ADMINISTRACIÓN DE BASE DE DATOS.

Tema Nº3:Lenguaje de Manipulación de Datos DML I.

Indicador de logro Nº3:

Aplica los comandos DML y las funciones predefinidas en Oracle.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº3:**

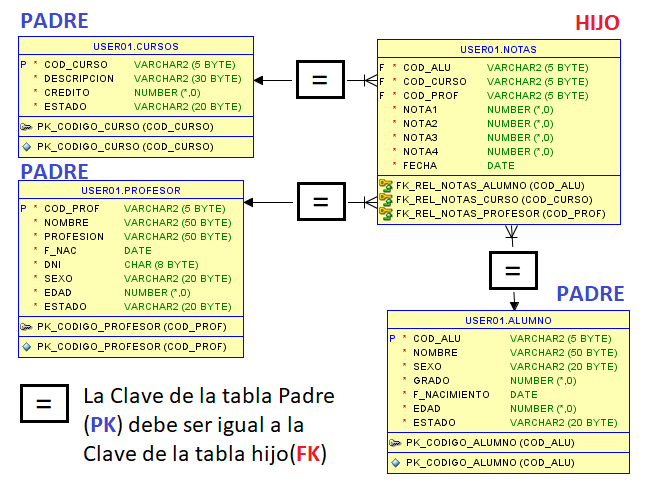
Lenguaje de Manipulación de Datos DML I.

**Subtema 3.1:**

Mantenimiento de la base de datos utilizando DML, Reglas de Inserción, Actualización y Eliminación de datos en tablas. Sentencia INSERT, DELETE, UPDATE.

Es una de las partes fundamentales del lenguaje SQL. El DML (Data Manipulation Language) lo forman las instrucciones capaces de modificar los datos de las tablas. Al conjunto de instrucciones DML que se ejecutan consecutivamente, se las llama transacciones y se pueden anular todas ellas o aceptar, ya que una instrucción DML no es realmente efectuada hasta que no se acepta (**COMMIT**) o rechazar/des hacer (**ROLLBACK**). En todas estas consultas, el único dato devuelto por Oracle es el número de registros que se han modificado.

**PARA REALIZAR LOS MANTENIMIENTOS, SE REQUIERE EL SIGUIENTE MODELO RELACIONAL:**



Este modelo relacional servirá para realizar las pruebas necesarias, en las tareas de inserción de registros, usando diferentes métodos de acceso de datos a cada tabla de dicho modelo relacional. El Script le será proporcionado por su instructor de clases.

**Instrucciones DML y operadores**

Las instrucciones de manipulación de datos (DML) permitidas son SELECT, INSERT, UPDATE y DELETE:

* **SELECT** devuelve filas de la base de datos que cumplan los criterios definidos en su cláusula WHERE.
* **INSERT** añade filas a una tabla de La base de datos.
* **UPDATE DELETE**, modifica las filas de una tabla que cumplan la cláusula WHERE
* **DELETE WHERE**, borra las filas identificadas por la cláusula WHERE.

**INSERCION DE DATOS (INSERT).**

**Inserción de datos Método Implícito:**

Es aquella instrucción **INSERT** que omite el nombre de las columnas en el momento que se realiza la inserción de datos, esta se basa en el orden de las columnas de izquierda a derecha, es decir se asignan datos en cada columna de la tabla de manera ordenada, desde la primera columna, uno por uno, hasta llegar a la última columna. **REGLA DE INSERCION: SIEMPRE SE DEBE EMPEZAR A INSERTAR DATOS EN LAS TABLAS PADRES Y LUEGO EN LAS TABLAS HIJOS**.

Sintaxis del Método implícito:

**INSERT INTO <**Nombre Tabla**> Values (valor1, valor2, valor3, . . ., valor(n));**

**Inserción de datos, Método Explicito.**

Es aquella instrucción INSERT que especifica los nombres de columnas donde se va a almacenar los valores, también se puede especificar la clave NULL es la cláusula VALUES.

Sintaxis del Método Explicito.

**INSERT INTO <**Nombre Tabla**> (**campo 1**,** Campo2**,** Campo3**, . . ..,** Campo**(n) Values (**valor1**,** valor2**,** valor3**, ……,** valor**(n));**

**Copiar filas de datos desde otra tabla:**

Simplemente se realiza un INSERT con una sub consulta.

Sintaxis para copiar filas de datos desde otra tabla

**INSERT INTO <**Nombre Tabla**> (**campo**1,** Campo**2,** Campo**3, . . ..,** Campo**(n)**

**<SELECT** Campo**1,** Campo**2,** Campo**3, …,** Campo**(n) FROM <**Tabla**>**

**WHERE <**Condición**>>**

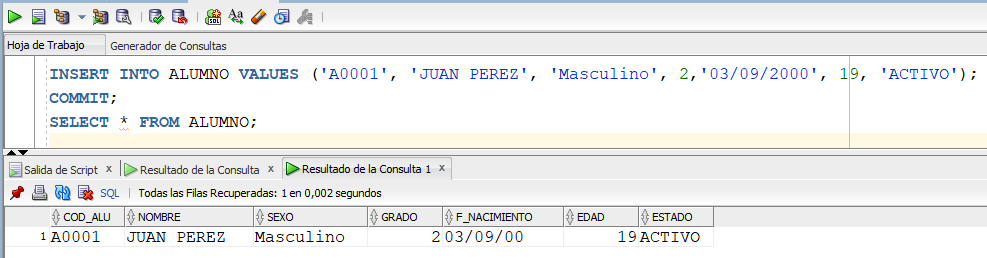
Ejemplo de inserción de método Implícito.

En este caso usaremos la tabla ALUMNO de nuestro modelo relacional, almacenaremos valores en cada columna de la tabla, sin necesidad de nombrar a las columnas, solo respetando el orden de los campos.

**INSERT INTO ALUMNO VALUES (‘A0001’, ’JUAN PEREZ’, ’Masculino’, 2,’03/09/2000’ 19, ’ACTIVO’);**



Si probamos la inserción y una consulta los datos, los resultados se verán de la siguiente manera:



**Nota:** Seleccione el código y luego ejecute.

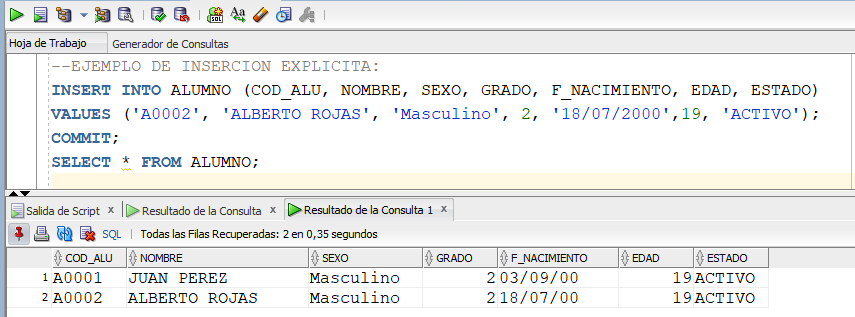
**Ejemplo de inserción de método Explicito.**

En este caso usaremos la tabla ALUMNO de nuestro modelo relacional, almacenaremos valores en cada columna de la tabla, nombrando las columnas.

**INSERT INTO** ALUMNO (COD\_ALU, NOMBRE, SEXO, GRADO, F\_NACIMIENTO, EDAD, ESTADO)

**VALUES** (‘A0002’, ’ALBERTO ROJAS’, ’Masculino’, 2, ‘18/07/2000’,19, ’ACTIVO’)**;**

**Si probamos la inserción y una consulta de datos, los resultados se verán de la siguiente manera:**

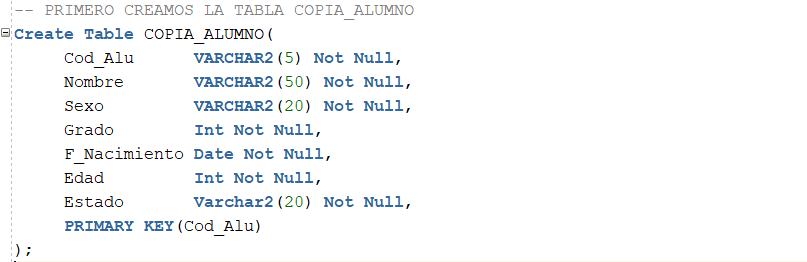


**Nota:** Seleccione el código y luego ejecute.

**Ejemplo de copiar filas de datos, desde otra tabla.**

En este ejemplo las tablas origen (**ALUMNO**) y destino (**COPIA\_ALUMNO**) deben existir, y tener la misma estructura.

**Creamos la tabla COPIA\_ALUMNO:**



**Como ambas tablas están ya creadas, se procede a realizar la copia de origen a destino:**

**INSERT INTO** COPIA\_ALUMNO (COD\_ALU, NOMBRE, SEXO, GRADO,

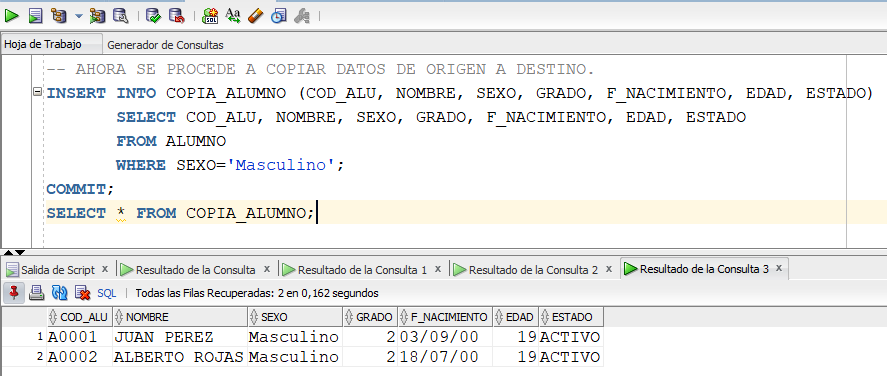
F\_NACIMIENTO, EDAD, ESTADO)

**SELECT** COD\_ALU, NOMBRE, SEXO, GRADO, F\_NACIMIENTO, EDAD, ESTADO

**FROM** ALUMNO

**WHERE** SEXO='Masculino'**;**

Realizando la Prueba:



**ACTUALIZACION DE DATOS (UPDATE).**

Revisar Script: **ACTUALIZAR.sql**

Sintaxis:

**UPDATE** <Nombre\_Tabla> SET

Nombre\_columna1 = expresion1,

Nombre\_columna2 = expresion2,

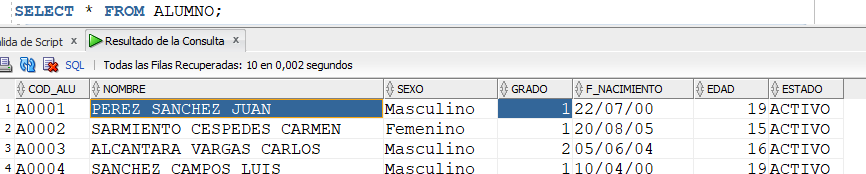
Nombre\_columna3 = expresion3, …

**WHERE** <Condición>;

**REGLA DE ACTUALIZACION:** NUNCA ACTUALICE DATOS EN COLUMNAS QUE TENGAN VALORES REFERENCIALES CON OTRAS TABLAS.

Ejemplo 1:

Se desea modificar el nombre del alumno: **PEREZ SANCHEZ JUAN** por **JUAN CARLOS PEREZ SANCHEZ**, y el grado 1 de dicho alumno ahora será **3**.

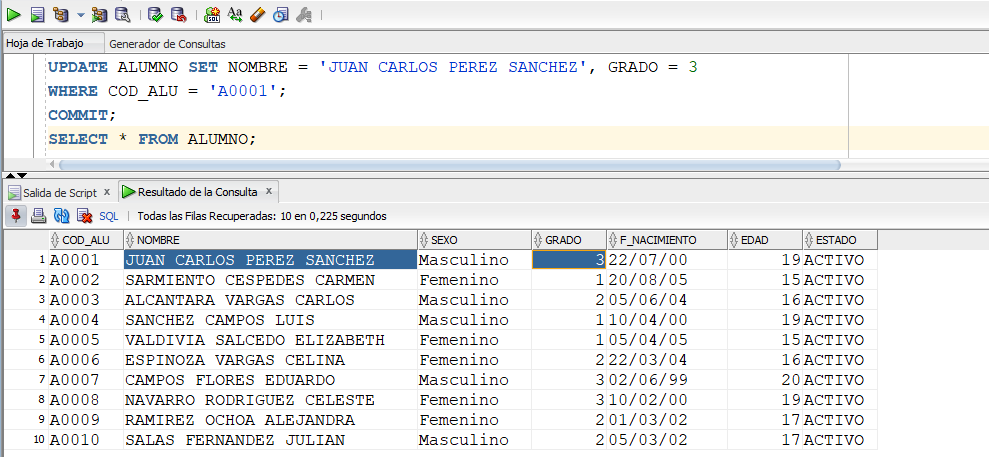


**SOLUCION:**

**UPDATE** ALUMNO **SET** **NOMBRE** = 'JUAN CARLOS PEREZ SANCHEZ’, **GRADO** = 3

**WHERE** **COD\_ALU** = 'A0001';

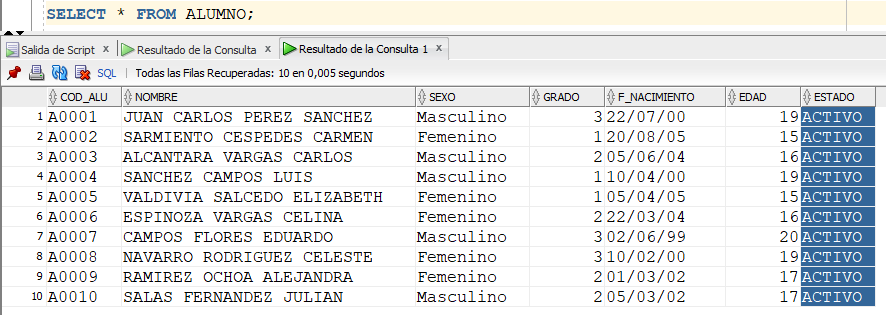
**SOLUCIÓN:** Seleccione el código y ejecute.



**Ejemplo 2:**

Se desea bloquear a todos los alumnos.

En este momento los Estados son **ACTIVOS**.



**SOLUCIÓN:**

**UPDATE** ALUMNO **SET** ESTADO = 'BLOQUEADO';

**COMMIT;**

**SELECT** \* **FROM** ALUMNO;

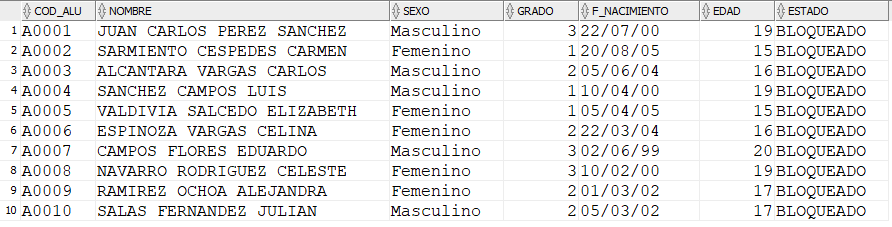
**El Código debe ser seleccionado y Ejecutado.**



**Ejemplo 3:**

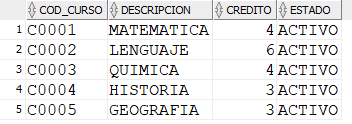
Se tienen las siguientes tablas:

**ALUMNO**



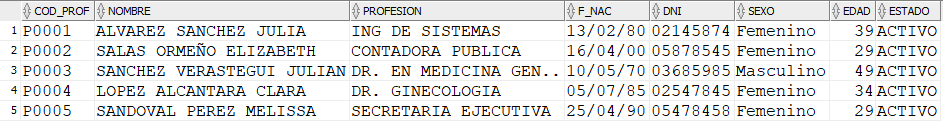
**1**

**CURSOS**

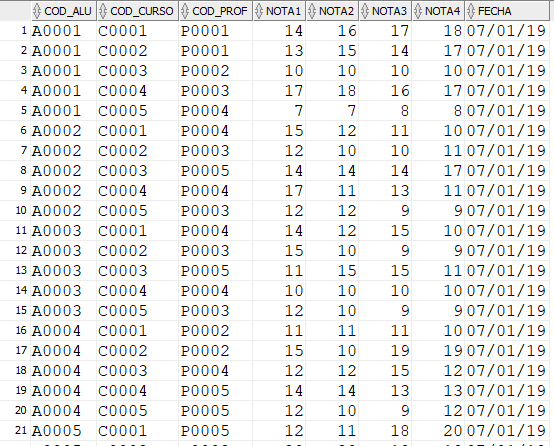


**3**

**PROFESOR**



**NOTAS**



**2**

El Alumno (1) **JUAN CARLOS PEREZ SANCHEZ**, tiene como código de alumno a **A0001**, dicho alumno tiene 2 cursos con bajas calificaciones, los códigos de curso son: (2) **C0003** y **C0005**, es decir los cursos de (3) QUÍMICA y GEOGRAFÍA, Se pide actualizar las notas de dichos cursos, asignar la calificación de 13 a los mencionados cursos.

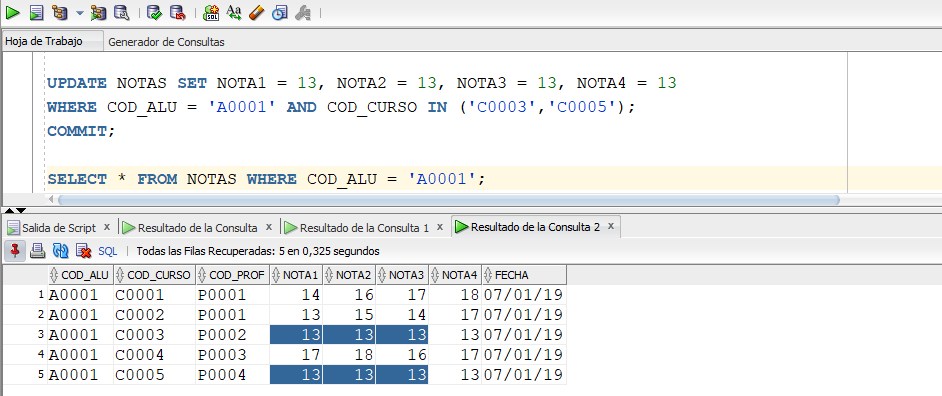
**SOLUCIÓN:** Selecciones el Código y Ejecute.

**UPDATE** NOTAS **SET** NOTA1 = 13, NOTA2 = 13, NOTA3 = 13, NOTA4 = 13

**WHERE** COD\_ALU = 'A0001' **AND** COD\_CURSO **IN** ('C0003','C0005');

**COMMIT;**

**SELECT** \* **FROM** NOTAS **WHERE** COD\_ALU = 'A0001';



**ELIMINAR REGISTROS**: DELETE FROM

La orden DELETE elimina los registros que cumplen con la condición definida en el WHERE.

La sintaxis es la siguiente:

**DELETE FROM** <Nombre\_de\_Tabla> WHERE <Condición>;

**Nota:** Si en algún momento se eliminan registro que tengan referencias con datos de otras tablas (Relación), debe tomar una decisión importante, y estas pueden ser:

* Eliminar los registros usando la finalización CASCADE CONTRAINT, este comando rompe la relación y procede a eliminar los registros.
* Simplemente no borrar.

**Ejemplo 1:**

Eliminar los registros de la tabla alumno, solo aquellos cuyo grado sea 1.

**Solución:**

**DELETE** **FROM** ALUMNO **WHERE** GREADO = 1;

**Ejemplo 2:**

Eliminar los registros de la tabla alumno, aquellos cuyos grados sean 3, 4, 5

**Solución:**

**DELETE** **FROM** ALUMNO **WHERE** GREADO = 3 AND GRADO = 4 AND GRADO = 5;

**Ejemplo 3:**

Elimine todos los registros de la tabla alumno.

**Solución:**

**DELETE** **FROM** ALUMNO;

**CONFIRMAR Y DESHACER.**

Comandos: **COMMIT** y **ROLLBACK**;

**COMMIT**: Comando de confirmación de datos, es decir su uso confirma la inserción, eliminación y actualización de datos en la base de datos.

**ROLLBACK**: Comando que deshace los datos, es decir su uso descarta toda inserción, eliminación y actualización de datos en la base de datos.

**EN QUE MOMENTO USARLO:**

Usar estos comandos después de una inserción, eliminación y actualización de datos.

**Subtema 3.2:**

Recupera información de una base de datos Oracle a través de sentencias SQL SELECT, ordenamiento, Condicionales, Funciones agregadas, Agrupación de datos.

**CONSULTA DE DATOS**:

COMANDO **SELECT**.

Una instrucción SELECT extrae datos de la base de datos. Permite consultar los valores que contienen los campos de tablas y/o vistas de la base de datos. La forma general de una instrucción SELECT es:

**SELECT** [**DISTINCT** | **ALL**] {\*, columna [alias], expresión, …}

**FROM** <tablas> [**WHERE** condición(es)]

[**GROUP BY** expresión [, expresión] … ]

[**HAVING** condición]

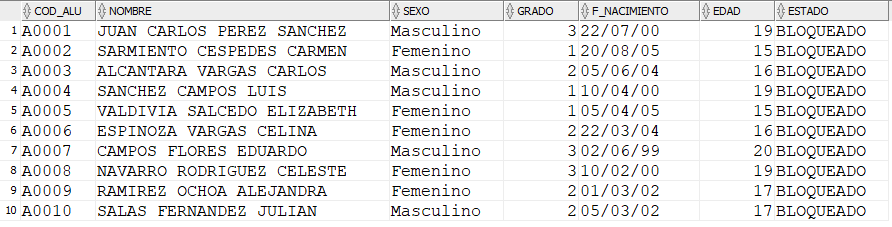
[**ORDER BY** {columna, expresión, alias} {**ASC** | **DESC**}]

**APRENDER A SELECCIONAR TODAS LAS COLUMNAS DE LA TABLA.**

**Carácter comodín \* (asterisco).**

**Ejemplo 1:**

**SELECT \*** **FROM** ALUMNO;



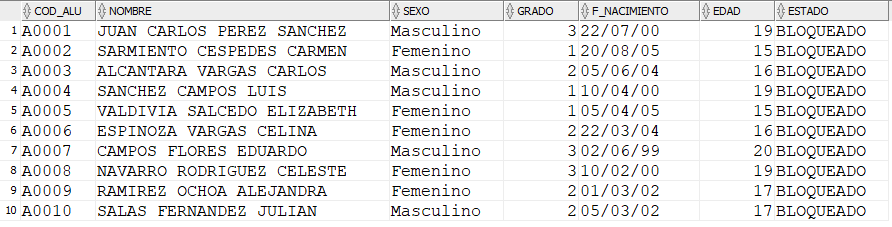
**APRENDER A SELECCIONAR COLUMNAS ESPECIFICAS**.

**Ejemplo 2:**

Seleccione las columnas de código, nombre y sexo del alumno.

**SOLUCIÓN:**

**SELECT** COD\_ALU, NOMBRE, SEXO **FROM** ALUMNO;



**APRENDER A CREAR CAMPOS CALCULADOS.**

Aritméticas

Crear expresiones con datos de fecha y números mediante operadores aritméticos.

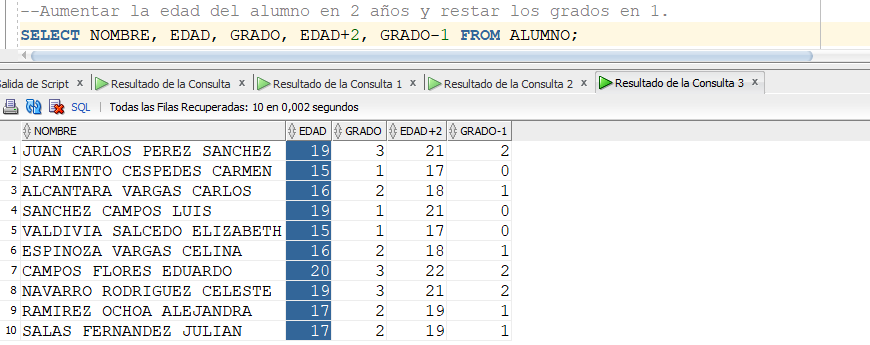
|  |  |
| --- | --- |
| **OPERADOR** | **DEFINICIÓN** |
| + | SUMA |
| - | RESTA |
| \* | MULTIPLICACION |
| / | DIVISION |

**Ejemplo 3:**

Aumentar la edad del alumno en 2 años y restar los grados en 1.

**SOLUCIÓN:**

**SELECT** NOMBRE, EDAD, GRADO, EDAD+2, GRADO-1 **FROM** ALUMNO;



**Ejemplo 4:**

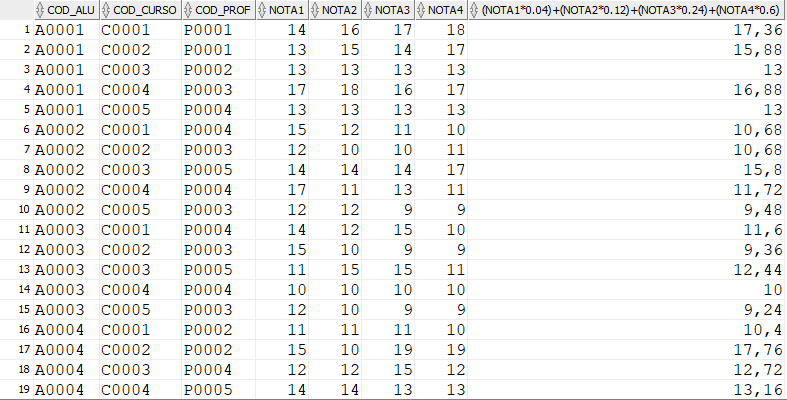
Obtener el Promedio de las calificaciones.

**SOLUCIÓN:**

**SELECT** COD\_ALU, COD\_CURSO, COD\_PROF, NOTA1, NOTA2,

NOTA3, NOTA4, **(NOTA1\*0.04) +** **(NOTA2\*0.12) + (NOTA3\*0.24) + (NOTA4\*0.6)**

**FROM** NOTAS;



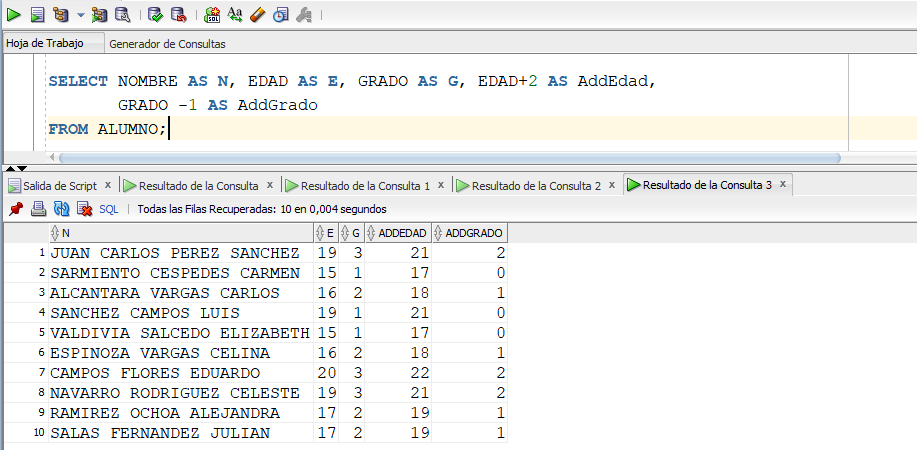
**USO DE ALIAS A LAS COLUMNAS.**

Asignar al campo **Nombre** el alias **N**, **Edad** el alias **E**, **Grado** el alias **G**, **Edad+2** el alias **AddEdad**, **Grado – 1** el alias **AddGrado**. Mediante una consulta realizada en la tabla ALUMNO.

**SELECT** NOMBRE AS **N**, EDAD AS **E**, GRADO AS **G**, EDAD+2 AS **AddEdad**,

GRADO -1 AS **AddGrado**

**FROM** ALUMNO;



**OPERADOR DE CONCATENACIÓN**

Un operador de concatenación enlaza columnas o cadenas de caracteres a otras columnas, se representa con dos barras verticales (||). Crea una columna resultante que es una expresión de carácter.

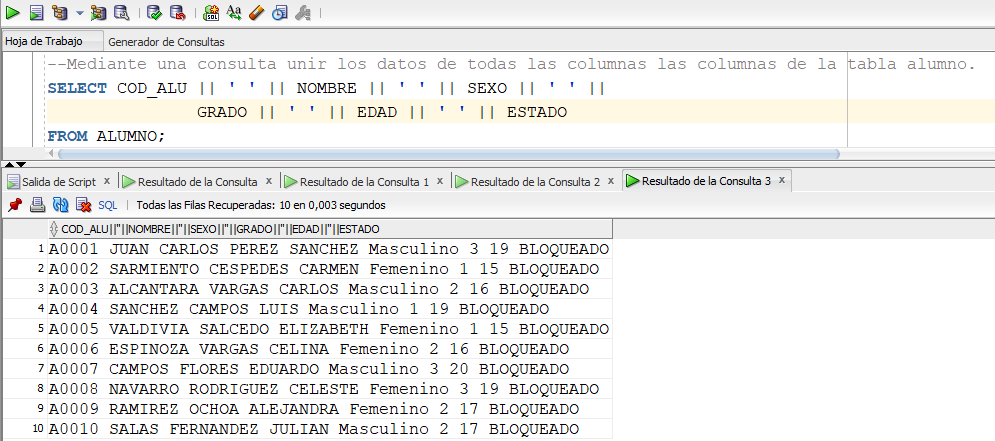
**Ejemplo 1:**

Mediante una consulta unir los datos de todas las columnas las columnas de la tabla alumno.

**SELECT** COD\_ALU **||** ' ' **||** NOMBRE **||** ' ' **||** SEXO **||** ' ' **||**

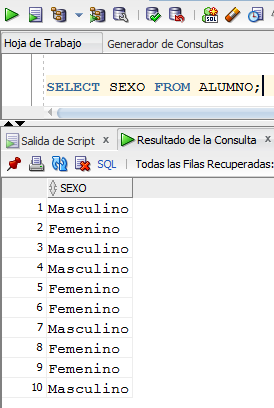
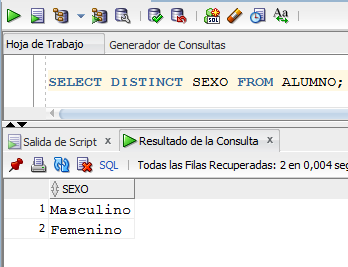
GRADO **||** ' ' **||** EDAD **||** ' ' **||** ESTADO

**FROM** ALUMNO;



**SEPARAR FILAS DUPLICADAS, Clausula: DISTINCT.**

La visualización por defecto de las consultas incluye todas las filas, también las filas duplicadas.



**VISUALIZAR LA ESTRUCTURA DE UNA TABLA**

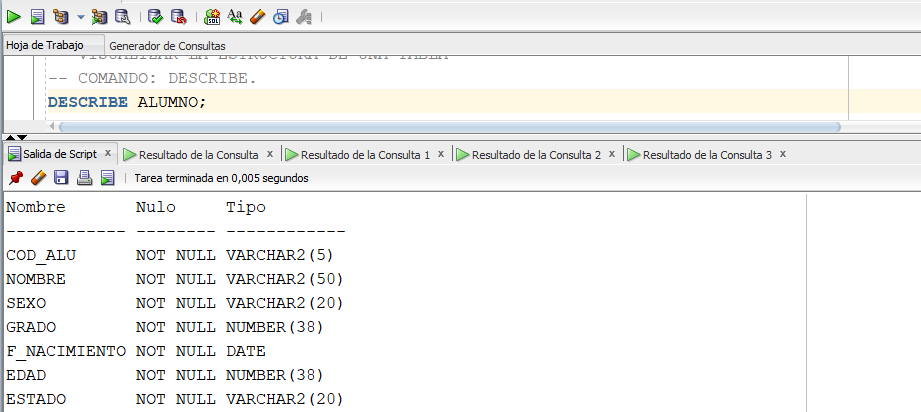
COMANDO**:** **DESCRIBE**.

**Sintaxis:**

**DESCRIBE** <Nombre\_Tabla>;

**Ejemplo**:

**DESCRIBE** ALUMNO;



**GENERAR CONSULTAS BASADAS EN CONDICIONALES**

CLAUSULA**: WHERE.**

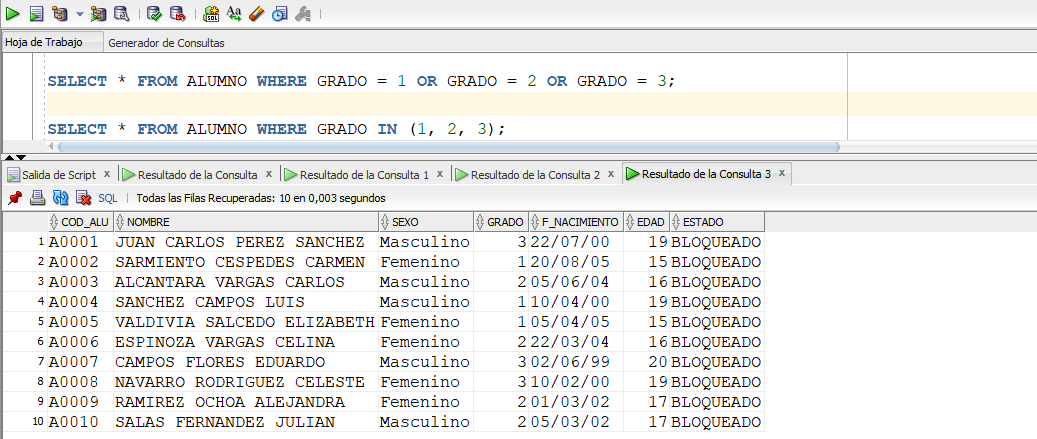
**Ejemplo 1**:

Mostrar aquellos alumnos cuyo grado sea 1, 2 ó 3.

**SELECT** \* **FROM** ALUMNO **WHERE** GRADO = 1 **OR** GRADO = 2 **OR** GRADO = 3;

ó

**SELECT** \* **FROM** ALUMNO **WHERE** GRADO **IN** (1, 2, 3);

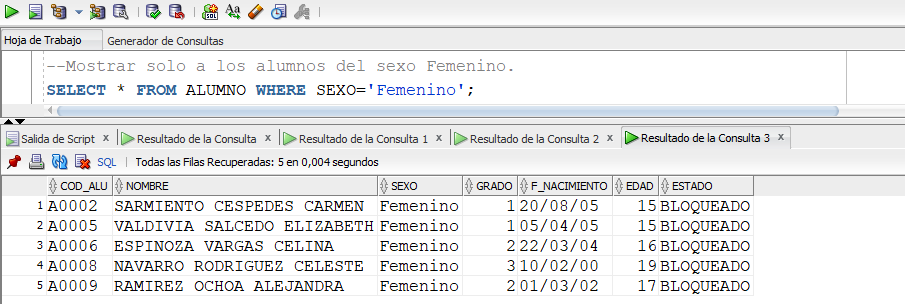


**Ejemplo 2:**

Mostrar solo a los alumnos del sexo Femenino.

**SOLUCIÓN:**

**SELECT** \* **FROM** ALUMNO **WHERE** SEXO=**'Femenino'**;



**Fechas y cadenas de caracteres**

* Las cadenas de caracteres y valores de fecha se incluyen entre comillas simples.
* Los valores de caracteres son sensibles a mayúsculas / minúsculas y los valores de datos son sensibles a formato.
* El formato de visualización de la fecha por defecto es DD-MON-RR

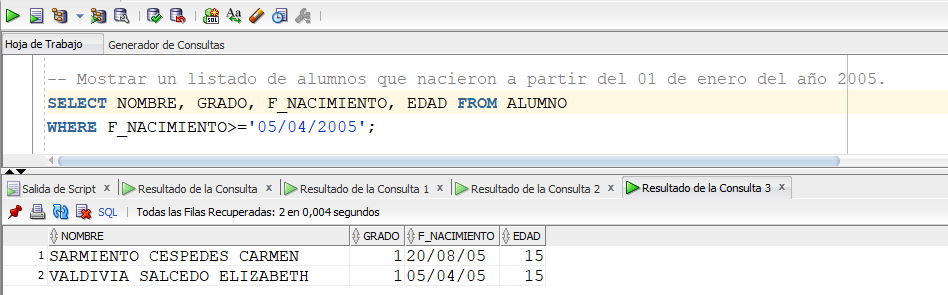
Ejemplo:

Mostrar un listado de alumnos que nacieron a partir del 01 de enero del año 2005.

**SOLUCION:**

**SELECT** NOMBRE, GRADO, F\_NACIMIENTO, EDAD **FROM** ALUMNO

**WHERE** F\_NACIMIENTO>=**'05/04/2005'**;



**OPERADORES DE COMPARACION:**

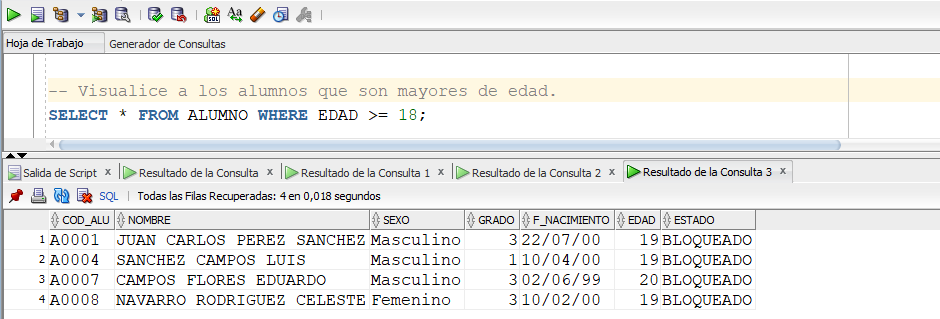
|  |  |
| --- | --- |
| **Operador** | **Descripción** |
| = | Igual que |
| > | Mayor que |
| < | Menor que |
| >= | Mayor igual que |
| <= | Menor igual que |
| <> | Diferente |
| BETWEEN - AND | Entre dos valores. |
| IN ( lista de valores) | Incluido en la lista de valores |
| LIKE | Que coincida con un patrón de valores. |
| IS NULL | Es un valor nulo. |

**USO DEL OPERADOR DE COMPARACIÓN:**

**Ejemplo 1: USO DEL OPERADOR >=**

Visualice a los alumnos que son mayores de edad.

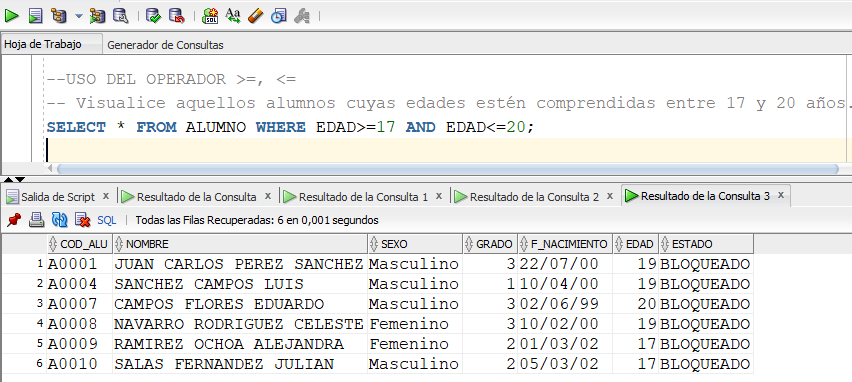
**SELECT** \* **FROM** ALUMNO **WHERE** EDAD **>=** 18;



**Ejemplo 2: USO DEL OPERADOR >=, <=**

Visualice aquellos alumnos cuyas edades estén comprendidas entre 17 y 20 años.

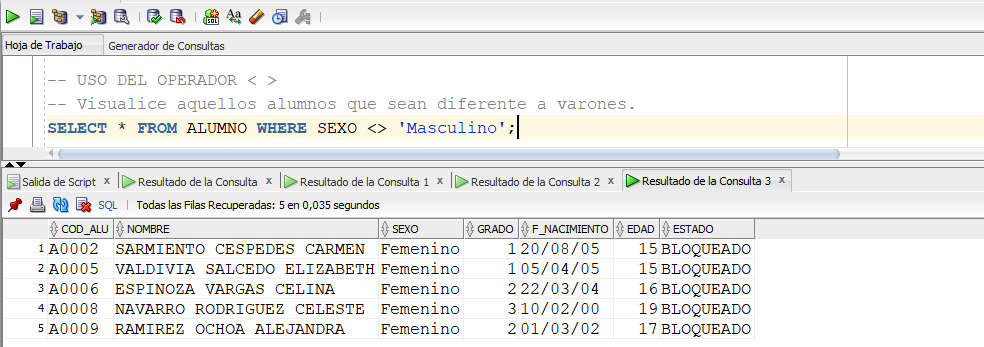
**SELECT** \* **FROM** ALUMNO **WHERE** EDAD**>=**17 AND EDAD**<=**20;



**Ejemplo 3: USO DEL OPERADOR < >**

Visualice aquellos alumnos que sean diferente a varones.

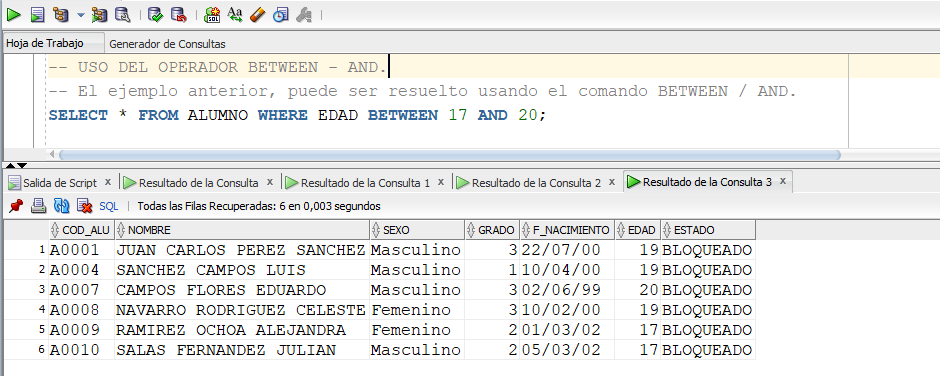
**SELECT** \* **FROM** ALUMNO **WHERE** SEXO <> **'Masculino'**;



**Ejemplo 4: USO DEL OPERADOR BETWEEN - AND.**

El ejemplo anterior, puede ser resuelto usando el comando BETWEEN / AND.

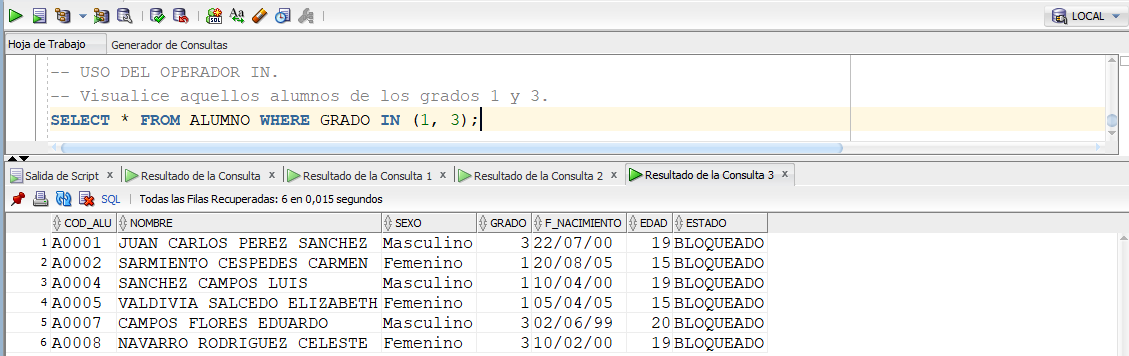
**SELECT** \* **FROM** ALUMNO **WHERE** EDAD **BETWEEN** 17 **AND** 20;



**Ejemplo 5: USO DEL OPERADOR IN.**

Visualice aquellos alumnos de los grados 1 y 3.

**SELECT** \* **FROM** ALUMNO **WHERE** GRADO **IN** (1, 3);



**USO DEL OPERADOR LIKE.**

Antes de usar este operador, debe tener en cuenta lo siguiente:

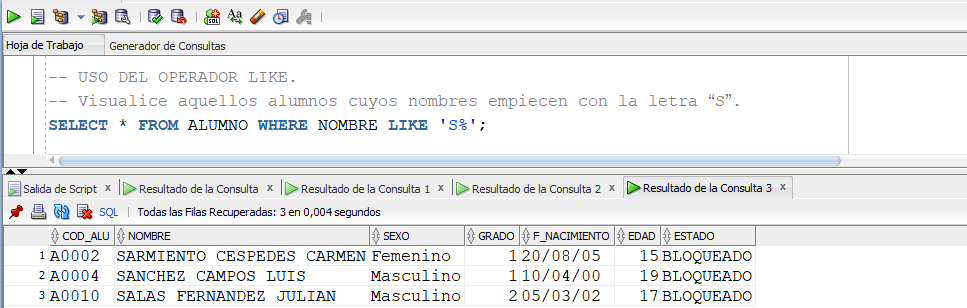
Utilizar el operador LIKE para realizar búsquedas con comodines de valores de cadena de búsqueda válidos.

* Las condiciones de búsqueda pueden contener caracteres literales o números
* % indica cero o varios caracteres
* \_ indica un carácter

**Ejemplo 6: USO DEL OPERADOR LIKE**.

Visualice aquellos alumnos cuyos nombres empiecen con la letra “S”.

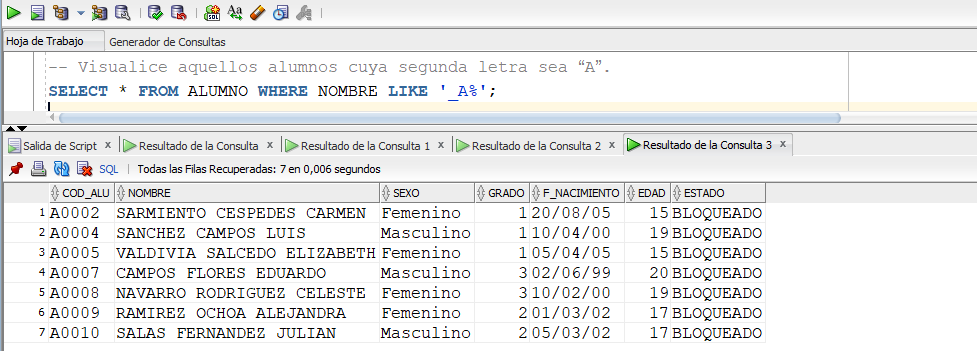
**SELECT** \* **FROM** ALUMNO **WHERE** NOMBRE **LIKE 'S%';**



**Ejemplo 7: USO DEL OPERADOR LIKE.**

Visualice aquellos alumnos cuya segunda letra sea “A”.

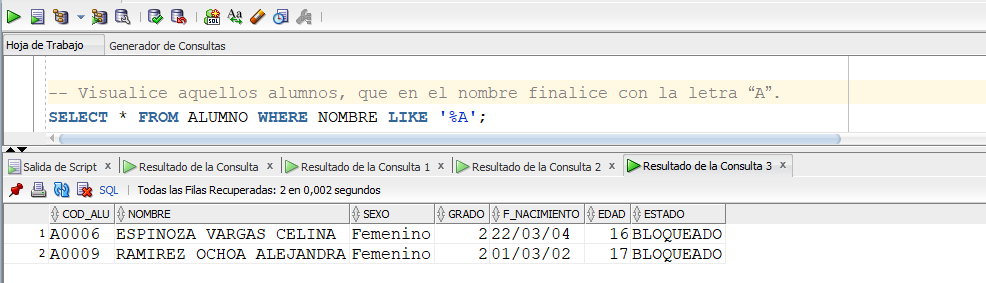
**SELECT** \* **FROM** ALUMNO **WHERE** NOMBRE **LIKE** '\_A%';



**Ejemplo 8: USO DEL OPERADOR LIKE.**

Visualice aquellos alumnos, que en el nombre finalice con la letra “A”.

**SELECT** \* **FROM** ALUMNO **WHERE** NOMBRE **LIKE '%A';**

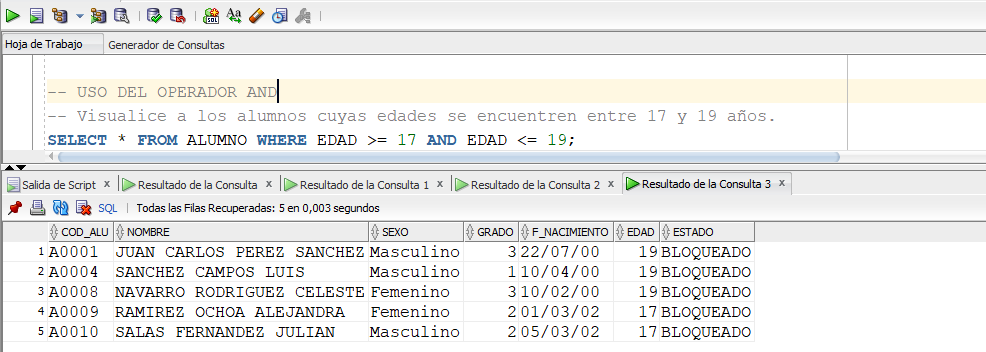


**USO DE OPERADORES LÓGICOS EN CONDICIONALES**

**Ejemplo 9: USO DEL OPERADOR AND**

Visualice a los alumnos cuyas edades se encuentren entre 17 y 19 años.

**SELECT** \* **FROM** ALUMNO **WHERE** EDAD >= 17 **AND** EDAD <= 19;

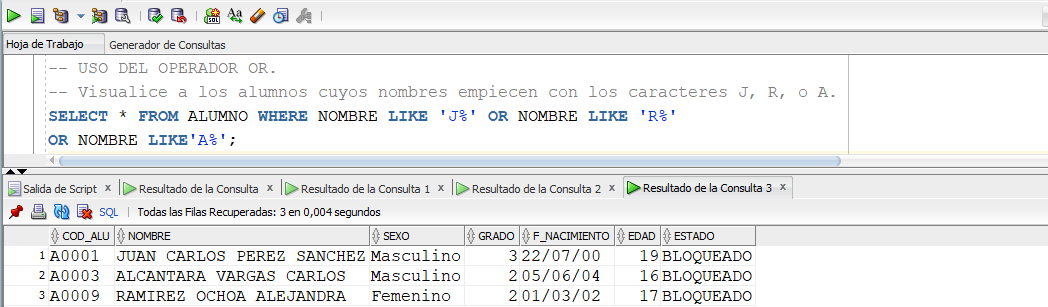


**Ejemplo 10: USO DEL OPERADOR OR.**

Visualice a los alumnos cuyos nombres empiecen con los caracteres J, R, o A.

**SELECT** \* **FROM** ALUMNO

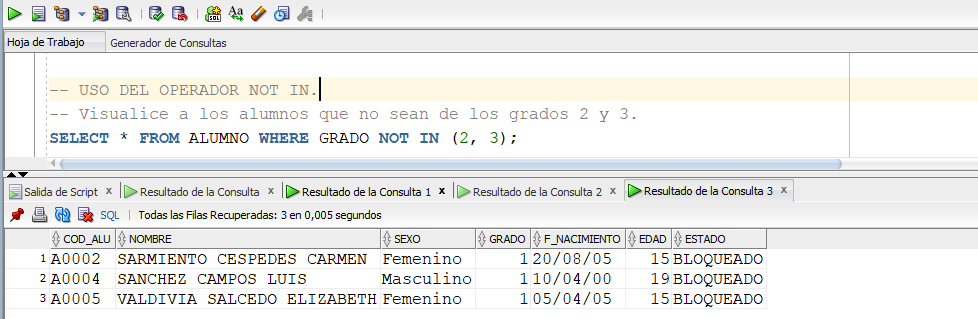
**WHERE** NOMBRE LIKE 'J%' **OR** NOMBRE LIKE 'R%' **OR** NOMBRE LIKE'A%';



**Ejemplo 11: USO DEL OPERADOR NOT IN**.

Visualice a los alumnos que no sean de los grados 2 y 3.

**SELECT** \* **FROM** ALUMNO **WHERE** GRADO **NOT IN** (2, 3);



**FUNCIONES AGREGADAS. SUM, MAX, MIN, AVG, COUNT**

Las funciones de grupo funcionan en juegos de filas para proporcionar un resultado por grupo.

Estos son los datos almacenados en la tabla alumno, se observa la columna Edad.

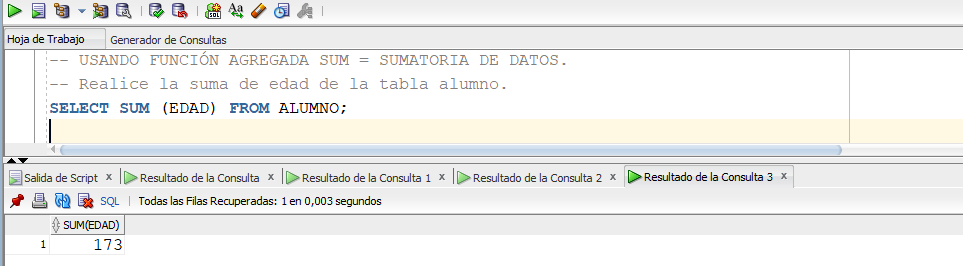


**Ejemplo 1:** USANDO FUNCIÓN AGREGADA SUM = SUMATORIA DE DATOS.

Realice la suma de edad de la tabla alumno.

Para saber cuál es la suma de las edades, se debe realizar la siguiente consulta:

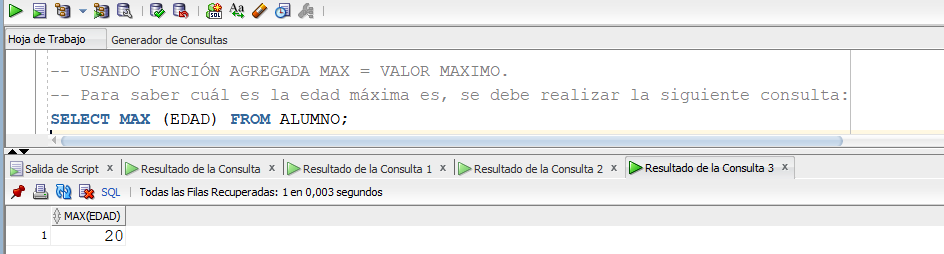
**SELECT** **SUM** (EDAD) **FROM** ALUMNO;



**Ejemplo 2: USANDO FUNCIÓN AGREGADA MAX = VALOR MAXIMO.**

Para saber cuál es la edad máxima es, se debe realizar la siguiente consulta:

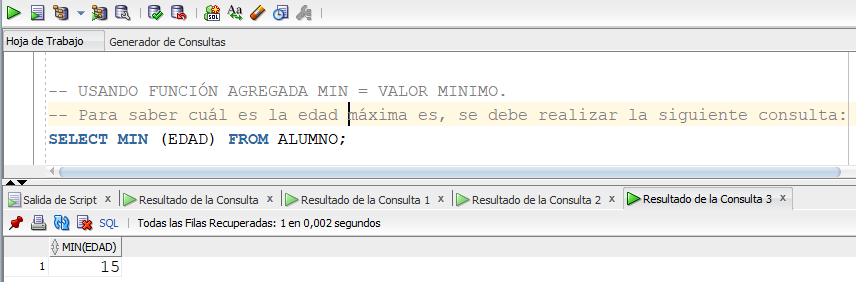
**SELECT** **MAX** (EDAD) **FROM** ALUMNO;



**Ejemplo 3: USANDO FUNCIÓN AGREGADA MIN = VALOR MINIMO.**

Para saber cuál es la edad máxima es, se debe realizar la siguiente consulta:

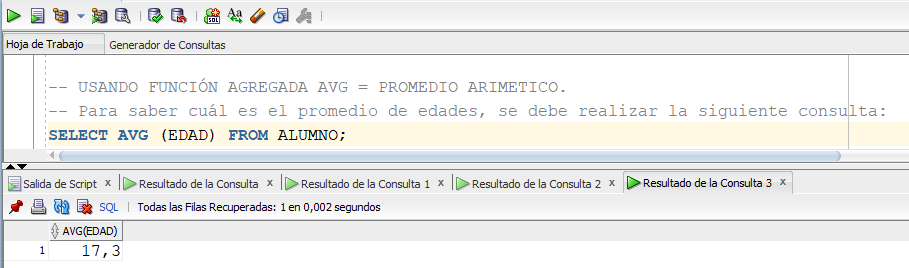
**SELECT** **MIN** (EDAD) **FROM** ALUMNO;



**Ejemplo 4: USANDO FUNCIÓN AGREGADA AVG = PROMEDIO ARIMETICO.**

Para saber cuál es el promedio de edades, se debe realizar la siguiente consulta:

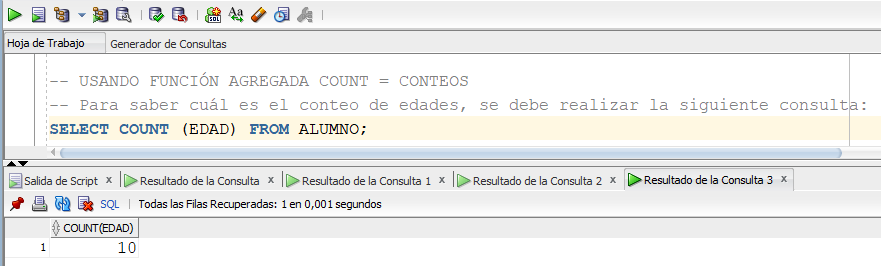
**SELECT** **AVG** (EDAD) **FROM** ALUMNO;



**Ejemplo 5: USANDO FUNCIÓN AGREGADA COUNT = CONTEOS**

Para saber cuál es el conteo de edades, se debe realizar la siguiente consulta:

**SELECT** **COUNT** (EDAD) **FROM** ALUMNO;



**CLAUSULA ORDER BY.**

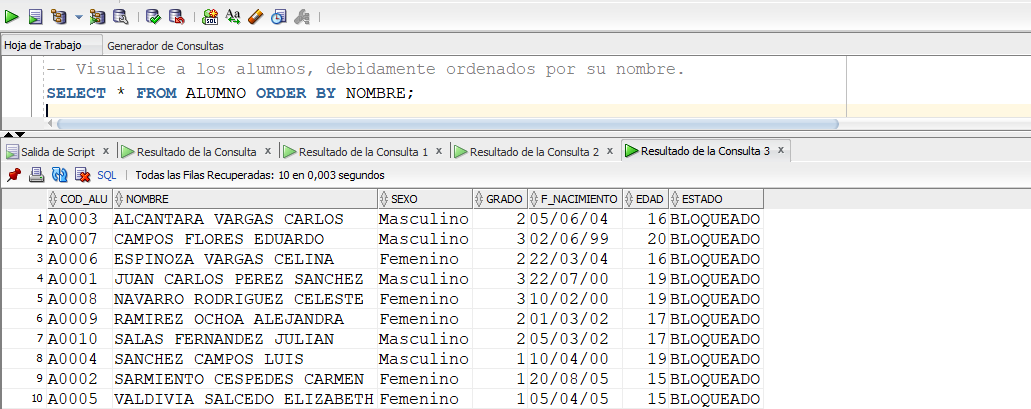
* Ordenar las filas recuperadas con la cláusula ORDER BY:
* ASC: ordena los registros de manera ascendente, Caracteres (A..Z, a...z) y Numéricos (1..n), si no se especifica será valor por defecto
* DESC: ordena los registros de manera descendente, Caracteres (Z .. A, z...a) y Numéricos (n..1), su especificación es obligatoria, si se decide usarlo.
* La cláusula ORDER BY es la última en una sentencia SELECT:

Ejemplo 1:

NO ES NECESARIO USAR ASC, ES EL VALOR POR DEFECTO SI NO SE USA.

Visualice a los alumnos, debidamente ordenados por su nombre.

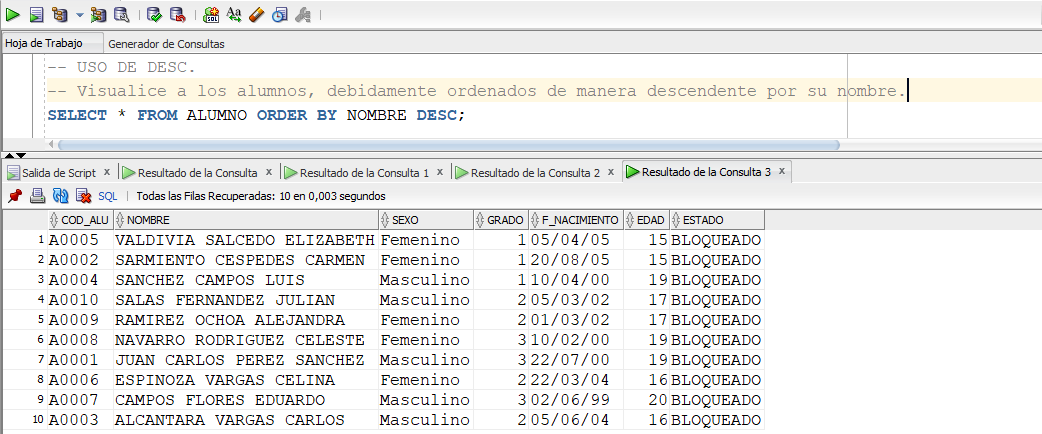
**SELECT** \* **FROM** ALUMNO **ORDER BY** NOMBRE;



**Ejemplo 2: USO DE DESC.**

Visualice a los alumnos, debidamente ordenados de manera descendente por su nombre.

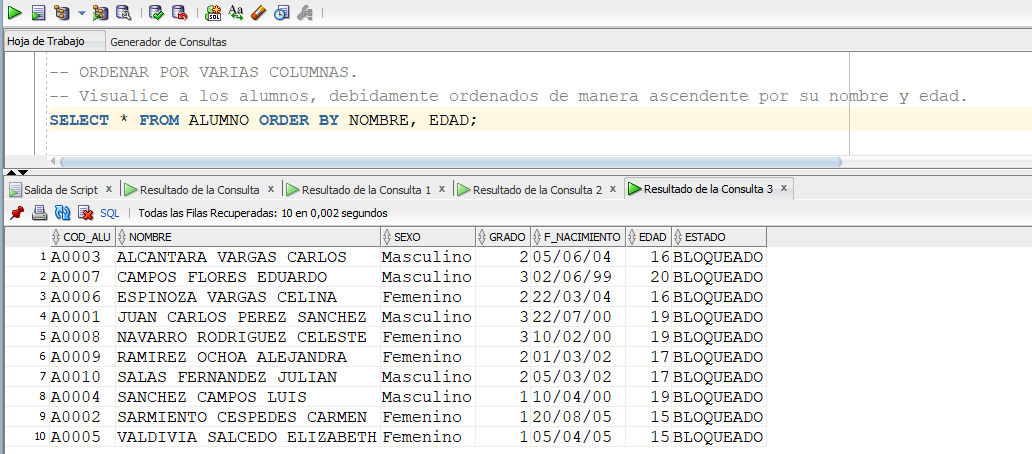
**SELECT** \* **FROM** ALUMNO **ORDER BY** NOMBRE **DESC**;



**Ejemplo 3: ORDENAR POR VARIAS COLUMNAS.**

Visualice a los alumnos, debidamente ordenados de manera ascendente por su nombre y edad.

**SELECT** \* **FROM** ALUMNO **ORDER BY** NOMBRE, EDAD;



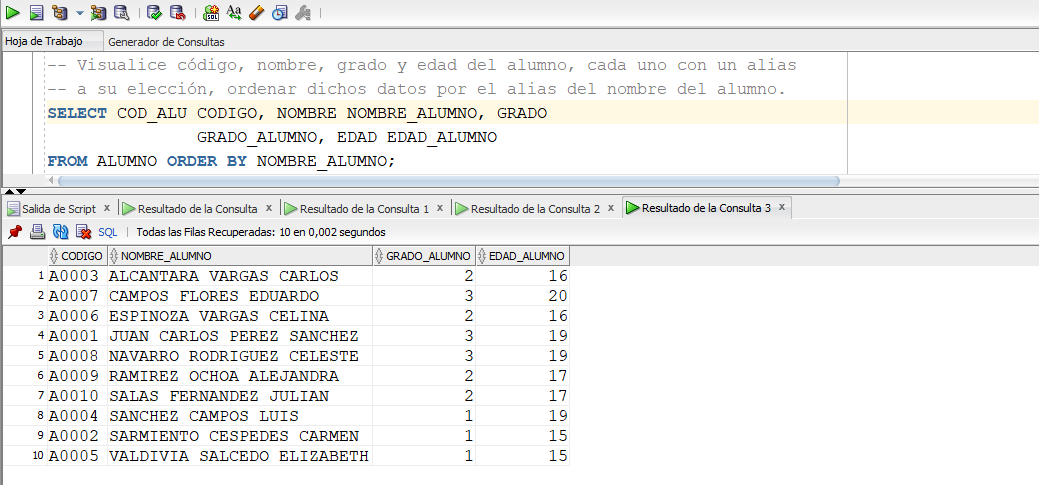
**Ejemplo 4: ORDENAR POR ALIAS DE COLUMNA.**

Visualice código, nombre, grado y edad del alumno, cada uno con un alias a su elección, ordenar dichos datos por el alias del nombre del alumno.

**SELECT** COD\_ALU **CODIGO**, NOMBRE **NOMBRE\_ALUMNO**, GRADO,

**GRADO\_ALUMNO**, EDAD **EDAD\_ALUMNO**

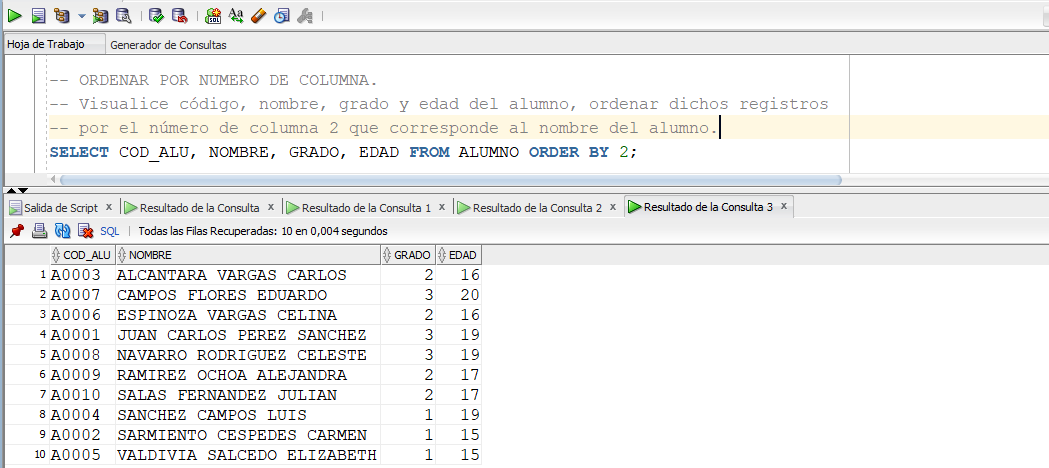
**FROM** ALUMNO **ORDER BY** NOMBRE\_ALUMNO;



**Ejemplo 5: ORDENAR POR NUMERO DE COLUMNA.**

Visualice código, nombre, grado y edad del alumno, ordenar dichos registros por el número de columna 2 que corresponde al nombre del alumno.

**SELECT** COD\_ALU, NOMBRE, GRADO, EDAD **FROM** ALUMNO **ORDER BY 2**;



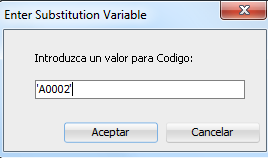
**USO DE VARIABLES DE SUSTITUCIÓN EN NUESTRAS CONSULTAS &.**

Utilizar una variable prefijada con un ampersand (&) para solicitar al usuario un valor:

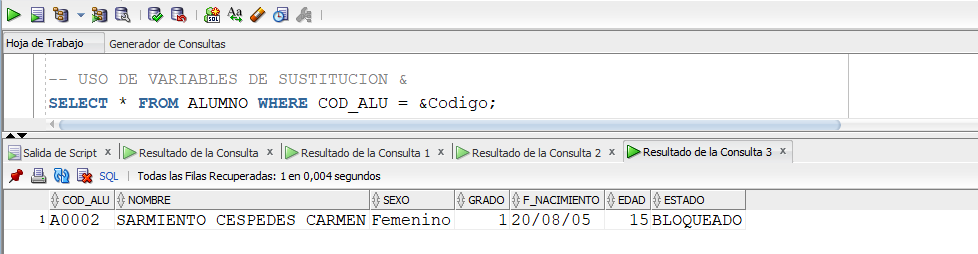
**Ejemplo 1:**

Visualice los datos de cualquier alumno usando como dato de búsqueda al código de alumno, este debe ser ingresado desde teclado a través de una ventana de acceso, en este ejemplo usaremos al código A0002.

**SELECT** \* **FROM** ALUMNO **WHERE** COD\_ALU = **&**Codigo;



**SALIDA POR PANTALLA:**

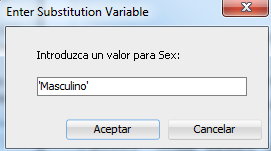


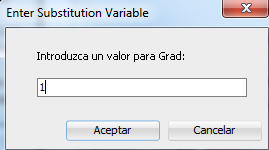
**Ejemplo 2:**

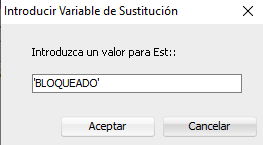
Visualice a los alumnos de cualquier sexo, grado, y estado, ordenarlos por cualquier columna.

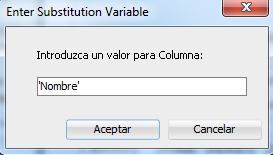
**SELECT** \* **FROM** ALUMNO **WHERE** SEXO = **&**Sex **AND** GRADO **=** **&**Grad **AND**

ESTADO = **&**Est **ORDER BY** **&**Columna;

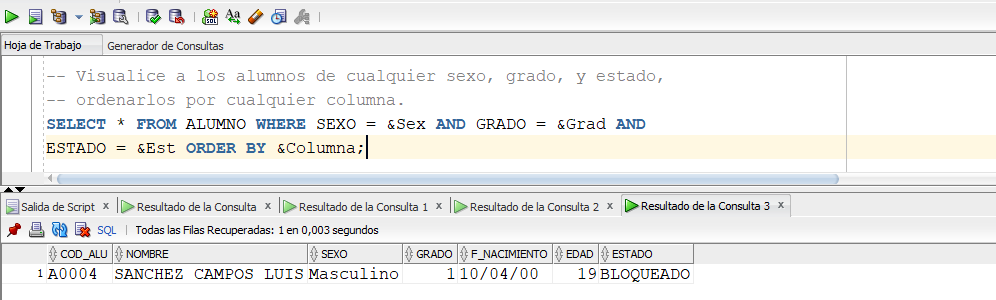








**SALIDA POR PANTALLA:**



**CLAUSULAS GROUP BY y HAVING.**

Genera grupos de datos permite condicionar los grupos.

**Ejemplo recreativo, para entender mejor, ¡QUE ES UN GRUPO?**

Tenemos un mundo de esferas de colores y estos tienen un valor referencial.

Para entender mejor lo que es un GRUPO, debemos separarlos, es decir clasificarlos y obtendremos algo así:



**Tenemos 5 grupos de esferas, ¿QUE PODEMOS HACER CON ESOS GRUPOS?**

Voy a tomar un Grupo: EN ESTE CASO EL AMARILLO.



**¿Me pregunto qué puedo hacer con este grupo MARILLO?**

Sumarlos: 1 + 2 + 2 + 5 + 5 + 1 + 5 = 21.

Contarlos: 1, 2, 2, 5, 5, 1, 5 = Son 7 Elementos.

Hallar el Máximo: 1, 2, 2, 5, 5, 1, 5 = El Mayor es 5.

Hallar el Mínimo: 1, 2, 2, 5, 5, 1, 5 = El Menor es: 1.

Promedio Aritmético: (1 + 2 + 2 + 5 + 5 + 1 + 5) / 7 = 3

**LO MISMO PODRÍA HACER CON LOS DEMÁS GRUPOS.**

¿Podríamos ser más específicos en los grupos? CLARO QUE SI.

**AQUÍ ENTRA EL USO DE LA CLAUSULA HAVING**.

Obtener la sumar valores del grupo amarillo y rojo por separado, exceptuando al verde.



**Es decir, Grupo Amarillo: 1 + 2 + 2 + 5 + 5 + 1 + 5 = 21**

**Es decir, Grupo Amarillo: 6 + 1 + 5 + 2 + 1 + 7 + 4 + 2 = 28**

(aquí entra el uso de la cláusula HAVING)

**La cláusula HAVING condiciona que grupos desea visualizar.**

**EJEMPLOS DE GRUPOS Y CONDICIONES DE GRUPO.**

**GROUP BY – HAVING.**

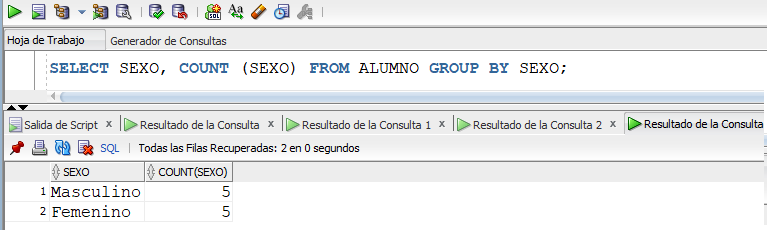
**Ejemplo 1: uso de solo GROUP BY**

Tenemos los siguientes datos, en la tabla **ALUMNO**, aquí observamos que existen tanto alumnos varones como damas, se pide una consulta que muestre el conteo y los diferencie.



**SOLUCIÓN:**

**SELECT** SEXO, **COUNT** (SEXO) **FROM** ALUMNO **GROUP BY** SEXO;

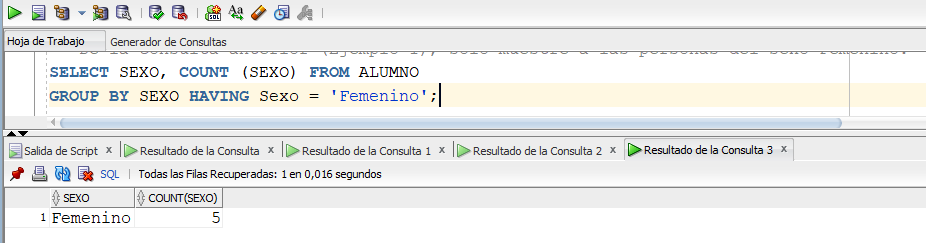


**Ejemplo 2: uso de GROUP BY – HAVING.**

De la consulta anterior (Ejemplo 1), solo muestre a las personas del sexo Femenino.

**SELECT** SEXO, **COUNT** (SEXO) **FROM** ALUMNO

**GROUP BY** SEXO **HAVING** Sexo = ‘Femenino’;

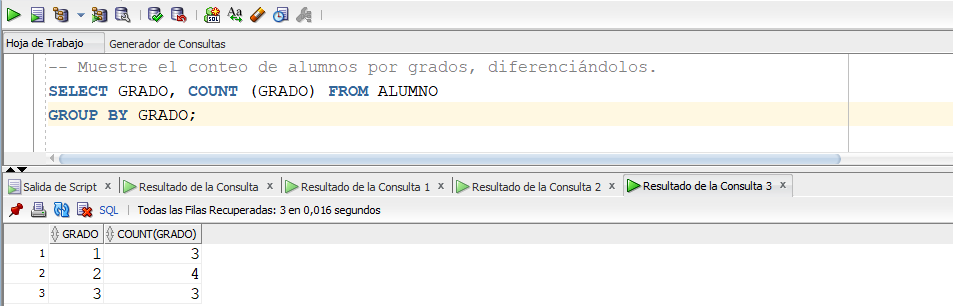


**Ejemplo 3:**

Muestre el conteo de alumnos por grados, diferenciándolos.

**SELECT** GRADO, **COUNT** (GRADO) **FROM** ALUMNO

**GROUP BY** GRADO;

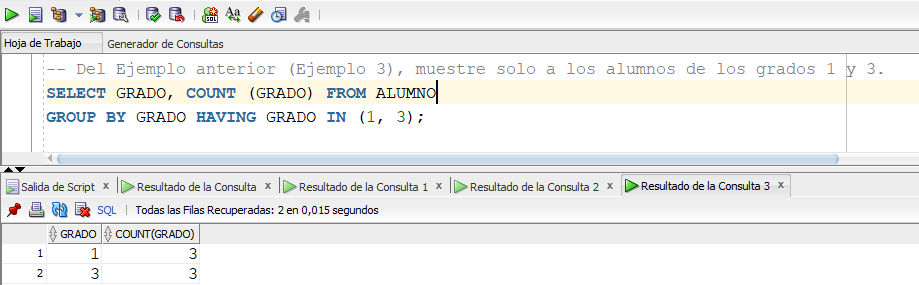


**Ejemplo 4:**

Del Ejemplo anterior (Ejemplo 3), muestre solo a los alumnos de los grados 1 y 3.

**SELECT** GRADO, **COUNT** (GRADO) **FROM** ALUMNO

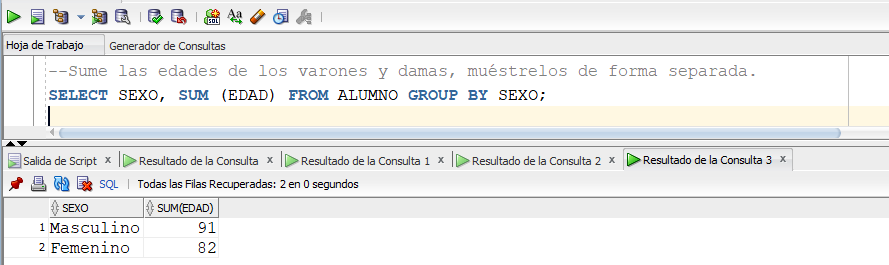
**GROUP BY** GRADO **HAVING** GRADO **IN** (1, 3);



**Ejemplo 5:**

Sume las edades de los varones y damas, muéstrelos de forma separada.

**SELECT** SEXO, **SUM** (EDAD) **FROM** ALUMNO **GROUP BY** SEXO;

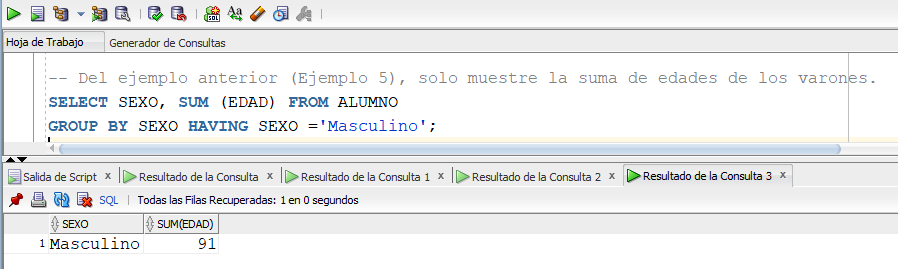


**Ejemplo 6:**

Del ejemplo anterior (Ejemplo 5), solo muestre la suma de edades de los varones.

**SELECT** SEXO, **SUM** (EDAD) **FROM** ALUMNO

**GROUP BY** SEXO **HAVING SEXO** ='Masculino';



**FUNCIONES DE TEXTO.**

**CONCAT** (cadena1, cadena2)

Devuelve "cadena1" concatenada con "cadena2".

**Por Ejemplo:**

SELECT CONCAT ('REY','ARTURO') FROM DUAL;

**Devuelve**

REYARTURO

**Tener en cuenta que CONCAT solo permite 2 argumentos es decir solo 2 cadenas**

**¿Si quiere concatenar más caracteres?**

Usamos “||” - (alt+124) =|

**Ejemplo:**

SELECT 'LLAVES PRINCIPALES'||' '||' Y '||' '||'FORANEAS' FROM DUAL;

**Devuelve:**

LLAVES PRINCIPALES Y FORANEAS

**SUBSTR** (char, m [, n])

Devuelve una sub Cadena de char de **n** caracteres a partir de la posición m.

**Ejemplo:**

**SELECT** **SUBSTR** ('CONSTANTINOPLA', 5, 5) **FROM** DUAL;

**Devuelve:**

TANTI

¿Por qué? -- si ubicamos la posición 5(m) desde la izquierda de la cadena, y contamos a partir de dicha posición 5 caracteres(n) a la derecha obtenemos la sub-cadena “TANTI”

**LENGTH** (char)

Devuelve un entero que representa la longitud de una cadena, es decir cuenta carácter por carácter incluido espacios en blanco dentro de una cadena. (los espacios en blanco deben estar entre los caracteres).

**Ejemplo:**

**SELECT** **LENGTH** (‘MJOLNIER') **FROM** DUAL;

**Devuelve:**

8

**INSTR** (char1, char2)

Devuelve la posición de inicio de la enésima aparición de char2 en Char1.

**Ejemplo:**

**SELECT** **INSTR** ('REY ARTURO','AR') **FROM** DUAL;

**Devuelve**:

5

**LPAD** (char1, n [, char2])

Rellena a una cadena por el lado izquierdo, hasta la longitud n con el carácter definido (char2)

**Ejemplo:**

**SELECT** **LPAD** (‘PEPE GRILLO', 20,'-') **FROM** DUAL;

**Devuelve:**

----------PEPE GRILLO

**RPAD** (char1, n [, char2])

Rellena la cadena por la derecha hasta la longitud n con el carácter definido (char2)

**Ejemplo**:

**SELECT** RPAD ('PEPE GRILLO', 20,'-') **FROM** DUAL;

**Devuelve:**

PEPE GRILLO----------

**TRIM** (char)

Esta función permite quitar los espacios en blanco a los extremos de una cadena.

**Ejemplo:**

**SELECT** **TRIM** (' JOJOJO ') **FROM** DUAL;

**Devuelve:**

JOJOJO

**RESUMEN DE LAS FUNCIONES DE MANIPULACIÓN DE CARACTERES:**



**FUNCIONES MATEMÁTICAS:**

**ROUND**: Redondea el valor a los decimales especificados.

**Ejemplo**:

**SELECT** **ROUND** (45.926, 2) **FROM** DUAL;

**Devuelve:**

45.93

**TRUNC**: Trunca el valor a los decimales especificados.

**SELECT** **TRUNC** (45.926, 2) FROM DUAL;

**Devuelve**:

45.92

**MOD**: Devuelve el resto de la división.

**SELECT** **MOD** (1600, 300) **FROM** DUAL;

**Devuelve**:

100

**ABS**: Calcula el valor absoluto de n.

**SELECT** **ABS** (-16) **FROM** DUAL;

16

**CEIL**: Calcula el menor número entero mayor o igual que n.

**SELECT** CEIL (16.7) **FROM** DUAL;

**Devuelve**:

17

**FLOOR**: Calcula el mayor número entero menor o igual que n.

**SELECT** **FLOOR** (16.7) **FROM** DUAL;

**Devuelve**:

16

**POWER**: Devuelve m elevado a la n potencia, n debe ser entero

**SELECT** **POWER** (3, 2) **FROM** DUAL;

**Devuelve**:

9

**FUNCIONES DE FECHA:**

|  |  |
| --- | --- |
| Función | Descripción |
| MONTHS\_BETWEEN | Numero de meses entre dos fechas. |
| ADD\_MONTHS | Suma meses de calendario a una fecha. |
| NEXT\_DAY | Siguiente día de la fecha especificada. |
| LAST\_DAY | Ultimo día del mes. |
| ROUND | Redondea la fecha. |
| TRUNC | Trunca la fecha. |

Cuantos meses existen entre el 11 de enero del 2019 y el 01 de septiembre del 2019

**SELECT** **MONTHS**\_**BETWEEN** ('01/09/2019','11/01/2019') **FROM** DUAL;

**Resultado**: 7,67741935483870967741935483870967741935

Se puede redondear la fecha al igual que las funciones numéricas de la siguiente forma

**SELECT** **ROUND** (**MONTHS**\_**BETWEEN** ('01/09/2019','11/01/2019'), 1) **FROM** DUAL;

**Resultado**: 7,7

Aumentar 6 meses al 12 de Agosto del 2019

**SELECT** **ADD\_MONTHS** (‘12/08/2019’, 6) **FROM** DUAL;

**Resultado**: 12/02/2020

Qué fecha cae el siguiente viernes después del 12 de agosto del 2019

**SELECT** **NEXT\_DAY** ('12/08/2019','VIERNES') **FROM** DUAL;

**Resultado**: 16/08/2019

Cuál es la fecha del último día del mes correspondiente a Febrero del 2019

**SELECT** **LAST\_DAY** ('01/02/2019') **FROM** DUAL;

**Resultado**: 28/02/2019

Como obtener la fecha del sistema.

**SELECT** **SYSDATE** **FROM** DUAL;

**Resultado**: 12/08/2019

Sumarle 3 días a la fecha actual

**SELECT** **SYSDATE** + 3 **FROM** DUAL;

**Resultado**: 15/08/2019

**CONVERSION A TIPOS DE DATOS.**

**DATE a CHAR (TO\_CHAR):**

Convertir la fecha del día de hoy al formato DD/MM/YYYY HH24:MI:SS

**SELECT** **TO\_CHAR** (**SYSDATE**,'DD/MM/YYYY HH24:MI:SS') **FROM** DUAL;

**Resultado**: 12/08/2019 01:43:11

Los formatos pueden ser:

|  |  |
| --- | --- |
| YYYY | Año completo en números |
| YEAR | Años en letra |
| MM | Valor de dos dígitos para el mes |
| MONTH | Nombre completo del mes. |
| MON | Abreviatura de tres letras del mes. |
| DY | Abreviatura de tres letras del día de la semana. |
| DAY | Nombre completo del día de la semana. |
| DD | Dia del mes en número. |

**CHAR a DATE (TO\_DATE):**

Convertir la cadena 23/10/2012 al formato fecha para realizar operaciones tipo fecha

**SELECT** **TO\_DATE** ('12/08/2019','DD/MM/YYYY') **FROM** DUAL;

Resultado: 12/08/2019 --(tipo de dato date)

**CHAR a NUMBER (TO\_NUMBER):**

Convertir el texto '1567,7' a formato numérico para realizar operaciones matemáticas

**SELECT** **TO\_NUMBER** ('1567, 7’) **FROM** DUAL;

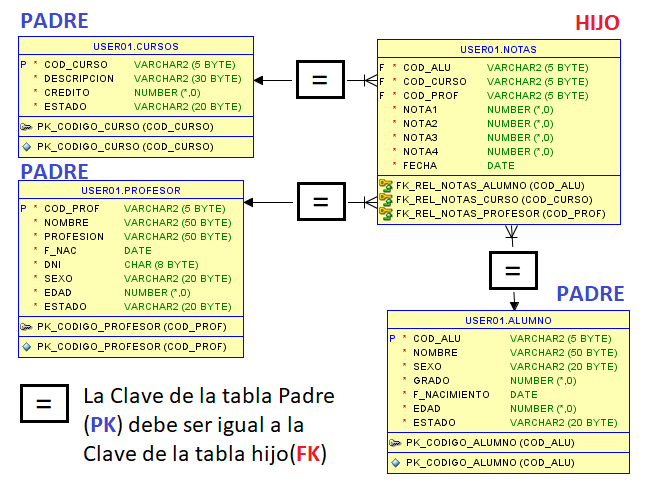
**Resultado**: 1567,7 --(tipo de dato number)

**CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA**

El objetivo de este capítulo es comprender el uso de consultas, de esta manera el estudiante se compenetra más con la base de datos de Oracle, al analizar la información, resultado de las búsquedas realizadas de manera exitosa. Mediante el uso de las herramientas SQL \*PLUS y PL/SQL.

**ACTIVIDAD VIRTUAL**

Revisar y analizar el material presentado del tema 03.

**DE ACUERDO AL SIGUIENTE MODELO RELACIONAL**.

**REALICE LAS SIGUIENTES CONSULTAS:**

* FECILITE UNA CONSULTA, USANDO LA CLAUSULA WHERE, LIKE, AND, OR, NOT IN, IN
* FACILITE UNA CONSULTA, USANDO CAMPOS CALCULADOS
* FACILITE UNA CONSULTA, CON FUNCIONES AGREGADAS.
* FACILITE UNA CONSULTA, USANDO VARIABLES DE SUSTITUCION.
* FACILITE UNA CONSULTA, USANDO GRUPOS.
* FACILITE UNA CONSULTA, USANDO GRUPOS CONDICIONADOS.