{Introducción a SQL}

[Clase 3]

Clase 3.

Contenidos principales:

- Breve repaso de los temas de la clase 2.
- Concepto de agrupación de los datos.
- Utilización de la sentencia **GROUP BY.**
- Restricciones en las agrupaciones utilizando la cláusula **HAVING**.
- Subconsultas.
- Formato condicional usando **CASE WHEN**.

Luego de esta clase serás capaz de:

- Realizar agrupaciones de los datos para resumir la información.
- Utilizar subconsultas como fuente para otras consultas sql.
- Generar nueva información a partir de usar el formato condicional.

Repaso clase 2:

La clase anterior vimos cómo combinar dos o más tablas en una sola, ya sea conectando los registros de las tablas mediante algún campo clave (joins) o bien “apilando” una tabla arriba de la otra (uniones).

Ejemplo join:

```
SELECT a.nombre,a.apellido,b.telefono
FROM  tabla_clientes a [INNER/LEFT/RIGHT/FULL]JOIN tabla_telefonos b ON
(a.id_cliente=b.id);
```

Ejemplo unión:

```
SELECT marca,modelo
FROM autos
UNION [ALL]
SELECT marca_camion, modelo
FROM camiones;
```

Agrupación de la información:

A menudo cuando trabajamos con tablas de datos nos encontramos con la necesidad de “resumir” la información agrupando los datos de acuerdo a algunas de sus variables. En SQL existe la cláusula **GROUP BY** que nos permite agrupar la información especificando las columnas por las cuales agrupar y las operaciones a calcular.

Las operaciones son funciones, entre las que se encuentran: **COUNT, SUM, MAX, MIN, AVG, STDDEV**

Sintaxis:

```
SELECT  <columnas agrupadoras>, <funciones de agrupación.>
FROM     nombre_de_tabla
GROUP BY <columnas a agrupadoras>
[HAVING] <condiciones sobre las funciones de agrupación aplicadas>;
```

Ejemplo:

GOLEADORES		
EQUIPO	JUGADOR	GOLES
A	1	13
A	2	13
A	3	12
A	4	3
B	5	15
B	6	11
B	7	17
C	8	14
C	9	15
C	10	5
C	11	16
C	12	8
D	13	11
D	14	1

```
SELECT equipo,AVG(goles) as PROMEDIO
FROM goleadores
GROUP BY equipo;
```

PROMEDIO DE GOL POR EQUIPOS	
EQUIPO	PROMEDIO
A	10,3
B	14,3
C	11,6
D	6,0

SUMA DE GOLES POR EQUIPOS	
EQUIPO	TOTAL
A	41
B	43
C	58
D	12

```
SELECT equipo,SUM(goles) as TOTAL
FROM goleadores
GROUP BY equipo;
```

HAVING

En español “HAVING” quiere decir “que contengan”; por eso es que sirve para filtrar los resultados de una consulta agrupada.

Ejemplo:

```
SELECT equipo,AVG(goles) as PROMEDIO
FROM goleadores
GROUP BY equipo
HAVING AVG(goles)>10,5;
```

Importante: solo podremos filtrar usando **HAVING** si empleamos alguna de las funciones de agrupación específicas.

Subconsultas.

En SQL es posible anidar las consultas de manera de utilizar el resultado de una de ellas como fuente de datos de otra consulta. A este tipo de sentencias se las conoce como **SUBCONSULTAS** (**SUBQUERYS** in english).

La principal ventaja es que no necesitamos crear una tabla con resultados precalculados; si no que podemos utilizar una subquery que los contenga.

La desventaja que tienen es que son costosas, es decir, aumentan bastante el trabajo que tiene que ejecutar el motor de base de datos.

Queremos saber quiénes fueron los goleadores del torneo.

GOLEADORES		
EQUIPO	JUGADOR	GOLES
A	1	15
A	2	3
A	3	19
A	4	8
B	5	8
B	6	3
B	7	17
C	8	8
C	9	4
C	10	16
C	11	13
C	12	3
D	13	19
D	14	8

GOLEADORES	
EQUIPO	JUGADOR
A	3
D	13

```
SELECT equipo, jugador
FROM goleadores a
     JOIN (SELECT MAX(goles) max_goles
           FROM goleadores) b
     ON a.goles=b.max_goles;
```

Otra forma:

```
SELECT equipo, jugador
FROM goleadores a,
     (SELECT MAX(goles) max_goles
      FROM goleadores)b
WHERE a.goles=b.max_goles;
```

Formato condicional

Cuando queremos que una columna tome ciertos valores dependiendo del valor original de la misma, debemos usar el formato condicional (si X entonces Y). En SQL necesitamos usar la sentencia **CASE WHEN** que es equivalente a la estructura if-then-else de otros lenguajes de programación o la función “SI” de Excel.

Sintaxis:

```
CASE WHEN <condición  V ó F>  THEN <valor>
          . . .
          WHEN <condición  V ó F>  THEN <valor>
          [ELSE] <valor>
     END
```

SALARIOS_EMPLEADOS	
ID_EMPLEADO	SALARIO
1	15000
2	18500
3	12000
4	41000
5	15000
6	18500
7	37000
8	18500
9	12000
10	15000

SALARIOS_EMPLEADOS		
ID_EMPLEADO	SALARIO	CLASIFICACION
1	15000	MEDIO
2	18500	MEDIO
3	12000	BAJO
4	41000	ALTO
5	15000	MEDIO
6	18500	MEDIO
7	37000	ALTO
8	18500	MEDIO
9	12000	BAJO
10	15000	MEDIO

```
SELECT id_empleado, salario,
            CASE WHEN salario<14000 THEN "BAJO"
                 WHEN salario BETWEEN 14000 AND 25000 THEN "MEDIO"
            ELSE "ALTO"
        END AS clasificacion
FROM salarios_empleados;
```

Podría usar un **CASE WHEN** para realizar una agrupación:

SALARIOS_EMPLEADOS	
ID_EMPLEADO	SALARIO
1	15000
2	18500
3	12000
4	41000
5	15000
6	18500
7	37000
8	18500
9	12000
10	15000

CLASIFICACION	CANTIDAD
MEDIO	6
ALTO	2
BAJO	2

```
SELECT
CASE WHEN salario<14000 THEN "BAJO"
     WHEN salario BETWEEN 14000 AND 25000
          THEN "MEDIO"
     ELSE "ALTO"
END AS clasificacion,
    COUNT(*) AS cantidad
FROM salarios_empleados
ORDER BY 2 DESC;
```

Realizar la ejercitación que se encuentra en el archivo: