Base de Datos I

Curso No: TWB22B

Lenguaje de Consulta Estructurado (SQL)

Unidad 1:

Lenguaje de Consulta Estructurado - Fundamentos

Objetivos del Aprendizaje

- Explicar SQL
- Discutir sobre DDL, DML, DCL y DQL
- Escribir una sentencia SELECT simple y una sentencia SELECT condicional
- Indicar el uso de las funciones agregadas
- Listar los operadores lógicos y el orden de precedencia
- Discutir sobre los operadores usados en las sentencias SELECT

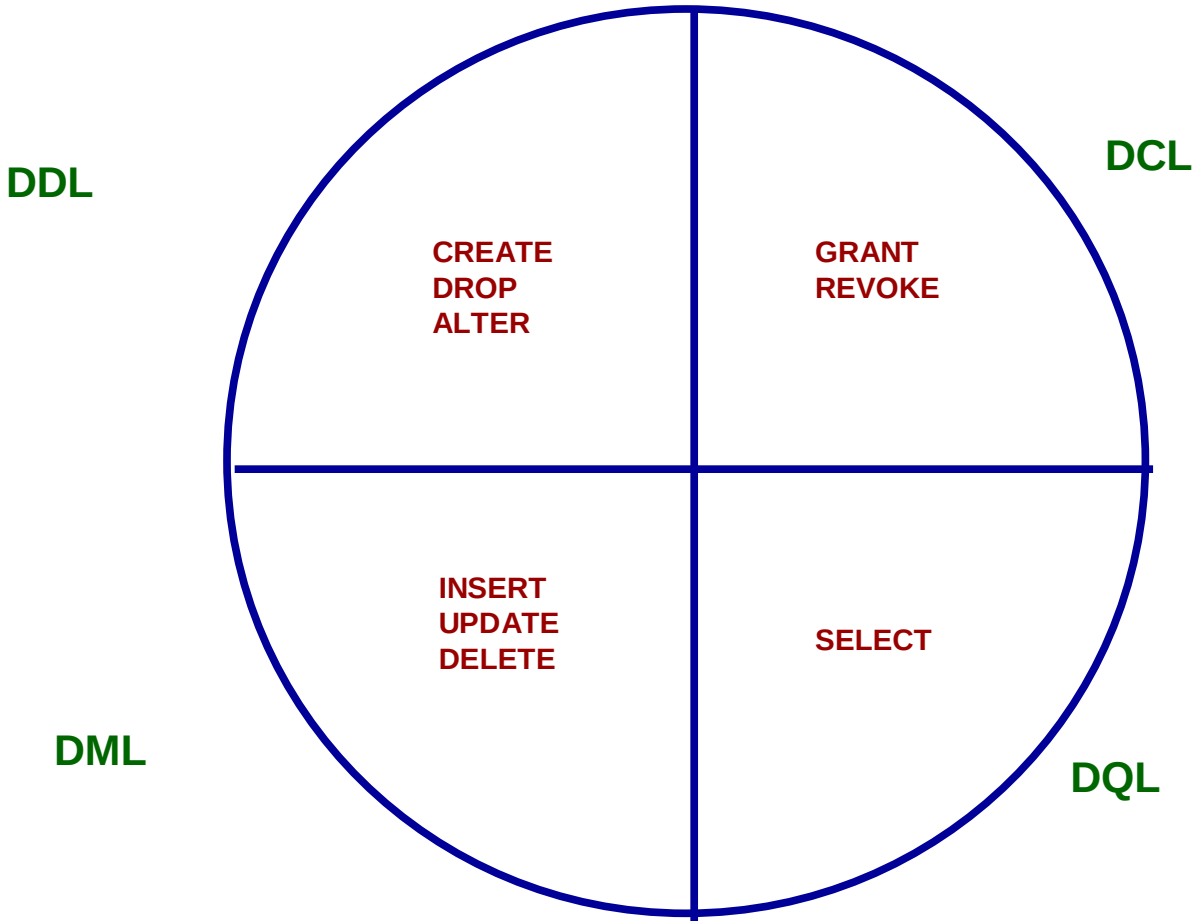
Historia del SQL

- Es un lenguaje usado para comunicarse con la base de datos
- Fue desarrollado originalmente por IBM en la década del 70
- Fue inicialmente llamado Sequel
- En 1986, ANSI presentó un nuevo estándar llamado SQL-86
- La versión actual es llamada SQL-99 (SQL3) estándar
- Es ampliamente aceptado como un estándar en implementaciones de RDBMS

Partes del SQL

- Lenguaje de Definición de Datos (DDL):Proporciona comandos para definir los objetos de la base de datos
- Lenguaje de Manipulación de Datos(DML): Proporciona comandos para insertar, eliminar y modificar registros en la(s) tabla(s)
- Lenguaje de Control de Datos (DCL): Proporciona comandos para manejar y controlar datos
- Lenguaje de Consulta de Datos (DQL): Proporciona comandos para recuperar datos desde tablas

Partes del SQL



DDL : Lenguaje de Definición de Datos

CREATE

- Crea objetos de la base de datos
- Algunos ejemplos son tablas, vistas e índices

ALTER

- Altera la estructura de una tabla
- Modifica o agrega a la estructura de datos existente

DROP

- Elimina objetos de la base de datos

Comando CREATE

- Crea objetos de la base de datos
- Tablas, vistas e índices son algunos de los ejemplos
- Describe la estructura del objeto de la base de datos

```
CREATE TABLE juguetes(
id_comprador INTEGER NOT NULL,
producto CHARACTER(40) NOT NULL,
precio DOUBLE);
```

Comando ALTER

- Altera la estructura de una tabla

- Modifica o agrega a la estructura de datos existente

```
ALTER TABLE juguetes
ADD COLUMN id_vendedor integer;
```

- El comando agrega una nueva columna **id_vendedor** de tipo de dato numérico

Comando DROP

- Elimina Objetos de la Base de Datos

- Ejemplos:

```
DROP TABLE juguetes;
DROP VIEW vista_juguetes;
DROP INDEX indice_juguetes;
```

DCL: Lenguaje de Control de Datos

- Controla y monitorea el acceso a los datos
- Garantiza la seguridad de los datos
- Previene el acceso ilegal a los datos
- Algunos de los comandos SQL son:
  - ü **GRANT**
  - ü **REVOKE**

Comandos GRANT y REVOKE

- **GRANT:**

  GRANT SELECT ON JUGUETES TO USER STUDENT1;

  GRANT SELECT ON JUGUETES TO GROUP SALON1;

- **REVOKE:**

  REVOKE SELECT ON JUGUETES FROM USER STUDENT1;

  REVOKE SELECT ON JUGUETES FROM GROUP SALON1;

DML: Lenguaje de Manipulación de Datos

- Manipula los datos
- Agrega, elimina y actualiza los valores
- Los Comandos SQL son:
  - ü **INSERT**
  - ü **DELETE**
  - ü **UPDATE**

¿Cómo Agregar Datos a la Base de Datos?


```
INSERT INTO nombre_tabla(columna1, columna2)
  VALUES (valor1,valor2)
```

Agregar Datos

- Una tabla de datos contiene una lista de columnas ordenadas y una lista correspondiente de valores ordenados

- La sentencia **`INSERT se`** puede escribir como sigue:

```
INSERT INTO juguetes
(id_comprador,id_vendedor, producto, precio)
VALUES (21,01, 'Barbie',200.00);
```

Agregar Datos

- **Otra forma de escribir la sentencia INSERT:**

Correcto:

```
INSERT INTO juguetes
VALUES (21, 'Barbie', 200.00, 01);
```

Incorrecto:

```
INSERT INTO juguetes
VALUES (21, 01, 'Barbie', 200.00);
```

Eliminar Datos

DELETE FROM nombre_tabla

- Una fila puede ser eliminada de la base de datos. Por ejemplo:

```
DELETE FROM juguetes
WHERE producto = 'Barbie';
```

- Si la **CONDICIÓN** no es especificada, todas las filas serán eliminadas.

Eliminar Datos ... 2

- Se pueden eliminar filas que contienen datos específicos

```
DELETE FROM juguetes
WHERE producto = 'Barbie' AND
id_comprador = 02 AND id_vendedor = 22;
```

Actualizar Datos

UPDATE nombre_tabla SET Col1=valor1, Col2=Valor2

Se pueden asignar datos a campos:

```
UPDATE juguetes
SET precio = 500.00
WHERE producto = 'Silla';
```

DQL: Lenguaje de Consulta de Datos

Simple SELECT

- Recupera datos de las tablas
- Puede ser un simple **SELECT** o uno condicional
- Selecciona datos de todas las columnas de la tabla y todos los registros de la tabla

```
SELECT * FROM nombre_tabla;
```

- Selecciona datos de columnas específicas de la tabla

```
SELECT nombrecol1, nombrecol2, …, nombrecoln
FROM nombre_tabla;
```

Selección Condicional

Se especifica la sentencia **SELECT** condicional usando la cláusula **WHERE:**

```
SELECT *
FROM NOMBRE_TABLA
WHERE NOMBRE_COL = VALOR;
```

Los Operadores Relaciónales

Operador	Significado
=	Igual a
<	Menor que
<=	Menor o Igual a
>	Mayor que
>=	Mayor o Igual a
<> o !=	Diferente

Ejemplos de Selección Condicional

```
SELECT idnoempleado
FROM EstadisticasDeEmpleados
WHERE salario >= 50000;
```

```
SELECT idnoempleado
FROM EstadisticasDeEmpleados
WHERE posicion = 'Gerente';
```

Funciones Agregadas

- Son también llamadas *funciones de grupo*
- Trabajan sobre conjuntos de filas y devuelven resultados
- Las funciones más importantes son: **SUM, AVG, MAX, MIN** y **COUNT**

```
SELECT SUM(salario)
FROM estadisticasdeempleados;

SELECT AVG(salario)
FROM estadisticasdeempleados;

SELECT MIN(beneficios)
FROM estadisticasdeempleados
WHERE posicion = 'Gerente';
```

Funciones Agregadas... 2

Ejemplo1

```
SELECT COUNT(*)
FROM estadisticasdeempleados
WHERE posicion = 'Personal';
```

- La función **COUNT** cuenta el número de empleados que pertenecen a una posición llamada `'Personal'`
- El * se puede usar sólo con la función **COUNT** y no con otras funciones agregadas

Ejemplo2:

```
SELECT MAX(SALARY)
FROM estadisticasdeempleados;
```

- La función **MAX()** retorna el valor máximo para una columna dada entre el conjunto de filas seleccionadas

Condiciones Compuestas y Operadores Lógicos

- **El Operador AND**

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE salario > 40000 AND posicion = 'Personal';
```

- **El Operador OR**

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE salario < 40000 OR beneficios < 10000;
```

Combinar los Operadores AND y OR

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE posicion = 'Gerente' AND
salario > 60000 OR beneficios > 12000;
```

- La precedencia es importante
- El operador **AND** precede al operador **OR**
- Limita las filas a las personas que son gerentes y ganan un salario mayor que $60,000
- Limita las filas a las personas del paso anterior o las personas que ganan beneficios mayores que $12,000

El Operador IN

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE posicion ='Gerente'
OR posicion= 'Personal';
```

- Lista todos los empleados que son gerentes o pertenecen a la posición Personal
- Puede ser escrito usando el operador **IN**

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE posicion
IN ('Gerente', 'Personal');
```

El Operador BETWEEN

- Comprueba si los valores están dentro de un rango dado
- Encuentra a los empleados que caen bajo un rango particular de salario ($30,000 a $50,000)

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE salario >= 30000
AND salario <= 50000;
```

- Usando operador **BETWEEN**

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE salario
BETWEEN 30000 AND 50000;
```

El Operador NOT

- Encuentra los empleados que no caen en el rango de salario de $30,000 a $50,000

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE salario
NOT BETWEEN 30000 AND 50000;
```

- Usando el operador **NOT** con el operador **IN**

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE posicion NOT IN ('Gerente');
```

El Operador LIKE

- Verifica patrones dentro de cadenas, los compara y muestra los resultados basados en el tipo requerido
- La siguiente sentencia lista todos los empleados cuyos apellidos empiezan con ‘S’

```
SELECT idnoempleado
FROM estadisticasdeempleados
WHERE apellido LIKE 'S%';
```

El Operador CONCATENATION

- Combina dos cadenas o dos campos (del tipo cadena de caracteres).
- Es denotado por | |
- Un ejemplo es:

  Concatenar el nombre y el apellido del empleado y presentar el resultado

```
SELECT nombre || '.' || apellido
FROM direcciondeempleado
```

Alias a los Nombres de Columnas

```
SELECT nombre || '.' || apellido
AS "Nombre Completo"
FROM direcciondeempleado;
```

- Esta sentencia crea un nombre alias, `Nombre Completo`
- Ayuda a crear títulos significativos cuando se imprimen las consultas

La Cláusula ORDER BY

- Da formato a la salida basándose en un campo y en un cierto orden, descendente o ascendente
- Por defecto lista los datos en orden ascendente

```
SELECT * FROM estadisticasdeempleados
ORDER BY salario ASC;

SELECT * FROM estadisticasdeempleados
ORDER BY salario DESC;

SELECT * FROM estadisticasdeempleados
ORDER BY salario;

SELECT * FROM estadisticasdeempleados
ORDER BY posicion ASC, salario DESC;
```

Manejo de Valores NULL

- Se usa cuando un campo escogido no tiene algún valor conocido válido
- Evalúa a sí mismo en cualquier expresión
- Un ejemplo es

  4+NULL = NULL

  2*3*NULL = NULL

- Uso de NULL en una sentencia

```
SELECT *
FROM estudiante
WHERE proyecto IS NULL;
```

Cláusula DISTINCT

Se usa para listar valores únicos (Filas únicas)

```
SELECT DISTINCT posicion
FROM estadisticasdeempleados;
```

Resumen

- Se explicó qué es SQL
- Se discutió acerca de DDL, DML y DCL
- Se aprendió a escribir una sentencia SELECT simple y una sentencia SELECT condicional
- Se presentó el uso de las funciones agregadas
- Se estudiaron los operadores lógicos y el orden de precedencia
- Se explicaron los operadores usados en las sentencias SELECT

Unidad 2:
Laboratorio sobre Fundamentos de SQL

Ejercicios de Laboratorio

Unidad 3:
SQL Avanzado

Objetivos del Aprendizaje

- Entender las diferentes operaciones con `JOIN`
- Comprender el uso de las cláusulas `GROUP BY` y `HAVING`
- Explicar cómo escribir subconsultas (subqueries)

Clave Primaria

- Es una columna ó varias columnas en una tabla
- Identifica de modo único un registro en la tabla
- Su valor no debe ser `NULL`
- Su valor debe ser único
- Puede ser también una **Clave Compuesta** donde más de una columna se combinan para actuar como la clave primaria

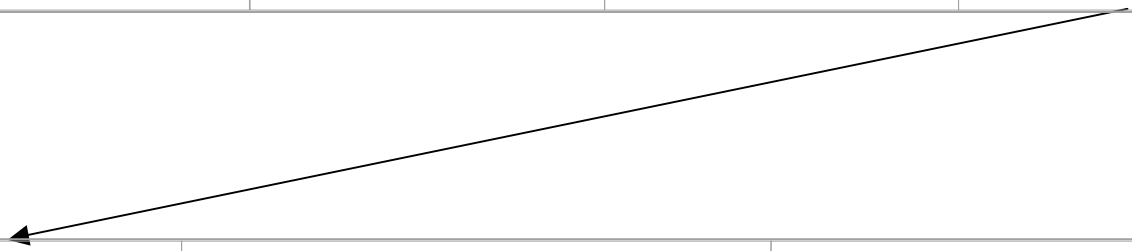
Clave Foránea

- Es una columna cuyos valores son dependientes de los valores existentes en otras tablas
- Una columna ó columnas es (son) una clave foránea sólo si se refiere a la clave primaria de otra tabla
- Se tiene una restricción de integridad referencial cuando una restricción está definida entre una(s) columna(s) de clave foránea y una(s) columna(s) de clave primaria
- Tabla padre es la tabla que contiene la(s) columna(s) de clave primaria, a la cual hace referencia la clave foránea
- Tabla hija es la tabla que contiene la clave foránea

Relación Padre-Hijo

Personal

no_personal	Nombre_personal	designacion	salario	no_division



Division

no_division	nombre_division	ubicacion

Relación Padre-Hijo …2

- La columna no_personal es la clave primaria en la Tabla Personal
- La columna no_division es la clave primaria en la Tabla Division
- La columna no_division en la Tabla Personal es la clave foránea, la cual hace referencia a la clave primaria en la Tabla Division
- La Tabla Division es la tabla padre, y la Tabla Personal es la tabla hija

La Condición JOIN (UNIR)

- Enlaza tablas de datos por medio de un valor de atributo común en ambas tablas
- Puede ser aplicada a dos o más tablas para poder recuperar registros de múltiples tablas
- Tipos de JOIN:
  - Cartesian
  - Natural / Equi (INNER JOIN)
  - Right Outer
  - Left Outer
  - Full Outer
  - Self

Cómo Trabaja el JOIN

Tabla Programa

Nombre_programa	no_curso	nota
Software Systems	SS G211	6
Computer Science	CS G311	5
Information Systems	IS G411	4
E-Commerce	EC G511	6
M-Commerce	MC G611	5

Cómo Trabaja el JOIN ... 2

Tabla Estudiante

nombre_estudiante	no_curso
John	SS G211
Robert	CS G311
Samuel	MC G611
Joseph	IS G511
David	CS G411
Gary	EC G211

La Sentencia Cartesian JOIN

Hace corresponder todas las filas de la primera tabla con todas las filas de la segunda tabla y presenta una combinación de todos los registros en ambas tablas.

```
SELECT programa.nombre_programa,
programa.no_curso, programa.nota,
estudiante.nombre_estudiante,
estudiante.no_curso
FROM programa, estudiante
```

Resultado del Cartesian JOIN

nombre_programa	no_curso	nota	nombre_estudiante	no_curso
Software Systems	SS G211	6	John	SS G211
Computer Science	CS G311	5	John	SS G211
Information Systems	IS G411	4	John	SS G211
E-Commerce	EC G511	6	John	SS G211
M-Commerce	MC G611	5	John	SS G211
Software Systems	SS G211	6	Robert	CS G311
Computer Science	CS G311	5	Robert	CS G311
Information Systems	IS G411	4	Robert	CS G311
E-Commerce	EC G511	6	Robert	CS G311
M-Commerce	MC G611	5	Robert	CS G311
Software Systems	SS G211	6	Samuel	MC G611
Computer Science	CS G311	5	Samuel	MC G611
Information Systems	IS G411	4	Samuel	MC G611
E-Commerce	EC G511	6	Samuel	MC G611
M-Commerce	MC G611	5	Samuel	MC G611
Software Systems	SS G211	6	Joseph	IS G511
.....	.....	.....	.....	.....

.

La Sentencia Natural/Equi JOIN (INNER JOIN)

- Hace corresponder los registros de la primera tabla con los de la segunda tabla basándose en la igualdad de los valores especificada en la condición `JOIN`
- Los registros que tienen una correspondencia exacta son extraídos de ambas tablas

```
SELECT programa.nombre_programa,
programa.no_curso, programa.nota,
estudiante.nombre_estudiante
FROM programa, estudiante
WHEREprograma.no_curso=estudiante.no_curso;
```

La Sentencia Natural/Equi JOIN..2 (INNER JOIN)

Otra manera de especificar la consulta Natural/Equi JOIN (INNER JOIN) es :

```
SELECT programa.nombre_programa,
programa.no_curso, programa.nota,
estudiante.nombre_estudiante
FROM programa
INNER JOIN estudiante
ON programa.no_curso=estudiante.no_curso;
```

Resultado del Natural/Equi `JOIN` (INNER JOIN)

Nombre_programa	no_curso	nota	nombre_estudiante
Software Systems	SS G211	6	John
Computer Science	CS G311	5	Robert
M-Commerce	MC G611	5	Samuel

La Sentencia Right Outer JOIN

- Hace corresponder los registros en la primera tabla con los de la segunda tabla.
- Se basa en la igualdad de valores que se especifica en la condición JOIN
- Incluye aquellos registros presentes en la segunda tabla que no están asociados con los registros de la primera tabla
- Ejemplo:

```
SELECT programa.nombre_programa,
programa.no_curso, programa.nota,
estudiante.nombre_estudiante,
estudiante.no_curso
FROM programa RIGHT OUTER JOIN estudiante
ON programa.no_curso=estudiante.no_curso;
```

Resultado del Right Outer JOIN

nombre_programa	no_curso	nota	nombre_estudiante	no_curso
Software Systems	SS G211	6	John	SS G211
Computer Science	CS G311	5	Robert	CS G311
M-Commerce	MC G611	5	Samuel	MC G611
NULL	NULL	NULL	Joseph	IS G511
NULL	NULL	NULL	David	CS G411
NULL	NULL	NULL	Gary	EC G211

La Sentencia Left Outer JOIN

- Hace corresponder los registros de la primera tabla con los de la segunda tabla basado en la igualdad de valores que es especificada en la condición JOIN

- Incluye aquellos registros presentes en la primera tabla, que no están asociados con los registros de la segunda tabla

```
SELECT programa.nombre_programa,
programa.no_curso, programa.nota,
estudiante.nombre_estudiante,
estudiante.no_curso
FROM programa
LEFT OUTER JOIN estudiante
ON programa.no_curso=estudiante.no_curso;
```

Resultado del Left Outer JOIN

nombre_programa	no_curso	Nota	nombre_estudiante	no_curso
Software Systems	SS G211	6	John	SS G211
Computer Science	CS G311	5	Robert	CS G311
Information Systems	IS G411	4	NULL	NULL
E-Commerce	EC G511	6	NULL	NULL
M-Commerce	MC G611	5	Samuel	MC G611

La Sentencia Full Outer JOIN

- Hace corresponder los registros de la primera tabla con los de la segunda tabla basándose en la igualdad de los valores especificada en la condición JOIN
- Es una combinación de Right Outer JOIN y Left Outer JOIN
- Se ocupa de todos los registros que no satisfacen la equivalencia entre los valores

```
SELECT programa.nombre_programa,
programa.no_curso, programa.nota,
estudiante.nombre_estudiante,
estudiante.no_curso
FROM programa FULL OUTER JOIN estudiante
ON programa.no_curso = estudiante.no_curso
```

Resultado del Full Outer JOIN

nombre_programa	no_curso	nota	nombre_estudiante	no_curso
Software Systems	SS G211	6	John	SS G211
Computer Science	CS G311	5	Robert	CS G311
M-Commerce	MC G611	5	Samuel	MC G611
Information Systems	IS G411	4	NULL	NULL
E-Commerce	EC G511	6	NULL	NULL
NULL	NULL	NULL	Joseph	IS G511
NULL	NULL	NULL	David	CS G411
NULL	NULL	NULL	Gary	EC G211

La Sentencia Self JOIN

- Por definición, un JOIN se aplica entre los valores de atributos comunes de dos o más tablas
- Puede ser realizado en la misma tabla si una tabla tiene dos atributos que comparten el mismo valor

```
SELECT  S.nombre, M.nombre
 FROM personal S, personal M
WHERE S.no_grnt = M.no_personal;
```

Resultado del Self JOIN

Tabla Personal

no_personal	nombre	no_grnt
1000	Andrew	600
2000	Raymond	400
3000	Mathew	200
200	Johnson	
400	George	
600	Tom	

Tabla Resultante

Nombre	nombre_gerente
Andrew	Tom
Raymond	George
Mathew	Johnson

Pasos para extender una Consulta Simple a un `JOIN`

- Agregar la nueva tabla a la cláusula **`FROM`**
- Agregar otra condición a la cláusula **`WHERE`**
- Probar la consulta
- Agregar una nueva columna a la lista **`SELECT`**

La Cláusula GROUP BY

- La salida de una sentencia SELECT puede requerir un agrupamiento lógico de los datos existentes, basados sobre el valor de un atributo

```
SELECT id_comprador,
COUNT(producto)AS Productos
FROM juguetes
GROUP BY id_comprador;
```

- Más de una columna puede ser incluida en la cláusula **GROUP BY**

La Cláusula HAVING

- Introduce una condición después de que el agrupamiento lógico es realizado
- Para encontrar compradores que han comprado más de dos productos introduzca una condición declarando que **COUNT(item)>2**

```
SELECT idcomprador ,
COUNT(producto) AS Productos
FROM juguetes
GROUP BY idcomprador
HAVING COUNT(producto) > 2;
```

Escribir Subconsultas No correlacionadas

- Una subconsulta (subquery) es una consulta dentro de una consulta
- Una sentencia SELECT anidada dentro de otra sentencia SELECT

```
SELECT producto
FROM juguetes
WHERE idcomprador=(
    SELECT idpropietario
    FROM propietariodejuguetes
    WHERE nombre='Bob');

SELECT producto
FROM juguetes WHERE idcomprador IN (
    SELECT idpropietario
    FROM propietariodejuguetes
    WHERE nombre='Bob');
```

Subconsultas Correlacionadas

```
SELECT depart.nombre
FROM depart
WHERE  EXISTS (
 SELECT id_depart
 FROM emp
 WHERE emp.id_depart = depart.id_depart);
```

- Lista los departamentos que tienen empleados asociados a éstos
- Tiene una consulta padre y una consulta hijo

```
SELECT depart.nombre
FROM depart
WHERE  EXISTS          (Consulta Padre)
   SELECT id_depart
   FROM emp
   WHERE emp.id_depart = depart.id_depart;
                       (Consulta Hijo)
```

Operación de Unión

• Es una operación de conjunto (SET) que combina los resultados de dos o más consultas, elimina los registros duplicados y presenta el resultado.

```
SELECT          noemp,nombreempleado          FROM
departamentoventas
UNION
SELECT          noemp,nombreempleado          FROM
departamentomarketing;
```

Resumen

- Entender las diferentes operaciones con `JOIN`
- Comprender el uso de las cláusulas `GROUP BY` y `HAVING`
- Explicar cómo escribir subconsultas (subqueries)

Unidad 2:
Laboratorio sobre SQL Avanzado

Ejercicios de Laboratorio