

MÉTODO ANTES DE UNA CONSULTA:

/*****

*** REGLA GENERAL PARA LAS QUERIES*****/

1. EXISTE ALGUN PRECALCULO-- UTILIZO SUBCONSULTAS
2. ME ENCUENTRO EN LA SUBCONSULTA. EXISTE ALGUN PRECALCULO--VUELVO AL PUNTO UNO
3. DONDE ESTA LA INFORMACIÓN-- ESTO LO PONGO EN EL FROM - SI HAY VARIAS TABLAS LAS UNO PK CON FK
4. QUE INFORMACION QUIERO MOSTRAR.
5. EXISTE ALGUNA FUNCION DE AGREGACION-- SI ES ASI, TODO LO QUE ESTA FUERA DE LA FUNCION DE AGREGACIÓN VA AL GROUP BY
6. EXISTE ALGUNA CONDICION SOBRE LAS COLUMNAS DE LAS TABLAS--ESTO VA AL WHERE
7. EXISTE ALGUNA CONDICION SOBRE LAS FUNCIONES DE AGREGACION-- ESTO VA AL HAVING
8. EXISTE ALGUNA ORDENACION-- ESTO VA EN EL ORDER BY.

*****/

EJEMPLO:

The screenshot shows a SQL IDE with a query editor and a results pane. The query is as follows:

```
--MOSTRAR EL EMPLEADO QUE TIENE LA FECHA DEL CONTRATO MAS ANTIGUO
DESCRIBE EMPLEADOS; --BUSCAMOS LA INFORMACION EN LAS TABLAS

V_FECHA_CONTRATO_MAS_ANTIGUO= SELECT MIN(FECCON) --HACEMOS PRIMERO
                                FROM EMPLEADOS;    --EL CALCULO

SELECT NOMBRE, FECCON --HACEMOS LA CONSULTA GENERAL
FROM EMPLEADOS --RECUERDA QUE DEBEN SER LAS MISMAS COLUMNAS DE CONSULTAS
WHERE FECCON= (SELECT MIN(FECCON) --HACEMOS PRIMERO
              FROM EMPLEADOS);    --EL CALCULO
```

The results pane shows the output of the query:

	NOMBRE	FECCON
1	ANDRADE GARCIA, ANDRES	11/10/70

MOSTRAR TODAS LAS TABLAS DE UNA BASE DE DATOS (CAT):

```
SELECT *
FROM CAT
```

ORDER BY: esta cláusula se ordena cualquier columna de menor a mayor (ASC) y de mayor a menor (DESC).

```

3
4 SELECT NOMBRE, APELLIDOS, EDAD
5 FROM ALUMNOS
6 ORDER BY (EDAD)ASC;
7
8
9

```

Resultado de la Consulta x

Todas las Filas Recuperadas: 42 en 0,005 segundos

	NOMBRE	APELLIDOS	EDAD
1	JUAN MARIA	POLEY BORGES	17
2	FRANCISCO	RIVERA GIL	18
3	DAVID	MARTINEZ MERENCIO	18
4	ANDRES	CABANAS PECHERO	18
5	IVAN	MATIAS DEL SOLAR	18
6	MARIA	LEON PEREZ	18
7	CESAR	CONCEPCION CAMILO	19
8	MARIO	CAPACETE VELASCO	19
9	LUIS CARLOS	PIMENTEL TORRES	20

```

3
4 SELECT NOMBRE, APELLIDOS, EDAD
5 FROM ALUMNOS
6 ORDER BY (EDAD)DESC;
7
8
9

```

Resultado de la Consulta x

Todas las Filas Recuperadas: 42 en 0,007 segundos

	NOMBRE	APELLIDOS	EDAD
1	ANTONIO	BUENO GUIADO	(null)
2	EUGENIO	PEREZ JIMENEZ	42
3	JESUS	PEREZ CAMPILLO	41
4	JUAN PABLO	PALOMARES AVILA	33
5	FRANCISCO JOSE	DIAZBAUTISTA	31
6	IRENE	SANCHEZ RUANO	30
7	nelson ramon	galicia carrero	29
8	ALBERTO	CARO BARRERA	28

WHERE: se usa para filtrar alguna información acompañado con los operadores relacionales <=, >=, <>.

```

3
4 SELECT NOMBRE, APELLIDOS, EDAD
5 FROM ALUMNOS
6 WHERE EDAD>=18;
7
8
9

```

	NOMBRE	APELLIDOS	EDAD
1	FRANCISCA MARIA	PEREZ URENA	20
2	JAIME	MARTIN BASTOS	21
3	LUIS CARLOS	PIMENTEL TORRES	20
4	MARIO	CAPACETE VELASCO	19
5	ANGEL	DIAZ DEL RIO YANGUAS	20
6	NYDIA	LOPEZ MONTERO	20
7	MARIA	LEON PEREZ	18
8	JAIME	GARCIA MOLINA	21
9	JESUS	CORTAZAR ROMERA	22

TO_DATE: es útil para filtrar un rango de fecha con un formato en específico. Por ejemplo:

```

4 SELECT *
5 FROM ALUMNOS
6 WHERE EDAD>=19
7 AND FECNAC>= TO_DATE('01/01/2000', 'DD/MM/YYYY')
8 AND FECNAC<= TO_DATE('01/12/2030', 'DD/MM/YYYY');
9
10

```

	CODIGO	NOMBRE	APELLIDOS	FECNAC	CURSO	SEXO	EDAD
1	4	MARIO	CAPACETE VELASCO	09/06/01	1DAM	H	19
2	103	DERIAN YERAY	BATALLAS MINDA	12/07/02	1DAW	H	20
3	111	ALEJANDRO	GOMEZ RIVERA	15/10/02	1DAW	H	20

FORMATO CON FECHA, HORAS, MINUTOS SEGUNDO

```
/*
SELECCIONAR LA FECHA DEL SISTEMA (DIA,MES,AÑO,HORAS24, MINUTOS, SEGUNDOS)
*/

SELECT TO_DATE(SYSDATE, 'DD/MM/YYYY HH24/MI/SS')
FROM DUAL;
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,004 segundos

TO_DATE(SYSDATE,'DD/MM/YYYYHH24/MI/SS')
1 18/01/23

```
153 /*
154 CALCULAR EL ULTIMO DIA DEL MES (FECHA) DEL MES ACTUAL,
155 CON HORAS, MINUTOS, SEGUNDOS
156 */
157 SELECT TO_DATE(LAST_DAY(SYSDATE), 'DD/MM/YYYY HH24/MI/SS')
158         AS ULTIMO_DIA_MES_H_M_SS
159 FROM DUAL;
160
161
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,007 segundos

ULTIMO_DIA_MES_H_M_S
1 31/01/23

PORCENTAJE(FORMULA):

Aulaprende

$$20\% \text{ de } 50 = \frac{50}{100} \cdot 20 = 10$$

FUNCIÓN ROUND: redondea un valor.

```
291 49- CALCULAR CUANTO SE GANA DE MEDIA POR CADA OFICIO. REDONDEAR
292 A 2 DECIMALES
293 */
294 V_MEDIO_OFICIO=
295
296 SELECT JOB, ROUND(AVG(SAL),2) SALARIO_MEDIO
297 FROM EMP
298 GROUP BY JOB;
299
300 /*
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 5 en 0,01 segundos

	JOB	SALARIO_MEDIO
1	ANALYST	3000
2	CLERK	1037,5
3	SALESMAN	1400
4	MANAGER	2758,33
5	PRESIDENT	5000

Activar Windows
Ve a Configuración para activar Window

FUNCIÓN TRUNC: corta el valor del decimal.

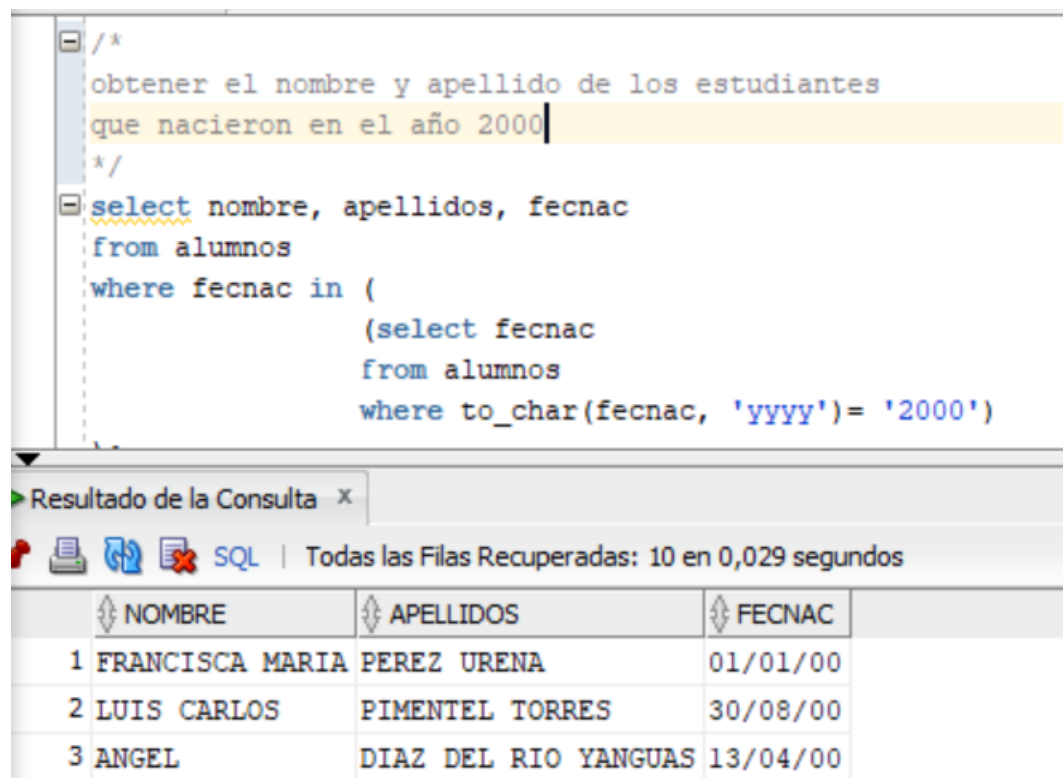
```
4 SELECT TRUNC(155.89) AS CORTA_VALOR
5 FROM DUAL;
6
7
8
9
10
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,003 segundos

	CORTA_VALOR
1	155

FUNCION TO_CHAR: SE USA EN EL WHERE PARA TOMAR EL VALOR DE UNA FECHA (DIA,MES, AÑO), CONVERTIRLO EN CADENA Y LUEGO BUSCAR EL VALOR INDICADO. VEAMOS UN EJEMPLO CON UNA SUBCONSULTA:



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains a comment and a SQL query. The results pane shows the output of the query, which is a table with three columns: NOMBRE, APELLIDOS, and FECNAC. The table contains three rows of data.

```
/*
obtener el nombre y apellido de los estudiantes
que nacieron en el año 2000
*/
select nombre, apellidos, fecnac
from alumnos
where fecnac in (
    (select fecnac
    from alumnos
    where to_char(fecnac, 'yyyy')= '2000')
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 10 en 0,029 segundos

	NOMBRE	APELLIDOS	FECNAC
1	FRANCISCA MARIA	PEREZ URENA	01/01/00
2	LUIS CARLOS	PIMENTEL TORRES	30/08/00
3	ANGEL	DIAZ DEL RIO YANGUAS	13/04/00

SABER LA CANTIDAD DE DIAS QUE HAN PASADO TO_DATE:

```
select fechapago, to_date(fechapago, 'yyyy/mm/dd/hh24/mi/ss')
- to_date('2008/11/08/00/00/00', 'yyyy/mm/dd/hh24/mi/ss') as cantidad_dias
from pagos;
```

```

100 */
101 SELECCIONAR NOMBRE, SUELDO Y SUELDO FORMATEADO CON EL
102 SIMBOLO DOLAR DE TODOS LOS EMPLEADOS
103 */
104 SELECT ENAME, TO_CHAR(SAL, '$9999.99') SALARIO
105 FROM EMP;

```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 14 en 0,002 segundos

	ENAME	SALARIO
1	KING	\$5000.00
2	BLAKE	\$2850.00
3	CLARK	\$2450.00
4	JONES	\$2975.00
5	SCOTT	\$3000.00

Activar Windows

```

113 SELECCIONAR LA FECHA DEL SISTEMA (DIA DE LA SEMANA, DD, MES, AÑO, HORAS24,
114 MINUTOS, SEGUNDOS)
115 */
116 SELECT TO_CHAR(SYSDATE, 'DAY/DD/MM/YYYY HH24/MI/SS') DIA_SEMANA_FECHA_HOY
117 FROM DUAL;
118

```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,003 segundos

	DIA_SEMANA_FECHA_HOY
1	MIÉRCOLES/18/01/2023 22/34/58

TO_DATE CON INTERVALO DE DIAS, HORAS, MINUTOS DE DOS COLUMNAS O FECHAS:

```

88 */
89 10.Devuelve un listado con el código de pedido, código de cliente,
90 fecha esperada y fecha de entrega de los pedidos cuya fecha de
91 entrega ha sido al menos dos días antes de la fecha esperada.
92 */
93 select p.codigopedido, c.codigocliente, p.FECHAESPERADA,
94 p.FECHAENTREGA
95 from pedidos p, clientes c
96 where c.codigocliente= p.codigocliente
97 and p.FECHAENTREGA<=p.FECHAESPERADA - interval '2' day;
98

```

EXPLICACION:

En SQL, INTERVAL es una palabra clave utilizada para representar un intervalo de tiempo específico. El intervalo de tiempo puede ser expresado en términos de días, horas, minutos, segundos y fracciones de segundo.

Por ejemplo, podemos utilizar la sintaxis INTERVAL '2' DAY para representar un intervalo de dos días, INTERVAL '5' HOUR para representar un intervalo de cinco horas, y así sucesivamente.

La palabra clave INTERVAL se utiliza a menudo en conjunción con otras palabras clave como DATE, TIME o DATETIME para realizar operaciones de fecha y hora en bases de datos relacionales.

CONCATENAR DOS COLUMNAS: se usa cuando se quiere combinar dos columnas en una. Ejemplo:

SINTAXIS: COLUMNA1 || ' ESPACIO O CARACTER' || COLUMNA2

```
5 SELECT NOMBRE || ' = ' || EDAD AS NOMBRE_Y_EDAD
6 FROM ALUMNOS;
7
8
9
10
```

Resultado de la Consulta x	
SQL Todas las Filas Recuperadas: 42 en 0,008 segundos	
	NOMBRE_Y_EDAD
1	FRANCISCA MARIA = 20
2	JAIME = 21
3	LUIS CARLOS = 20
4	MARIO = 19
5	ANGEL = 20
6	NYDIA = 20
7	MARIA = 18
8	JAIME = 21
9	JESUS = 22

4	SELECT NOMBRE ' ' APELLIDOS ' = '
5	EDAD AS NOMBRE_APELLIDOS_AÑO
6	FROM ALUMNOS;
7	
8	
9	

Resultado de la Consulta x	
SQL Todas las Filas Recuperadas: 42 en 0,006 segundos	
NOMBRE_APELLIDOS_AÑO	
1 FRANCISCA MARIA PEREZ URENA = 20	
2 JAIME MARTIN BASTOS = 21	
3 LUIS CARLOS PIMENTEL TORRES = 20	
4 MARIO CAPACETE VELASCO = 19	
5 ANGEL DIAZ DEL RIO YANGUAS = 20	
6 NYDIA LOPEZ MONTERO = 20	
7 MARIA LEON PEREZ = 18	
8 JAIME GARCIA MOLINA = 21	
9 JESUS CORTAZAR ROMERA = 22	

FUNCION NVL: esta funcion lo que hace es que cambia un valor nulo por otro valor. Veamos un ejemplo:

--BUSCAMOS LOS VALORES NULOS
SELECT *
FROM ALUMNOS
WHERE EDAD IS NULL;
--ESCRIBIMOS LOS VALORES QUE ESTAN NULOS
SELECT NVL(EDAD, 17) AS EDAD_CAMBIADA,
NVL(FECNAC, TO_DATE('21/01/2021', 'DD/MM/YYYY')) AS FECHA_MOD
FROM ALUMNOS;

Resultado de la Consulta x	
SQL Todas las Filas Recuperadas: 42 en 0,034 segundos	
EDAD_CAMBIADA	NVL(FECNAC,TO_DATE('21/01/2021','DD/MM/YYYY'))
1 20	01/01/00
2 21	12/10/99
3 20	30/08/00
4 19	09/06/01

CONCATENAR LAS COLUMNAS: es útil cuando queremos unir dos columnas. Veamos un ejemplo:

```
--BUSCAMOS EL VALOR QUE QUEREMOS CONCATENAR
SELECT *
FROM ALUMNOS;

--CONCATENAMOS LAS COLUMNAS
SELECT NOMBRE || ' ' || APELLIDOS AS NOMBRE_COMPLETO
FROM ALUMNOS;
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 42 en 0,016 segundos

	NOMBRE_COMPLETO
1	FRANCISCA MARIA PEREZ URENA
2	JAIME MARTIN BASTOS
3	LUIS CARLOS PIMENTEL TORRES
4	MARIO CAPACETE VELASCO
5	ANGEL DIAZ DEL RIO YANGUAS
6	NYDIA LOPEZ MONTERO

FUNCION LOWER: sirve para colocar los datos de las tablas en minúsculas. Ejemplo:

```
--BUSCAMOS EL VALOR QUE QUEREMOS CONCATENAR
SELECT *
FROM ALUMNOS;

SELECT LOWER(NOMBRE)
FROM ALUMNOS;
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 42 en 0,029 segundos

	LOWER(NOMBRE)
1	francisca maria
2	jaime
3	luis carlos
4	mario
5	angel
6	nydia
7	maria
8	jaime

FUNCION UPPER: PASA VALORES QUE ESTÁN EN MINUSCULAS A MAYUSCULAS.
EJEMPLO:

```
--BUSCAMOS LA INFORMACION EN MINUSCULA
SELECT NOMBRE, APELLIDOS
FROM ALUMNOS
WHERE NOMBRE= 'nelson ramon'
AND APELLIDOS= 'galicia carrero';

--PASAMOS LOS VALORES A MAYUSCULAS
SELECT UPPER(NOMBRE), UPPER(APELLIDOS)
FROM ALUMNOS
WHERE NOMBRE= 'nelson ramon'
AND APELLIDOS= 'galicia carrero';
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,018 segundos

	UPPER(NOMBRE)	UPPER(APELLIDOS)
1	NELSON RAMON	GALICIA CARRERO

FUNCION INITCAP: ES UTIL PARA COLOCAR EN MAYUSCULA UN CARACTER DE UNA CADENA. POR EJEMPLO COLOCAR EN MAYUSCULAS EL PRIMER CARACTER DEL NOMBRE Y EL APELLIDO. VEAMOS MAS:

```
1 SELECT INITCAP(NOMBRE), INITCAP(APELLIDOS)
2 FROM ALUMNOS
3 WHERE NOMBRE= 'nelson ramon'
4 AND APELLIDOS= 'galicia carrero';
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,013 segundos

	INITCAP(NOMBRE)	INITCAP(APELLIDOS)
1	Nelson Ramon	Galicia Carrero

FUNCION LPAD Y RPAD:

Hoja de Trabajo Generador de Consultas

```

1  /*
2  FUNCION LPAD (LEFT) Y RPAD (RIGHT)
3  ESTA FUNCION LO QUE HACE ES QUE TE RELLENA CON ESPACIOS
4  O CARACTERES UNA CASILLA.
5
6  RECUERDA QUE DEBES COLOCAR LA LONGITUD DE LOS ESPACIOS
7  */
8
9  SELECT LPAD('NELSON', 10, '*') AS FUNCION_LPAD,
10         RPAD('GALICIA', 10, '*') AS FUNCION_RPAD
11 FROM DUAL;
12
13
14
15

```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,014 segundos

	FUNCION_LPAD	FUNCION_RPAD
1	*****NELSON	GALICIA***

```

9  SELECT LPAD(NOMBRE, 25, '*'),
10         RPAD(APELLIDOS, 25, '*')
11 FROM ALUMNOS
12 WHERE NOMBRE= 'FRANCISCA MARIA';
13
14

```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,048 segundos

	LPAD(NOMBRE,25,'*')	RPAD(APELLIDOS,25,'*')
1	*****FRANCISCA MARIA	PEREZ URENA*****

FUNCION SUBSTR: selecciona de una cadena de A hasta B. La sintaxis es la siguiente:
SUBSTR(NOMBRE_COLUMNNA, 1, 5)

```
1  /*
2  unir en una sola columna el primer nombre y el segundo apellido de la alumna FRANCISCA
3  */
4
5  select substr(nombre,1,10) || ' ' || substr(apellidos,7,11) as
6  from alumnos
7  where nombre like '%FRANCISCA%';
8
```

Resultado de la Consulta x

Todas las Filas Recuperadas: 1 en 0,012 segundos

SUBSTR(NOMBRE,1,10) " " SUBSTR(APELLIDOS,7,11)AS
1 FRANCISCA URENA

BORRAR REGISTROS DE UNA TABLA:

Sintaxis:

DELETE FROM TABLA;

INSERTAR DATOS EN UNA TABLA:

Sintaxis:

RECUERDA QUE SI TE SALE UN ERROR, DEBES VERIFICAR SI HAY ATRIBUTOS QUE NO PUEDEN SER NULOS

INSERT INTO JUGADORES (CODJUGADOR, NOMBRE, FECNAC, DORSAL)
VALUES(25, 'MARCOS ANDRE', TO_DATE('20/10/1996', 'DD/MM/YYYY'),22);

FUNCION LIKE Y NO LIKE: SIRVE PARA BUSCAR UNA PALABRA EN ESPECÍFICO. HAY QUE TENER EN CUENTA QUE LOS % SON IMPORTANTES. SINTAXIS:

SELECT COLUMNAS

FROM TABLAS

WHERE COLUMNA LIKE '%JOSE%'

EN ESTA CONSULTA NOS VA A BUSCAR EN LA COLUMNA DE LA TABLA LAS LETRAS QUE COINCIDAN CON LA PALABRA JOSE:

```
SELECT NOMBRE, HIJOS
FROM EMPLEADOS
WHERE NOMBRE LIKE '%JOSE%';
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 2 e

	NOMBRE	HIJOS
1	TORRES SANCHEZ, JOSE	2
2	PANADERO DURAN, JOSEFA	0

LOS % SON IMPORTANTE PARA INDICARLE HASTA DONDE VAN A BUSCAR, POR EJEMPLO, QUEREMOS BUSCAR EN LA COLUMNA NOMBRE DE LA TABLA EMPLEADO QUE COMIENCEN CON LAS LETRAS 'AN%' AL PRINCIPIO, EL % AL FINAL NOS DICE QUE BUSQUE 'AN%' DESDE EL COMIENZO:

```
28 SELECT NOMBRE, HIJOS
29 FROM EMPLEADOS
30 WHERE NOMBRE LIKE 'AN%';
31
32
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 e

	NOMBRE	HIJOS
1	ANDRADE GARCIA, ANDRES	2

PERO SI QUEREMOS QUE NOS BUSQUE AL FINAL DE CADA CADENA DE CARACTERES LO HACEMOS AL CONTRARIO:

```
28 | SELECT NOMBRE, HIJOS
29 | FROM EMPLEADOS
30 | WHERE NOMBRE LIKE '%AN';
31 |
32 |
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,00

	NOMBRE	HIJOS
1	PEREZ ARREBOLA, ESTEBAN	3

ASIMISMO, TENEMOS EL NOT LIKE PARA DECIRLE QUE NOS SAQUE DE LA BÚSQUEDAS CIERTOS CARACTERES:
AQUÍ DECIMOS QUE EN LA COLUMNA HIJO TIPO NUMBER (ENTERO) NOS SAQUE EL CERO Y DE NOMBRE LAS PALABRAS 'AN' EN CUALQUIER CADENA.

```
28 | SELECT NOMBRE, HIJOS
29 | FROM EMPLEADOS
30 | WHERE HIJOS NOT LIKE '%0%'
31 | AND NOMBRE NOT LIKE '%AN%';
32 |
33 |
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 4 en 0,00

	NOMBRE	HIJOS
1	BENITEZ RAMIREZ, RAFAEL	1
2	LARIOS GIRON, MARTIN	1
3	MARTINEZ LOPERA, INES	2
4	MARIN PALOMO, PABLO	1

FUNCION IN: ESTA NOS PERMITE BUSCAR VARIOS VALORES EN UNA COLUMNA

```
36 | SELECT NOMBRE, SALARIO
37 | FROM EMPLEADOS
38 | WHERE SALARIO IN (180,175);
39 |
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 3 en 0,00

	NOMBRE	SALARIO
1	MENA SALAZAR, RAMON	180
2	SANCHEZ TORO, JESUS	175
3	ANDRADE GARCIA, ANDRES	180

FUNCION CALCULAR MESES TRANSCURRIDOS ENTRE DOS FECHAS
 MONTHS_BETWEEN: DEVUELVE EL NUMERO DE MESES ENTRE UNA FECHA MAYOR
 Y UNA FECHA MENOR. EJEMPLO:

```

138 30--calcular el numero de meses transcurridos entre la fecha de
139 contratacion de cada empleado y hoy
140 */
141 SELECT * FROM EMP;
142
143 SELECT ENAME NOMBRE,
144        MONTHS_BETWEEN(SYSDATE, HIREDATE) MESES_TRANSCURRIDO
145 FROM EMP;
146

```

	NOMBRE	MESES_TRANSCURRIDO
1	KING	494,223235513739545997610513739545997611
2	BLAKE	500,73936454599761051373954599761051374
3	CLARK	499,481300029868578255675029868578255675

FUNCION CALCULA EL ULTIMO DIA DEL MES LAST_DAY: MUESTRA EL ULTIMO DIA
 DEL MES. EJEMPLO:

```

47 30--
48 CALCULAR EL ULTIMO DIA DEL MES (FECHA) DEL MES ACTUAL
49 */
50 SELECT LAST_DAY(SYSDATE) AS ULTIMO_DIA_MES_ACTUAL
51 FROM DUAL;
52
53
54

```

	ULTIMO_DIA_MES_ACTUAL
1	31/01/23

FUNCION INSTR: BUSCA UNA LETRA EN UNA CADENA Y TE MUESTRA LA POSICION
 EN DONDE SE ENCUETRA. PUEDES INDICAR LA POSICION EN DONDE QUIERE QUE
 EMPIECE.

```

--INSTR
SELECT NOMBRE, APELLIDOS, INSTR(NOMBRE, 'A', 10)
FROM ALUMNOS;

```

EJEMPLO 2

4	
5	<code>SELECT NOMBRE, APELLIDOS, INSTR(NOMBRE, 'A')</code>
6	<code>FROM ALUMNOS;</code>
7	

Resultado de la Consulta x			
SQL Todas las Filas Recuperadas: 42 en 0,01 segundos			
	NOMBRE	APELLIDOS	INSTR(NOMBRE, 'A')
1	FRANCISCA MARIA	PEREZ URENA	3
2	JAIME	MARTIN BASTOS	2

FUNCIÓN DISTINCT: EVITA QUE LOS VALORES SE REPITAN EN UNA FILA. EJEMPLO:

1	<code>/*</code>
2	<code>cuantos oficios distintos hay en la tabla empleado</code>
3	<code>*/</code>
4	<code>select * from emp;</code>
5	
6	<code>SELECT DISTINCT(JOB)</code>
7	<code>FROM EMP;</code>
8	

Resultado de la Consulta x	
SQL Todas las Filas Recuperadas: 5 en 0,009 segundos	
JOB	
1 ANALYST	
2 CLERK	
3 SALESMAN	
4 MANAGER	
5 PRESIDENT	

FUNCION DECODE: ES PARECIDA A UN CONDICIONAL. SINTAXIS:

SELECT COLUMNA, DECODE(COLUMNA, SI COLUMNA HAY UNA P, IMPRIME HAY UNA P, SINO BUSCA UNA K, IMPRIME UNA K, SINO MUESTRA EL VALOR POR DEFECTO 'NO HAY NADA')
FROM TABLA
EJEMPLO:

```

206
207 SELECT JOB, DECODE(JOB, 'PRESIDENTE', 'TIENE DINERO', 'MANAGER', 'TIENE
208 ALGO DE DINERO', 'SON POBRES')
209 FROM EMP;
210

```

JOB	DECODE(JOB,'PRESIDENTE','TIENEDINERO','MANAGER','TIENEALGODEDINERO','SONPOBRES')
1 PRESIDENT	SON POBRES
2 MANAGER	TIENEALGO DE DINERO
3 MANAGER	TIENEALGO DE DINERO
4 MANAGER	TIENEALGO DE DINERO
5 ANALYST	SON POBRES
6 ANALYST	SON POBRES
7 CLERK	SON POBRES

Activar Windows

OPERACIONES ARITMÉTICAS

CUANDO TE DIGAN LO QUE HA PAGADO CADA CLIENTE (ES UNA FUNCION SUM)

```

1 --CALCULAR CUANTAS SEMANAS COMPLETAS HA TRABAJADO CADA EMPLEADO
2 SELECT UPPER(ENAME) NOMBRE, TRUNC((SYSDATE-HIREDATE) / 7) AS SEMANAS
3 FROM EMP;
4

```

NOMBRE	TRUNC((SYSDATE-HIREDATE)/7)
1 KING	2147
2 BLAKE	2176
3 CLARK	2170

```

117  CALCULAR EL NUMERO DE DIAS VIVIDOS POR
118  UNA PERSONA NACIDA EL DIA 3 DE JULIO DE 1970
119  */
120  SELECT ROUND(SYSDATE - TO_DATE('03/07/1970', 'DD/MM/YYYY')) DIAS_VIVIDOS
121  FROM DUAL;
122
123

```

Resultado de la Consulta x

Todas las Filas Recuperadas: 1 en 0,004 segundos

DIAS_VIVIDOS	
1	19193

FUNCION SUM:

```

218  39-CALCULAR CUANTO SE PAGA MENSUALMENTE A TODOS LOS EMPLEADOS
219  */
220  SELECT SUM(SAL) SALARIO_TOTAL
221  FROM EMP;
222
223
224
225

```

Salida de Script x Resultado de la Consulta x

Todas las Filas Recuperadas: 1 en 0,002 segundos

SALARIO_TOTAL	
1	29025

FUNCION COUNT(CANTIDAD):

```

225  /*
226  40- CALCULAR CUANTOS EMPLEADOS HAY EN LA TABLA
227  */
228  SELECT COUNT(*) AS EMPLEADOS
229  FROM EMP;
230

```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,003 segundos

EMPLEADOS	
1	14

```

231  /*
232  CALCULAR EL SUELDO MEDIO DE TODOS LOS EMPLEADOS
233  */
234  SELECT ROUND(AVG(SAL)) "SUELDO PROMEDIO"
235  FROM EMP;
236
237  SELECT SUM(SAL) / COUNT(*)
238  FROM EMP;

```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,002 segundos

SUELDO PROMEDIO	
1	2073

```

239
240  /*
241  CALCULAR LA COMISION MEDIA DE TODOS LOS EMPLEADOS
242  */
243  SELECT SUM(COMM) / COUNT(COMM) COMISION_MEDIA
244  FROM EMP;
245
246
247

```

Salida de Script x Resultado de la Consulta x

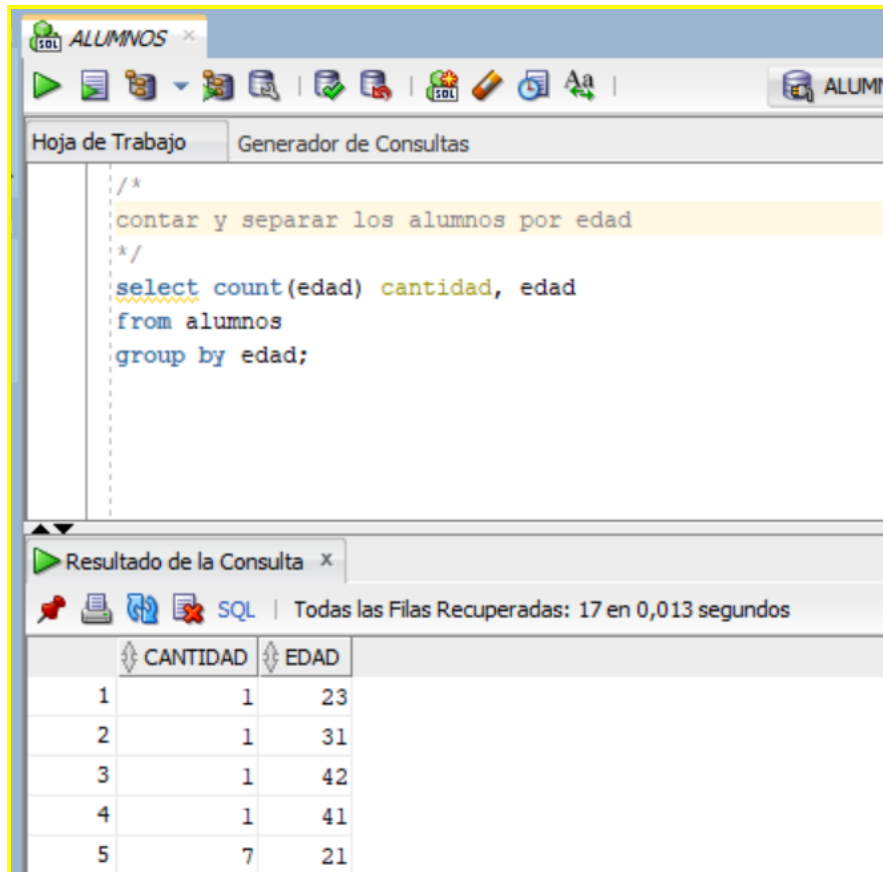
SQL | Todas las Filas Recuperadas: 1 en 0,001 segundos

COMISION_MEDIA	
1	550

GROUP BY: PUEDE AGRUPAR POR COLUMNA ALGUNA CONDICION Y VA ACOMPAÑADA DE FUNCION DE AGREGACIÓN (COUNT, SUM, AVG).

RECUERDA QUE PARA AGRUPAR Y USAR LAS FUNCIONES CON RESULTADOS LÓGICOS DEBEMOS USAR LA CLAVE PRIMARIA DE LA TABLA CON LOS VALORES QUE QUEREMOS REALIZAR ALGUNA OPERACIÓN.

VEAMOS UN EJEMPLO:



The screenshot shows a SQL query editor window titled 'ALUMNOS' with a toolbar and tabs for 'Hoja de Trabajo' and 'Generador de Consultas'. The query text is as follows:








```
/*
contar y separar los alumnos por edad
*/
select count(edad) cantidad, edad
from alumnos
group by edad;
```

Below the editor is a 'Resultado de la Consulta' window showing the results of the query. It indicates that all 17 rows were recovered in 0.013 seconds. The results are displayed in a table with two columns: 'CANTIDAD' and 'EDAD'.

	CANTIDAD	EDAD
1	1	23
2	1	31
3	1	42
4	1	41
5	7	21

FUNCION MIN Y MAX: OFRECE EL MINIMO Y MAXIMO DE UN VALOR. RECUERDA QUE VA CON EL GROUP BY.

EJEMPLO:

<pre> select * from pagos; --mostrar el pago total de cada cliente select codigocliente, sum(cantidad) as pago_total from pagos group by codigocliente; </pre>			
<div>  Resultado de la Consulta x </div> <div>     SQL Todas las Filas Recuperadas: 18 en 0,021 segundos </div>			
	 CODIGOCLIENTE	 PAGO_TOTAL	
1	1	4000	
2	3	10926	
3	4	81849	
4	5	23794	

HAVING: SELECCIONA O RECHAZA CIERTOS NUMEROS DE REGISTROS Y VA ACOMPAÑADA CON GROUP BY. ES ÚTIL PARA REGISTROS ESPECIFICOS. PUEDE IR ACOMPAÑADA DEL HAVING (COUNT(*)), SUM, AVG, ETC

EL HAVING ES UTIL CUANDO NECESITAMOS FILTRAR O PONER UNA CONDICIÓN SOBRE UNA FUNCION

EJEMPLO:

```

/*
contar y separar los alumnos que tenga la edad mayor
o igual a 18 años
*/
select count(edad) cantidad, edad
from alumnos
group by edad
having edad <= 18;

```

Resultado de la Consulta x


 SQL | Todas las Filas Recuperadas: 2 en 0,022 segundos

	CANTIDAD	EDAD
1	1	17
2	5	18

```

17
18 --mostrar el nombre y la cantidad de clientes que hay en españa que
19 --que tengan un límite de credito >20000
20 select nombrecliente, pais, limitecredito, count(*)
21 from clientes
22 where lower(pais) like '%spain%'
23 group by nombrecliente, pais, limitecredito
24 having limitecredito >20000;
25

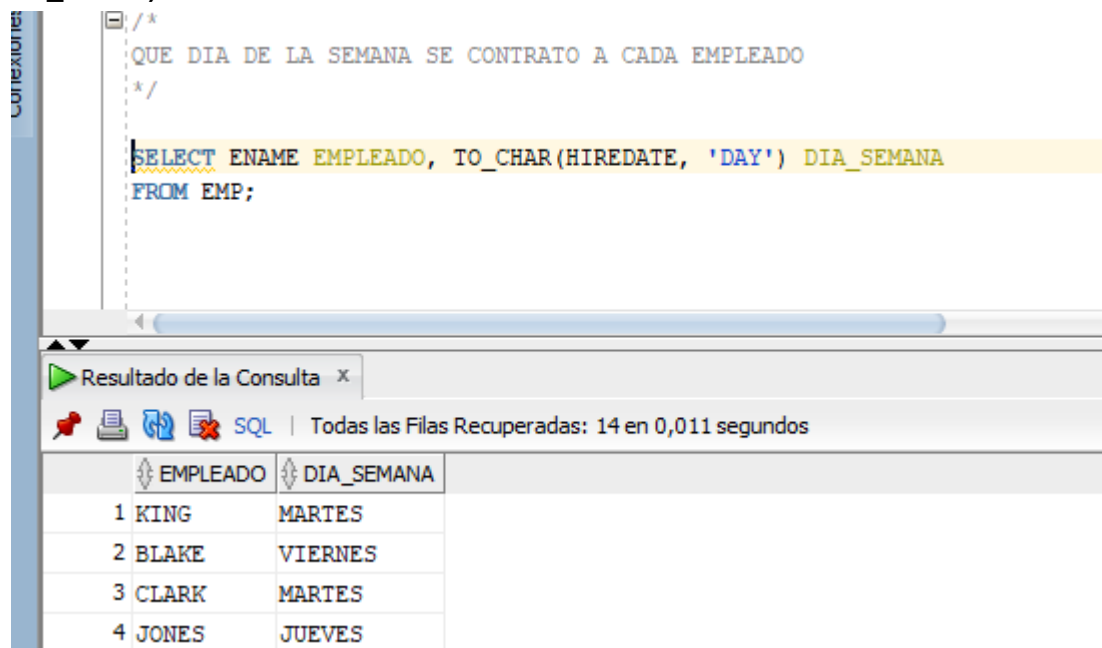
```

Salida de Script x  Resultado de la Consulta x

 SQL | Todas las Filas Recuperadas: 12 en 0,037 segundos

	NOMBRECLIENTE	PAIS	LIMITECREDITO	COUNT(*)
5	Jardines y Mansiones Cactus SL	Spain	76000	1
6	Lasas S.A.	Spain	154310	2
7	Golf S.A.	Spain	30000	1
8	Aloha	Spain	50000	1
9	Sotogrande	Spain	60000	1
10	Jardinerías Matías SL	Spain	100500	1
11	Club Golf Puerta del hierro	Spain	40000	1
12	El Prat	Spain	30000	1

DAY, MONTH: SE OBTIENE EL MES LOS DIAS DE LA SEMANA (VA CON EL TO_CHAR). EJEMPLO:



```
/*
QUE DIA DE LA SEMANA SE CONTRATO A CADA EMPLEADO
*/

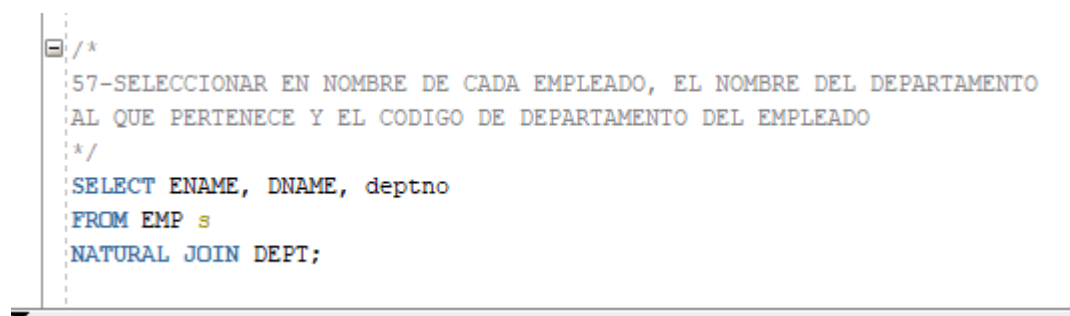
SELECT ENAME EMPLEADO, TO_CHAR(HIREDATE, 'DAY') DIA_SEMANA
FROM EMP;
```

Resultado de la Consulta x

Todas las Filas Recuperadas: 14 en 0,011 segundos

EMPLEADO	DIA_SEMANA
1 KING	MARTES
2 BLAKE	VIERNES
3 CLARK	MARTES
4 JONES	JUEVES

NATURAL JOIN: ES UTIL COMO DOS ATRIBUTOS DE DOS TABLAS SON IGUALES. EJEMPLO:



```
/*
57-SELECCIONAR EN NOMBRE DE CADA EMPLEADO, EL NOMBRE DEL DEPARTAMENTO
AL QUE PERTENECE Y EL CODIGO DE DEPARTAMENTO DEL EMPLEADO
*/

SELECT ENAME, DNAME, deptno
FROM EMP s
NATURAL JOIN DEPT;
```

(+) FUNCION OUTER JOIN: PARA DARLE PRIORIDAD A UNA TABLA O COLUMNA

```
/*
60- LISTAR EL NOMBRE DEL EMPLEADO Y EL NOMBRE DE SU
JEFE. INCLUIR EMPLEADOS QUE NO TENGAN JEFE
*/
SELECT E.ENAME EMPLEADO, J.ENAME JEFE
FROM EMP E, EMP J
WHERE E.MGR= J.EMPNO(+)
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 14 en 0,017 segundos

	EMPLEADO	JEFE
1	BLAKE	KING
2	CLARK	KING
3	JONES	KING
4	ALLEN	BLAKE

```
select c.nombrecliente, c.codigocliente, p.IDTRANSACCION
from clientes c, pagos p
where p.codigocliente(+)= c.codigocliente
and p.IDTRANSACCION is null
group by c.nombrecliente, c.codigocliente, p.IDTRANSACCION;
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 18 en 0,063 segundos

	NOMBRECLIENTE	CODIGOCLIENTE	IDTRANSACCION
15	ruenla city	25	(null)
16	Naturajardin	18	(null)

RELACION REFLEXIVA: Cuando dos columnas son clave primaria y secundaria y con la unión de ella encontramos información en específico. Recuerda que se debe cumplir la siguiente:

- 1- En el from colocar la misma tabla (2 veces)pero con topónimos distintos
- 2- En el where usar el topónimo de la tabla 1 junto con la clave primaria y unirla con el otro topónimo de la tabla 2 junto a esa misma clave primaria
- 3- Si hay otra tabla duplicar la tabla (2) en el from con sus topónimos y hacer lo mismo del punto 2.

sintaxis:

```
SELECT COLUMNAS
FROM TABLA1 A, TABLA1 B
WHERE A.CLAVEPRIMARIA= B.CLAVEFORANEA
```

Ejemplo:

DELETE:

Sintaxis básica:

```
delete
from table
where columna [<=> <>] condicion;
```

Ejemplo:

```
DELETE FROM EMP
WHERE UPPER(ENAME) = 'MILLER';
```

Sintaxis más avanzada:

```
delete
from table
where columna [<=> <>, in] (subconsulta);
```

Ejemplo:

```
-- SOLO SE PUEDE BORRAR DE UNA TABLA
--BORRAR LOS EMPLEADOS CUYO NOMBRE DE DEPARTAMENTO CONTIENE LA
LETRA I
```

Hago la consulta:

```
SELECT DEPTNO
FROM DEPT
WHERE UPPER(DNAME) LIKE '%I%';
```

Hago el delete

```
DELETE
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
                  FROM DEPT
                  WHERE UPPER(DNAME) LIKE '%I%')
;
```

UPDATE:

```
SINTAXIS
UPDATE TABLA
SET CAMPO = VALOR,
.....
CAMPO_N = VALOR_N
WHERE (CONDICIONES);
```

EJEMPLO





```

34  --ACTUALIZAR DEL ALUMNO NELSON LOS CAMPOS NOMBRE, APELLIDO, CURSO
35  --Y SEXO A MAYUSCULAS
36  UPDATE ALUMNOS
37  SET NOMBRE= 'NELSON RAMON',
38      APELLIDOS= 'GALICIA CARRERO',
39      CURSO= '1DAW',
40      SEXO= 'H'
41  where UPPER(NOMBRE) = 'NELSON RAMON';
42
43

```

Salida de Script x

Resultado de la Consulta x

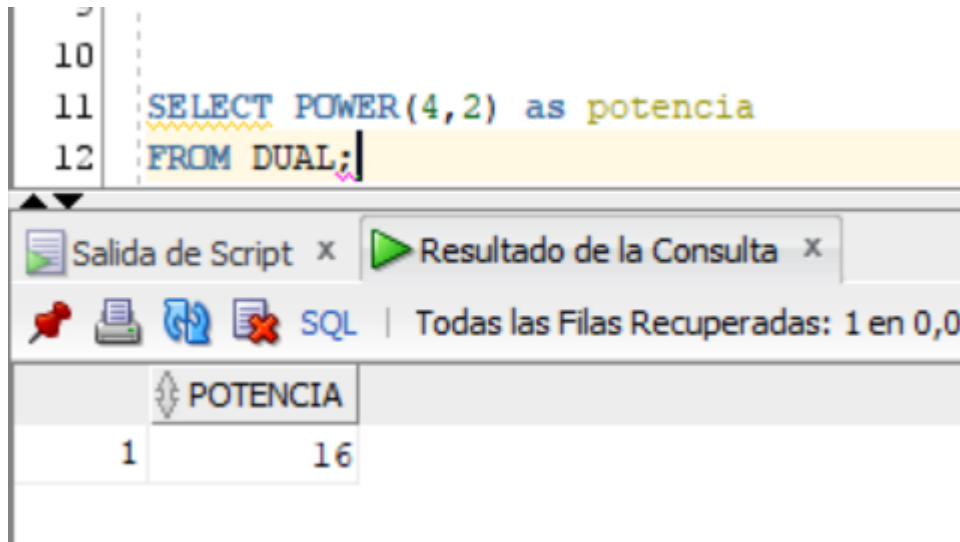


SQL | Todas las Filas Recuperadas: 1 en 0,024 segundos

	CODIGO	NOMBRE	APELLIDOS	FECNAC	CURSO	SEXO	EDAD	
1	112	NELSON	RAMON GALICIA	CARRERO	20/08/93	1DAW	H	29

PROGRAMACIÓN EN PL - SQL

Potencia en sql: en las versiones actuales es con ** pero en versiones anteriores se hace de la siguiente manera:



Atributos de tipos

%TYPE Y CURSORES IMPLICITOS: se usa cuando queremos que una variable tenga el mismo tipo de dato que la columna de la tabla.

Cursores implicito, explicitos: es una consulta select dentro de la programacion en pl/sql. Usualmente, son útiles para devolver datos de una consulta select con una fila

CURSORES IMPLICITOS: DEVUELVE UNA FILA DE UNA TABLA

```

DECLARE
    v_nombre      alumnos.nombre%type;
    v_apellidos   alumnos.apellidos%type;
BEGIN
    SELECT
        nombre, apellidos
    INTO
        v_nombre,
        v_apellidos
    FROM
        alumnos
    WHERE
        edad = 41;

    dbms_output.put_line('EL NOMBRE Y APELLIDO DEL ESTUDIANTE ES '
                        || v_nombre
                        || ' '
                        || v_apellidos);

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('NO SE ENCONTRARON DATOS');
END;

```

CURSORES IMPLICITOS Y CONTROL DE ERRORES:

```

DECLARE
    v_nombre      alumnos.nombre%TYPE;
    v_apellidos   alumnos.apellidos%TYPE;
    v_sexo        alumnos.sexo%TYPE;
BEGIN
    SELECT
        nombre,
        apellidos,
        sexo
    INTO
        v_nombre,
        v_apellidos,
        v_sexo
    FROM
        alumnos
    WHERE
        upper(sexo) = 'M';

```

```

        dbms_output.put_line('EL NOMBRE Y APELLIDO DEL ESTUDIANTE ES '
                               || v_nombre
                               || ' '
                               || v_apellidos
                               || ' '
                               || v_sexo);

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('EL CARACTER ES INVALIDO Y APELLIDO SON INVA
    WHEN too_many_rows THEN
        dbms_output.put_line('LA SELECT DEVUELVE MAS DE 1 VALOR');
END;
```

CURSOS IMPLICITOS CON VALOR POR PANTALLA

```

DECLARE
    v_nombre      alumnos.nombre%TYPE;
    v_apellidos   alumnos.apellidos%TYPE;
    v_sexo        alumnos.sexo%TYPE;
BEGIN
    SELECT
        nombre,
        apellidos,
        sexo
    INTO
        v_nombre,
        v_apellidos,
        v_sexo
    FROM
        alumnos
    WHERE
        upper(sexo) = &INTRODUCE_EL_SEXO;

    dbms_output.put_line('EL NOMBRE Y APELLIDO DEL ESTUDIANTE ES '
                           || v_nombre
                           || ' '
                           || v_apellidos
                           || ' '
                           || v_sexo);

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('EL CARACTER ES INVALIDO Y APELLIDO SON INVA
    WHEN too_many_rows THEN
        dbms_output.put_line('LA SELECT DEVUELVE MAS DE 1 VALOR');
END;
```

CURSORES EXPLICITOS (LOOP): DEVUELVE VARIAS FILAS DE UNA TABLA

```

DECLARE
    CURSOR c_datos_alumnos IS
    SELECT
        nombre,
        apellidos,
        fecnac
    FROM
        alumnos;

    v_nombre      alumnos.nombre%TYPE;
    v_apellidos   alumnos.apellidos%TYPE;
    v_fecnac      alumnos.fecnac%TYPE;
BEGIN
    OPEN c_datos_alumnos;
    LOOP
        FETCH c_datos_alumnos INTO
            v_nombre,
            v_apellidos,
            v_fecnac;

        dbms_output.put_line('DATOS: '
                               || v_nombre
                               || ' '
                               || v_apellidos
                               || ' '
                               || v_fecnac
                               || ' NUMERO DE ESTUDIANTES PROCESADOS: '
                               || c_datos_alumnos%ROWCOUNT);

        EXIT WHEN (c_datos_alumnos%NOTFOUND);
    END LOOP;

    CLOSE c_datos_alumnos;
END;

```

CURSORES EXPLICITOS (WHILE/ MISMO EJEMPLO): DEVUELVE VARIAS FILAS DE UNA TABLA


```

7 DECLARE
3     CURSOR c_datos_alumnos IS
3     SELECT
3         nombre,
1         fecnac
2     FROM
3         alumnos;
4
5     v_nombre alumnos.nombre%TYPE;
5     v_fecnac alumnos.fecnac%TYPE;
7 BEGIN
3     OPEN c_datos_alumnos;
3     WHILE c_datos_alumnos%found LOOP
3         FETCH c_datos_alumnos INTO
1         v_nombre,
2         v_fecnac;
3         EXIT WHEN ( c_datos_alumnos%notfound );
4         dbms_output.put_line('DATOS: '
                                || v_nombre
                                || ' '
                                || v_fecnac
                                || ' NUMERO DE ESTUDIANTES PROCESADOS: '
                                || c_datos_alumnos%rowcount);

        END LOOP;

        CLOSE c_datos_alumnos;
END;

```

CURSORES EXPLICITOS (BUCLE FOR/ MISMO EJEMPLO): DEVUELVE VARIAS FILAS DE UNA TABLA

```

7 DECLARE
3     CURSOR c_datos_alumnos IS
3     SELECT
3         nombre,
1         fecnac
2     FROM
3         alumnos;
4
5     v_nombre alumnos.nombre%TYPE;
5     v_fecnac alumnos.fecnac%TYPE;
7     v_total number:=0;
3 BEGIN
3     FOR valor IN c_datos_alumnos LOOP
3         v_total:= v_total +1;
1         v_nombre:= valor.nombre;
2         v_fecnac:= valor.fecnac;
3         dbms_output.put_line('DATOS: '
                                || v_nombre
0                                || ' '
5

```

```

        || ' '
        || v_fecnac);
    END LOOP;
    dbms_output.put_line('TOTAL DE FILAS RECUPERADAS: ' || v_total);
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('OCURRIO UN ERROR: '
            || sqlcode
            || '--MENSAJE-- '
            || sqlerrm);
END;
```

SOLICITAR POR PANTALLA UN VALOR “&”:

```

1  SET SERVEROUTPUT ON;
2
3  DECLARE
4      NOMBRE VARCHAR2(10) := &INTRODUCE_NOMBRE;
5      APELLIDO VARCHAR2(10) := &INTRODUCE_APELLIDO;
6
7  BEGIN
8      DBMS_OUTPUT.PUT_LINE(NOMBRE || ' ' || APELLIDO);
9
10 END;
```

**EJEMPLOS DE ESTRUCTURAS DE CONTROL (BLOQUE ANONIMO)
IF / ELSE**

```

1 SET SERVEROUTPUT ON;
2
3 DECLARE
4     NOTA NUMBER(5,2) := &INTRODUCE_NOTA;
5
6 BEGIN
7     IF (NOTA >= 0 AND NOTA <= 5) THEN
8         DBMS_OUTPUT.PUT_LINE('SUSPENSO ' );
9     ELSIF (NOTA >= 5) THEN
10        DBMS_OUTPUT.PUT_LINE('APROBADO ' );
11    ELSIF (NOTA >= 7) THEN
12        DBMS_OUTPUT.PUT_LINE('BIEN ' );
13    ELSIF (NOTA >= 8) THEN
14        DBMS_OUTPUT.PUT_LINE('NOTABLE' );
15    ELSIF (NOTA >= 10) THEN
16        DBMS_OUTPUT.PUT_LINE('SOBRESALIENTE' );
17    END IF;
18 END;

```

ESTRUCTURA CASE

```

2
3 DECLARE
4     NOTA NUMBER(5,2) := &INTRODUCE_NOTA;
5
6 BEGIN
7     CASE
8     WHEN (NOTA >= 0 AND NOTA <= 5) THEN
9         DBMS_OUTPUT.PUT_LINE('SUSPENSO ');
10    WHEN (NOTA >= 5) THEN
11        DBMS_OUTPUT.PUT_LINE('APROBADO ');
12    WHEN (NOTA >= 7) THEN
13        DBMS_OUTPUT.PUT_LINE('BIEN ');
14    WHEN (NOTA >= 8) THEN
15        DBMS_OUTPUT.PUT_LINE('NOTABLE ');
16    WHEN (NOTA >= 10) THEN
17        DBMS_OUTPUT.PUT_LINE('SOBRESALIENTE ');
18    END CASE;
19 END;

```

Estructura LOOP

```

set SERVEROUTPUT ON;
/*
REALIZA UN PROGRAMA QUE EJECUTE UN BUCLE LOOP Y SALGA CON UN
EXIT WHEN. Para ello crea una variable entero inicializada a 0
y que se vaya incrementando en el bucle, además de mostrar por
pantalla su valor; la condicion de salida sera cuando dicha
variable valga mas de 20
*/
declare
    numero integer:= 0;
begin
    loop
        numero:= numero + 1;
        dbms_output.put_line(numero);

        exit when numero > 20;
    end loop;
end;

```

OTRA FORMA

```

1  set SERVEROUTPUT ON;
2  declare
3      numero integer:= 0;
4  begin
5      loop
6          if (numero <= 21) then
7              dbms_output.put_line(numero);
8              numero:= numero + 1;
9          else
10             exit;
11         end if;
12     end loop;
13 end;

```

ESTRUCTURA WHILE

```
declare
    numero integer:= 0;
begin
    while (numero <= 21) loop
        if (numero <= 21) then
            dbms_output.put_line(numero);
            numero:= numero + 1;
        else
            exit;
        end if;
    end loop;
end;
```

ESTRUCTURA FOR

```
set SERVEROUTPUT ON;
DECLARE
    numero INTEGER:=0;
BEGIN
    FOR i in 1..22+1 LOOP
        IF (numero <= 21) THEN
            --dbms_output.put_line(i);
            dbms_output.put_line(numero);
            numero:= numero + 1;
        ELSE
            EXIT;
        END IF;
    END LOOP;
END;
```

USO DE LA FUNCION MOD CON BUCLE FOR Y CONDICIONAL

```

set SERVEROUTPUT ON;

begin
  for i in 1..40+1 loop
    if mod(i,2) =0 then
      dbms_output.put_line(i);
    end if;
  end loop;
end;

```

PROCEDIMIENTO

```

5. Codificar un procedimiento que reciba una cadena
y la visualice al revés.
*/

create or replace procedure p_cadena_al_reves (cadena varchar2)
is
  cadena_al_reves varchar2(100);
  cadena_final  varchar2(100);
begin
  cadena_al_reves := cadena;
  for i in reverse 1..length(cadena_al_reves) loop
    cadena_final := cadena_final || SUBSTR(cadena_al_reves,i , 1);
  end loop;
  DBMS_OUTPUT.PUT_LINE(cadena_final);
end;

```

PROCEDIMIENTOS CON CONSULTAR

Crea un procedimiento que se llame consultarEmpleado. Debe tomar una variable de entrada v_empno con el tipo de dato del campo empno de la tabla emp. Debe tomar como variables de salida v_ename y v_job, cuyos tipos de datos deben coincidir con los de los campos ename y job de la tabla emp. Controla con una excepción que no se encuentre ningún dato con el valor de v_empno de entrada, mostrando el mensaje “No se encontraron datos”.

PROCEDIMIENTO

```

1 SET SERVEROUTPUT ON;
2 CREATE OR REPLACE PROCEDURE P_CONSULTAR_EMPLEADO
3 (V_EMPNO EMP.EMPNO%TYPE, V_ENAME OUT EMP.ENAME%TYPE,
4 V_JOB OUT EMP.JOB%TYPE)
5 IS
6 BEGIN
7     SELECT ENAME, JOB
8     INTO V_ENAME, V_JOB
9     FROM EMP
10    WHERE empno= V_EMPNO;
11
12    DBMS_OUTPUT.PUT_LINE('Nombre: ' || V_ENAME || ',
13    Puesto: ' || V_JOB);
14 EXCEPTION
15     WHEN NO_DATA_FOUND THEN
16         DBMS_OUTPUT.PUT_LINE ('No se encontraron datos');
17 END;

```

LLAMADA AL PROCEDIMIENTO

```

1 DECLARE
2     V_ENAME EMP.ENAME%TYPE;
3     V_JOB EMP.JOB%TYPE;
4 BEGIN
5     P_CONSULTAR_EMPLEADO(7369, V_ENAME, V_JOB);
6 END;

```

OTRO EJEMPLO CON INSERT PERO SACANDO MULTIPLO

```

1 -- CREAMOS UN PROCEDIMIENTO QUE PERMITA INSERTAR UN DEPARTAMENTO
2 -- Y QUE DESPUES DE INSERTAR TE DEVUELVA EL CODIGO DE ESTE. DICHO
3 -- CODIGO DE DEPARTAMENTO TIENE QUE SER EL SIGUIENTE MULTIPLO DE
4 -- 10 SUPERIOR AL ULTIMO DADO
5
6 create or replace PROCEDURE spInsertarDeptDevCodigo
7 (p_codigo out dept.deptno%type,
8 p_nombre dept.dname%type,
9 p_loc dept.loc%type)
10 IS
11     v_codigo_mas_10 dept.deptno%type;
12 BEGIN
13
14     -- calcular el codigo con el que voy a insertar
15     select max(deptno)+10
16     into v_codigo_mas_10
17     from dept;
18
19     p_codigo := v_codigo_mas_10 - mod(v_codigo_mas_10,10);
20
21     --insertar
22     INSERT INTO dept (
23         deptno,

```



```

    deptno,
    dname,
    loc
) VALUES (
    p_codigo,
    p_nombre,
    p_loc
);

--fin
end;

declare

    v_dept dept.deptno%type;

begin

    spInsertarDeptDevCodigo(v_dept, 'DAW2', 'SEVILLA2');

    DBMS_OUTPUT.PUT_LINE ('EL DEPT EN SEVILLA2 HA SIDO REGISTRADO CON EL CODIGO ' || v_dept);

end;

```

LLAMADA AL PROCEDIMIENTO

```

declare
    v_dept dept.deptno%TYPE;
begin
    spInsertarDeptDevCodigo(v_dept, 'DATA_ANALISIS', 'CADIZ');
    DBMS_OUTPUT.PUT_LINE('EL REGISTRO HA SIDO GUARDADO CON EL CODIGO ' || v_dept);
END;

SELECT *
FROM DEPT
ORDER BY DEPTNO DESC;

DELETE FROM dept WHERE deptno= 50;

```

PROCEDIMIENTO INSERTANDO DATOS CON EDAD CALCULADA, COD Y MENSAJE DE REGISTRO ALMACENADO

```

SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE spInsertarAlumnoDevCodigo (

    p_codigo IN OUT alumnos.codigo%TYPE,
    p_nombre alumnos.nombre%TYPE,
    p_apellidos alumnos.apellidos%TYPE,
    p_fecnac alumnos.fecnac%TYPE,
    p_curso alumnos.curso%TYPE,
    p_sexo alumnos.sexo%TYPE)
is
    --usamos el tipo de dato del campo edad
    v_edad alumnos.edad%TYPE;
begin
    --calculamos la edad y se guarda en v_edad
    v_edad:= trunc(months_between(sysdate, p_fecnac) / 12);

    --buscamos el codigo mayor y sumamos 1 para crear un cod disponible
    select max(codigo) +1
    into p_codigo
    from alumnos;

    --instruccion del nuevo registro con el codigo y la edad calculada
    INSERT INTO alumnos (
        codigo,
        nombre,
        apellidos,
        fecnac,
        curso,
        sexo,
        edad
    ) VALUES (
        p_codigo,
        p_nombre,
        p_apellidos,
        p_fecnac,
        p_curso,
        p_sexo,
        v_edad
    );
end;

```

LLAMADA AL PROCEDIMIENTO

```

DECLARE
--se usa para almacenar el codigo generado por el procedimiento
--que se va a mostrar por pantalla
    v_codalu alumnos.codigo%TYPE;
BEGIN
--agregamos los argumentos en los parametros
    spinsertaralumnodevcodigo(v_codalu, 'RAMON', 'GALICIA', to_date('20/08/1993'), '2DAW',
        'H');
--imprimimos el mensaje del registro guardado con el codigos
    dbms_output.put('EL ALUMNO HA SIDO REGISTRADO CON EL CODIGO: ' || v_codalu);
END;

SELECT *
FROM ALUMNOS
ORDER BY codigo DESC;

DELETE FROM ALUMNOS
WHERE CODIGO= 31;

```

PROCEDIMIENTO CON PEDIDA DE CODIGO USUARIO POR TECLADO

```

CREATE OR REPLACE PROCEDURE P_CONSULTAR_EMPLEADO
(V_EMPNO EMP.EMPNO%TYPE, V_ENAME OUT EMP.ENAME%TYPE,
V_JOB OUT EMP.JOB%TYPE)
IS
BEGIN
    SELECT ENAME, JOB
    INTO V_ENAME, V_JOB
    FROM EMP
    WHERE empno= V_EMPNO;

    --DBMS_OUTPUT.PUT_LINE('Nombre: ' || V_ENAME || ',
    --Puesto: ' || V_JOB);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE ('No se encontraron datos');
END;

```

LLAMADA AL PROCEDIMIENTO

```

DECLARE
    V_EMPNO EMP.EMPNO%TYPE:= &identificador;
    V_ENAME EMP.ENAME%TYPE;
    V_JOB EMP.JOB%TYPE;
BEGIN
    P_CONSULTAR_EMPLEADO(V_EMPNO, V_ENAME, V_JOB);
    dbms_output.put_line('NUM EMPLEADO: ' ||V_EMPNO
    || ' ' || 'NOMBRE: ' || V_ENAME || ' ' || 'PUESTO: ' || V_JOB);
END;

```

FUNCION CON CONSULTAS

```

CREATE OR REPLACE FUNCTION NOMBRE_ESTUDIANTE
(V_COD_ESTUDIANTE ALUMNOS.CODIGO%TYPE)
RETURN VARCHAR2
IS
    V_ALUM ALUMNOS%ROWTYPE;
BEGIN
    SELECT *
    INTO V_ALUM
    FROM ALUMNOS
    WHERE CODIGO= V_COD_ESTUDIANTE;
    RETURN (V_ALUM.NOMBRE || ', ' || V_ALUM.APELLIDOS);

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN ('NO EXISTE EL ESTUDIANTE CON CODIGO '
                || V_COD_ESTUDIANTE);
END;

```

OTRA SIN PARAMETROS

```

CREATE OR REPLACE FUNCTION TOTAL_ESTUDIANTES
RETURN NUMBER
IS
    V_TOTAL_ALUM NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO V_TOTAL_ALUM
    FROM ALUMNOS;
    RETURN (V_TOTAL_ALUM);
END;

BEGIN
    DBMS_OUTPUT.PUT_LINE('EL NUMERO TOTAL DE ESTUDIANTE ES '
        || TOTAL_ESTUDIANTES);
END;

```

LLAMADA A LA FUNCION

```

BEGIN
    DBMS_OUTPUT.PUT_LINE(NOMBRE_ESTUDIANTE(1));
END;

```

FUNCION CON DATE

```

6. Escribir una función que reciba una fecha y devuelva el año,
en número, correspondiente a esa fecha.
*/
CREATE OR REPLACE FUNCTION f_Fecha(fecha date)
RETURN NUMBER
IS
    v_anio NUMBER;
BEGIN
    v_anio := to_number(to_char(fecha, 'YYYY'));
    DBMS_OUTPUT.PUT('La fecha es ');
    RETURN v_anio;
END;

--forma 1
BEGIN
    DBMS_OUTPUT.PUT_LINE(f_Fecha('01/01/1993'));
END;

```

FUNCION CON SUMA DE FECHAS

```

41 --FUNCION QUE PASEMOS UNA FECHA, NUMERO DE AÑOS Y SUMAR ESA
42 --CANTIDAD DE AÑOS A LA FECHA
43 CREATE OR REPLACE FUNCTION f_sumaAnio
44     (p_fecha date, p_numero integer)
45 return date
46 is
47
48 begin
49     return (add_months(p_fecha, 12 * p_numero));
50 end;
51
52 select f_sumaAnio(to_date('01/06/2009', 'dd/mm/yyyy'),2)
53 from dual;
54
55

```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,047 segundos

F_SUMAANIO(TO_DATE('01/06/2009','DD/MM/YYYY'),2)
1 01/06/11

REGISTROS Y TABLAS

--tablas asociativa (indexed table) es una estructura de datos
--que almacena los datos en pares de clave-valor, y se accede a
--ellos a través de un índice.

declare

/*

El uso de "TYPE PAIS IS RECORD" en el código es para
definir una estructura de datos personalizada, que
contiene tres campos: CO_PAIS, DESCRIPCION y CONTINENTE.
Esta estructura se utiliza para definir el tipo de datos
que se almacena en la tabla asociativa (indexed table)
llamada PAISES.

*/

TYPE PAIS IS RECORD (--is record agrupa varios tipos de datos en
--solo tipo

CO_PAIS NUMBER,
DESCRIPCION VARCHAR2(50),
CONTINENTE VARCHAR2(20));

/*

La sintaxis de "TABLE OF" se utiliza para crear tablas que contienen
un tipo de registro o datos determinados, en este caso, una tabla
que contiene registros de tipo PAIS.

"INDEX BY BINARY_INTEGER" especifica el tipo de índice que se utilizará
para acceder a los elementos de la tabla. En este caso, se utiliza
BINARY_INTEGER para que los índices sean números enteros no negativos.

*/

TYPE PAISES IS TABLE OF PAIS INDEX BY BINARY_INTEGER;

3 /*

La línea "tPAISES PAISES;" crea una variable llamada tPAISES que
es de tipo PAISES. Esta variable se utiliza para almacenar los
datos de la tabla asociativa que se definieron previamente.

*/

tPAISES PAISES;

3 /*

En resumen, "TYPE PAISES IS TABLE OF PAIS INDEX BY BINARY_INTEGER;"
se utiliza para crear una tabla asociativa que contiene elementos
de tipo PAIS, y "tPAISES PAISES;" crea una variable que es una
instancia de esa tabla asociativa. De esta manera, se puede
almacenar información sobre varios países utilizando la
tabla asociativa tPAISES.

*/

```

begin
  --asignamos los valores a las columnas de arriba a la tabla
  --asociativa tPAISES. El indice (1) es para acceder a la primera
  --fila de la tabla
  tPAISES(1).CO_PAIS:= 27;
  tPAISES(1).DESCRIPCION:= 'ITALIA';
  tPAISES(1).CONTINENTE:= 'EUROPA';
  DBMS_OUTPUT.PUT_LINE(tPAISES(1).CO_PAIS || '-' ||
  tPAISES(1).DESCRIPCION || '-'
  || tPAISES(1).CONTINENTE);
end;

```

CONTROL DE EXCEPCIONES:

```

CREATE OR REPLACE FUNCTION f_division
  (num_1 integer, num_2 integer)
return integer
is
  fNombre varchar2(50):= 'f_division';
  v_resultado integer;
begin
  if num_2 = 0 then
    raise zero_divide;
  else
    v_resultado:= (num_1 / num_2);
    return v_resultado;
  end if;
EXCEPTION
when zero_divide then
  dbms_output.put_line(fNombre||'--> EXISTE UNA DIVISIÓN POR CERO');
  return 0;
end;

begin
  dbms_output.put_line(f_division(0,0));
end;

```

PAQUETE: ES COMO UNA CLASE PADRE QUE TIENE ADENTRO PROCEDIMIENTOS Y FUNCIONES Y SE PUEDE LLAMAR DESDE UN BLOQUE ANÓNIMO.

PARA CREAR UN PAQUETE, DEBE DEFINIRSE UNA PLANTILLA (ESPECIFICACION) QUE DEBE COMPILARSE PRIMERO:

En este ejemplo, se nombra un procedimiento pero si hay una función se debe hacer lo mismo

Escriba un paquete gestionEMP con:

☐ Procedimiento nuevoEmpleado que inserte un nuevo empleado con los siguientes datos:

☐ Empno = 8000

☐ Ename = JUAN

☐ JOB = CLERK

☐ MGR = 7902

☐ HIREDATE = 01/05/22

☐ SAL = 1500

☐ COMM = NULL

☐ DEPTNO = 20

☐ Al insertar el nuevo empleado, debe mostrar un mensaje

"Registro creado correctamente".

Llamar al procedimiento en un bloque anónimo y muestra el mensaje.

*/

```
CREATE OR REPLACE PACKAGE gestionemp IS
    PROCEDURE nuevoempleado (
        p_empno      emp.empno%TYPE,
        p_ename      emp.ename%TYPE,
        p_job        emp.job%TYPE,
        p_mgr        emp.mgr%TYPE,
        p_hiredate    emp.hiredate%TYPE,
        p_sal        emp.sal%TYPE,
        p_comm       emp.comm%TYPE,
        p_deptno     emp.deptno%TYPE
    );
END gestionemp;
/
```

LUEGO DEFINIMOS EL CUERPO DEL PAQUETE QUE VA A TENER EL PROCEDIMIENTO MENCIONADO EN EL ANTERIOR BLOQUE:

```
CREATE OR REPLACE PACKAGE BODY gestionemp IS
    PROCEDURE nuevoempleado (
        p_empno      emp.empno%TYPE,
        p_ename      emp.ename%TYPE,
        p_job        emp.job%TYPE,
        p_mgr        emp.mgr%TYPE,
        p_hiredate    emp.hiredate%TYPE,
        p_sal        emp.sal%TYPE,
        p_comm       emp.comm%TYPE,
        p_deptno     emp.deptno%TYPE
    ) IS
```



```

BEGIN
    INSERT INTO emp (
        empno,
        ename,
        job,
        mgr,
        hiredate,
        sal,
        comm,
        deptno
    ) VALUES (
        p_empno,
        p_ename,
        p_job,
        p_mgr,
        p_hiredate,

        p_sal,
        p_comm,
        p_deptno
    );

    dbms_output.put_line('Registro creado correctamente');
END nuevoempleado;

END gestionemp;
/

```

Llamamos al paquete en un bloque anónimo de la siguiente manera:
Recuerda que hay que llamar al paquete.procedimiento(valores)

```

begin
    gestionemp.nuevoempleado(
        8000, 'JUAN', 'CLERK', 7902, to_date('01/05/22', 'dd/mm/yyyy'),
        1500, NULL, 20);
end;

```