Funciones

¿Qué es SQL?

- **Structured Query Language** (Lenguaje de Consulta Estructurado)

- **DDL** (Lenguaje de Definición de Datos): Es el lenguaje encargado de la definición, alteración y eliminación de objetos en la base de datos. (CREATE, ALTER, DROP)

- **DML** (Lenguaje de Manipulación de Datos): Es el lenguaje encargado de la manipulación de los datos.(SELECT, INSERT, UPDATE, DELETE)

- **DCL** (Lenguaje de Control de Datos): Es el lenguaje que controla el acceso a las bases de datos. (GRANT, REVOKE)

- **TCL** (Lenguaje de Control de Transacciones): Es el lenguaje que controla la ejecución de comandos DML; indispensable para la consistencia e integridad de los datos.

Funciones

Una función es una rutina creada para tomar parámetros, procesarlos y regresar un resultado.

- Pueden o no recibir parámetros.
- Siempre regresan un valor con algún tipo de dato definido.
- Existen funciones predefinidas en MySQL.
  - (funciones de tiempo, texto, agregación)
- Se pueden crear nuevas funciones en MySQL.
- Se ejecutan con la instrucción:
  - SELECT nombre_funcion([params]);

Funciones de agregación

count, avg, min, max, sum

Funciones de agregación

Las Funciones de Agregación realizan un cálculo sobre un conjunto de datos y regresan un sólo valor.

AVG() - Regresa el valor promedio

COUNT() - Regresa el número total de filas no nulas

MAX() - Regresa el valor mayor

MIN() - Regresa el valor menor

SUM() - Regresa la suma de los valores indicados

COUNT

Sintaxis

select **count(**nombre_columna**)** from nombre_tabla;

Ejemplos:

1. Contar todos los clientes, contar todos los discos, todos los municipios;

select **count(*)** from cliente;

La palabra '**as**' se utiliza para generar un alias del nombre de la columna, le asigna una etiqueta temporal en la vista del reporte

Select **count**(*) **as** total_disqueras from disquera;

COUNT

Se usan comillas si se desea asignar una etiqueta que contiene espacios

```
select count(*) as "Total de municipios" from municipio;
```

2. Contar todos los discos que fueron lanzados en 2010 y 2018 usando un alias en la etiqueta de la columna

```
select count(*) as "Discos 2010 y 2018" from disco
where year(fecha_lanzamiento) in (2010,2018);
```

SUM

Sintaxis

select **sum(**nombre_columna**)** from nombre_tabla;

Ejemplos:

1. Sumar la cantidad total de discos existentes:

select **sum(cantidad_disponible) as total_discos** from disco;

2. Se quiere saber la cantidad total de discos existentes que sean de la disquera 'EMI'

SUM

2. Se quiere saber la cantidad total de discos existentes que sean de la disquera 'EMI'

Solución:

1) Obtener el id de la disquera EMI

```
select * from disquera;
```

2) Hacer la consulta con el id obtenido

```
select sum(cantidad_disponible) as total_discos_EMI
from disco
where  id_disquera = 14;
```

AVG

Sintaxis

select **avg(**nombre_columna**)** from nombre_tabla;

Ejemplos:

1. Obtener el promedio del precio de los discos

select **avg(precio) as promedio** from disco;

2. Obtener el promedio del precio de los discos que fueron lanzados en 2010 y 2018

select avg(precio) from disco where year(fecha_lanzamiento) in (2010,2018);

MAX

Sintaxis

select **max(**nombre_columna**)** from nombre_tabla;

Ejemplos:

1. Obtener el mayor precio de los discos

select **max(precio) as disco_masCaro** from disco;

2. Obtener cuál es la cantidad disponible más alta

select **max(cantidad_disponible)** from disco;

3. Obtener el lanzamiento más reciente

select **max(fecha_lanzamiento)** from disco;

MIN

Sintaxis

select **min(**nombre_columna**)** from nombre_tabla;

Ejemplos:

1. Obtener el menor precio de los discos

select **min(precio) as disco_masBara** from disco;

2. Obtener cuál es la cantidad disponible menor

select **min(cantidad_disponible)** from disco;

3. Obtener el lanzamiento más antiguo

select **min(fecha_lanzamiento)** from disco;

GROUP BY

GROUP BY: se utiliza con las funciones de agregación para agrupar el conjunto de resultados de una o más columnas.

Sintaxis:

```
SELECT nombre_columnaX, funcion_de_agregacion(nombre_columnaY) FROM nombre_tabla [WHERE condicion] GROUP BY nombre_columnaX;
```

Ejemplo:

1. Mostrar un reporte con la cantidad_disponible de discos que hay de cada disquera

GROUP BY

Ejemplo:

1. Mostrar un reporte con la cantidad disponible de discos que hay de cada disquera.

Primero: una consulta sobre las cantidades y disqueras

```
Select id_disquera,cantidad_disponible from disco;
```

Segundo: una consulta agrupada

```
select id_disquera,sum(cantidad_disponible) as
total_discosXdisquera
from disco group by id_disquera;
```

Se puede ordenar y agrupar por el número de columna

```
select id_disquera,sum(cantidad_disponible)
from disco group by 1 order by 2 desc;
```

GROUP BY

Ejemplo: Agrupar y condición WHERE

2. Mostrar un reporte con la cantidad de discos por fecha de lanzamiento, sólo de los años 2003 y 2018

Primero: una consulta auxiliar (opcional) sobre las cantidades y fechas de lanzamiento para ver la información

```
select id_disco,titulo,cantidad_disponible,fecha_lanzamiento from disco where year(fecha_lanzamiento) in (2003,2018) order by 4;
```

Segundo: la consulta agrupada que corresponde a lo solicitado

```
select fecha_lanzamiento,sum(cantidad_disponible) from disco where year(fecha_lanzamiento) in (2003,2018) group by 1;
```

GROUP_CONCAT

Se utiliza con las funciones de agregación para regresar una cadena concatenada resultado de agrupar el conjunto de datos de alguna columna.

Ejemplo:

1. Mostrar un reporte con el número de discos lanzados por año

```
select year(fecha_lanzamiento) as Año,count(titulo) from disco group by 1 order by 1;
```

2. Mostrar un reporte con los titulos de discos lanzados por año

```
select year(fecha_lanzamiento) as Año,group_concat(titulo) from disco group by 1;
```

GROUP_CONCAT

Por default, el separador es una coma, se puede indicar un separador específico

3. Mostrar un reporte con los titulos de discos lanzados por año, separados por ;

```
select year(fecha_lanzamiento) as Año,
group_concat(titulo separator " ; ") from disco group by 1;
```

4. Mostrar un reporte con los titulos de discos lanzados por año, separados por /

```
select year(fecha_lanzamiento) as Año,
group_concat(titulo separator " / ") from disco group by 1;
```

JSON_ARRAYAGG

Regresa un conjunto de datos como un arreglo en formato JSON

Ejemplo: lista de discos lanzados por año como un arreglo json

```
select year(fecha_lanzamiento),json_arrayagg(titulo)
from disco group by 1;
```

JSON_OBJECTAGG

Regresa un conjunto de datos como un arreglo en formato JSON

Ejemplos:

1. Lista de discos vendidos por ticket regresados como arreglo json

```
select id_ticket,json_arrayagg(id_disco) from detalle_ticket group by 1;
```

2. Lista de discos con cantidad vendidos por ticket regresados como objeto json

```
select id_ticket,json_objectagg(id_disco,cantidad) from detalle_ticket group by 1;
```

HAVING

HAVING: Se utiliza para establecer una condición relacionada con la agrupación y donde la cláusula where no aplica.

Sintaxis:

SELECT nombre_columnaX, funcion_de_agregacion(nombre_columnaY) FROM nombre_tabla GROUP BY nombre_columnaX HAVING funcion_de_agregacion(nombre_columnaY) operador valor;

Ejemplo:

¿Cómo podría obtenerse un reporte para saber de qué disqueras hay más de 1000 discos?

HAVING

Ejemplo:

Listar las disqueras donde hay más de 1000 discos

Ideas....

Primero, una consulta para obtener el listado de disqueras con la cantidad disponible

```
select id_disquera,cantidad_disponible from disco order by id_disquera;
```

Segundo, sumar las cantidades y agrupar por disquera

```
select id_disquera,sum(cantidad_disponible) from disco group by id_disquera;
```

HAVING

Ejemplo:

¿De qué disqueras hay más de 1000 discos?

Tercero, se puede usar la clausula WHERE?

```
select id_disquera,sum(cantidad_disponible) from disco where
cantidad_disponible > 1000 group by id_disquera;
```

Cuál es el resultado?

Lo correcto es usar la clausula HAVING

```
select id_disquera,sum(cantidad_disponible)
from disco
group by id_disquera
having sum(cantidad_disponible)>1000;
```

¿Preguntas?