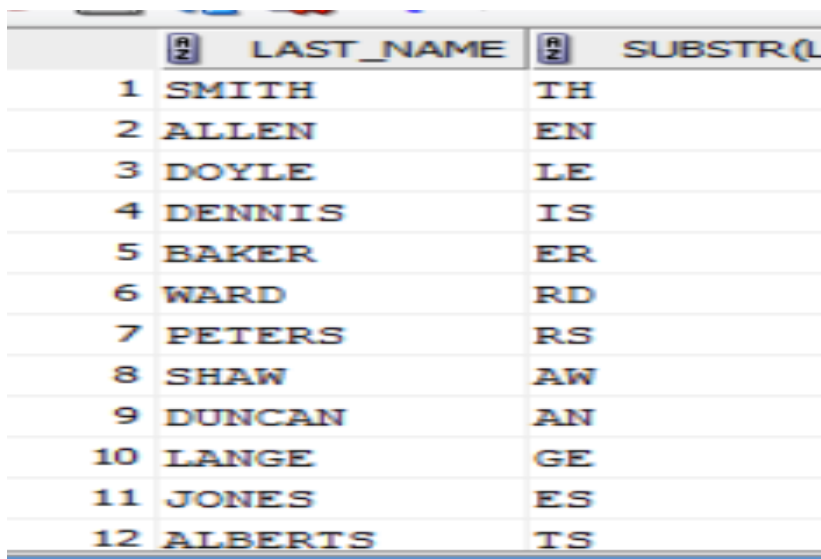
Todas las Filas Recuperadas: 32

	LAST_NAME	SUBSTR(LAST_NAME,2,3)
1	SMITH	MIT
2	ALLEN	LLE
3	DOYLE	OYL
4	DENNIS	ENN
5	BAKER	AKE
6	WARD	ARD
7	PETERS	ETE
8	SHAW	HAW
9	DUNCAN	UNC
10	LANGE	ANG
11	JONES	ONE
12	ALBERTS	LBE

Mensajes - Log x

Visualizar los dos últimos caracteres de los apellidos de los empleados

```
SELECT last_name, SUBSTR(last_name,length(last_name)-1,2)
from employee;
```



	LAST_NAME	SUBSTR(L
1	SMITH	TH
2	ALLEN	EN
3	DOYLE	LE
4	DENNIS	IS
5	BAKER	ER
6	WARD	RD
7	PETERS	RS
8	SHAW	AW
9	DUNCAN	AN
10	LANGE	GE
11	JONES	ES
12	ALBERTS	TS

TRANSLATE (,,)

Esta función sustituye en el primer argumento cada ocurrencia del segundo argumento por el tercero.

```
SELECT last_name,job_id,TRANSLATE(job_id,6,7)
FROM employee;
```

	LAST_NAME	JOB_ID	TRANSLATE(JOB_ID,6,7)
1	SMITH	667	777
2	ALLEN	670	770
3	DOYLE	671	771
4	DENNIS	671	771
5	BAKER	671	771
6	WARD	670	770
7	PETERS	670	770
8	SHAW	670	770
9	DUNCAN	670	770
10	LANGE	670	770
11	JONES	671	771
12	ALBERTS	671	771
13	PORTER	670	770
14	LEWIS	668	778
15	MARTIN	670	770
16	SOMMERS	668	778
17	BLAKE	671	771
18	CLARK	671	771
19	SCOTT	669	779
20	WEST	670	770
21	FISHER	669	779
22	ROSS	670	770
23	KING	672	772
24	TURNER	670	770
25	ADAMS	667	777
26	JAMES	667	777

||

Operador de concatenación. Añade el segundo argumento a la derecha del primero.

```
SELECT last_name, job_id, last_name || ',' || job_id
FROM employee;
```

	LAST_NAME	JOB_ID	LAST_NAME ',' JOB_ID
1	SMITH	667	SMITH,667
2	ALLEN	670	ALLEN,670
3	DOYLE	671	DOYLE,671
4	DENNIS	671	DENNIS,671
5	BAKER	671	BAKER,671
6	WARD	670	WARD,670
7	PETERS	670	PETERS,670
8	SHAW	670	SHAW,670
9	DUNCAN	670	DUNCAN,670
10	LANGE	670	LANGE,670
11	JONES	671	JONES,671
12	ALBERTS	671	ALBERTS,671
13	PORTER	670	PORTER,670
14	LEWIS	668	LEWIS,668
15	MARTIN	670	MARTIN,670
16	SOMMERS	668	SOMMERS,668
17	BLAKE	671	BLAKE,671
18	CLARK	671	CLARK,671
19	SCOTT	669	SCOTT,669
20	WEST	670	WEST,670
21	FISHER	669	FISHER,669
22	ROSS	670	ROSS,670
23	KING	672	KING,672
24	TURNER	670	TURNER,670
25	ADAMS	667	ADAMS,667
26	JAMES	667	JAMES,667

TO_NUMBER ()

Esta función convierte el argumento de tipo alfanumérico en el correspondiente numérico, siempre que los caracteres del argumento sean caracteres numéricos.

```
SELECT last_name , TO_NUMBER ('123')
FROM employee;
```

	LAST_NAME	TO_NUMBER('123')
1	SMITH	123
2	ALLEN	123
3	DOYLE	123
4	DENNIS	123
5	BAKER	123
6	WARD	123
7	PETERS	123
8	SHAW	123
9	DUNCAN	123
10	LANGE	123
11	JONES	123
12	ALBERTS	123
13	PORTER	123
14	LEWIS	123
15	MARTIN	123
16	SOMMERS	123
17	BLAKE	123
18	CLARK	123
19	SCOTT	123
20	WEST	123
21	FISHER	123
22	ROSS	123
23	KING	123
24	TURNER	123
25	ADAMS	123
26	JAMES	123

TO_CHAR

Esta función convierte el argumento de tipo numérico en la correspondiente cadena alfanumérica.

```
SELECT last_name , department_id, TO_CHAR (department_id)
FROM employee;
```

	LAST_NAME	DEPARTMENT_ID	TO_CHAR(DEPARTMENT_ID)
1	SMITH	20	20
2	ALLEN	30	30
3	DOYLE	13	13
4	DENNIS	23	23
5	BAKER	14	14
6	WARD	30	30
7	PETERS	13	13
8	SHAW	13	13
9	DUNCAN	23	23
10	LANGE	23	23
11	JONES	20	20
12	ALBERTS	12	12
13	PORTER	13	13
14	LEWIS	24	24
15	MARTIN	30	30
16	SOMMERS	34	34
17	BLAKE	30	30
18	CLARK	10	10
19	SCOTT	20	20
20	WEST	23	23
21	FISHER	12	12
22	ROSS	43	43
23	KING	10	10
24	TURNER	30	30
25	ADAMS	20	20
26	JAMES	30	30

GREATEST (,,...)

Esta función devuelve el mayor de sus argumentos según criterios v alfabéticos.

```
SELECT GREATEST ('rosa', 'rosae', 'rosarum', 'rosi')
FROM dual;
```

	GREATEST('ROSA','ROSAE','ROSARUM','ROSI')
1	rosi

(PARA HACER UNA CONSULTA GENERICA A ORACLE SE UTILIZA ESTA TABLA)

LEAST (,,...)

Esta función devuelve el menor de sus argumentos.

```
SELECT salary, commission, LEAST (salary, commission)
FROM employee;
```

Todas las Filas Recuperadas: 32 en 0 segundos

	SALARY	COMMISSION	LEAST(SALARY,COMMISSION)
1	800	(null)	(null)
2	1600	300	300
3	2850	(null)	(null)
4	2750	(null)	(null)
5	2200	(null)	(null)
6	1250	500	500
7	1250	300	300
8	1250	1200	1200
9	1250	(null)	(null)
10	1250	300	300
11	2975	(null)	(null)
12	3000	(null)	(null)
13	1250	900	900
14	1800	(null)	(null)
15	1250	1400	1250
16	1850	(null)	(null)
17	2850	(null)	(null)
18	2450	(null)	(null)
19	3000	(null)	(null)
20	1500	1000	1000
21	3000	(null)	(null)
22	1300	800	800
23	5000	(null)	(null)
24	1500	0	0
25	1100	(null)	(null)
26	950	(null)	(null)

DECODE (columna, valor1, valor11, valor2, valor22, ... , valorn, valorn1, otro)

Si el primer argumento tiene el valor 'valor1' esta función devuelve 'valor11',
 si tiene 'valor2' devuelve 'valor22',...,
 si tiene 'valorn' devuelve 'valorn1',
 y si no toma ninguno de los valores especificados devuelve 'otro'.

```
SELECT department_id, DECODE (department_id,10,'MADRID', 20, 'SEVILLA', department_id)
FROM employee;
```

	DEPARTMENT_ID	DECODE(DEPARTMENT_ID,10,'MADRID',20,'SEVILLA',DEPARTMENT_ID)
7	13	13
8	13	13
9	23	23
10	23	23
11	20	SEVILLA
12	12	12
13	13	13
14	24	24
15	30	30
16	34	34
17	30	30
18	10	MADRID
19	20	SEVILLA
20	23	23
21	12	12
22	43	43
23	10	MADRID
24	30	30
25	20	SEVILLA
26	30	30
27	20	SEVILLA
28	12	12
29	12	12
30	10	MADRID
31	13	13
32	23	23

5. EJERCICIOS PROPUESTOS

1. Visualizar la potencia de 5 elevado a 3.
2. Visualizar la raíz cuadrada de 81.
3. Visualizar la posición ASCII de la letra M.
4. Visualizar el carácter que le corresponde al ascii 65.
5. Visualizar los códigos de departamento de todos los empleados (sin que se repitan).
6. Mostrar nombre y lo que cobra (salario + comisión + 10% del salario).
7. Si la paga es mensual cuanto cobraría un trabajador con 14 pagas anuales incluida la comisión.
8. Mostrar nombre de los empleados que perderían capital en caso de asignarles un 15% de salario adicional en vez de comisión.
9. Generar la siguiente salida

```
El tiempo
-----
Hoy Esta Nublado
```

10. Visualizar la longitud tiene la cadena "Hoy esta Nublado"
11. Visualizar los empleados que se llamen MILLER en minúsculas.
12. Visualizar salarios en longitud de 10 con el carácter del \$ a la izquierda

```
-----
JOHN          $$$$$$800
KEVIN         $$$$$$1600
...
```

13. Visualizar, de la tabla precios, la lista de precios en longitud 15 con el símbolo \$ a la derecha
14. Visualizar el nombre de los empleados cambiando las B por J.
15. Visualizar los caracteres 3º y 4º de los nombres de los empleados
16. Visualizar los caracteres 2 a 5 de los nombres de los productos de la tabla de productos

17. Visualiza el nombre y los apellidos de los empleados cuyos apellidos empiecen por la letra A. No utilizar LIKE.
18. Visualiza el nombre y los apellidos de los empleados cuyo nombre termine por E. No utilizar LIKE.
19. Mostrar de la tabla empleados si aparece en el campo middle_initial A que se muestre Alto, y si aparece una M que se muestre mediano y si aparece otra cosa que ponga otros.
20. Sacar el puesto de trabajo en castellano de cada empleado, utilizando la función Decode.
21. Mostrar de la tabla Departamentos, si en el campo LOCATION_ID aparece 122, ponga new york, si aparece 123 ponga chicago, si aparece 124 ponga dallas y si aparece 167 ponga boston.
22. Visualizar los datos de los empleados cuyas iniciales en nombre y apellido sean consecutivas en el alfabeto.
23. Mostrar la última letra del nombre de los empleados.
24. Visualiza un listado con todos los empleados donde aparezca de cada uno de ellos:

 <NOMBRE> cobra anualmente <cantidad>
25. Visualiza un listado con los productos que están a la venta en la tabla precios. Debe aparecer:

 <cod_prod> es un producto a la venta y su precio es de <list_price>
26. Quitar la primera A de los nombres de los empleados independientemente del lugar en el que se encuentren en el nombre.
27. Quitar la última S de los nombres de la tabla clientes independientemente del lugar en el que se encuentre en el nombre