

Bases de datos



Menú del día

- Consultas avanzadas **SQL**:
 - Funciones de agregación
 - Agrupación de datos
 - Cruce de tablas

Contabilizar la cantidad de registros de una tabla

Se utiliza la operación **COUNT** aplicado a la columna que quiero contar.

```
SELECT COUNT(NombreDeColumna)
FROM NombreDeLaTabla
WHERE condición
```

```
SELECT COUNT(*)
FROM NombreDeLaTabla
WHERE condicion
```

Las instrucciones detalladas en azul son optativas.

Contabilizar la cantidad de registros de una tabla

```
SELECT COUNT(DNI)
FROM Clientes
WHERE Apellido="Lopez"
```

```
SELECT COUNT(*)
FROM Clientes
```

Clientes
DNI (PK)
Nombre
Apellido

Operaciones numéricas

Suma de registros:

```
SELECT SUM(NombreDeColumna)
FROM NombreDeLaTabla
WHERE condición
```

Promedio de registros:

```
SELECT AVG(NombreDeColumna)
FROM NombreDeLaTabla
WHERE condición
```

Registro con valor máximo:

```
SELECT MAX(NombreDeColumna)
FROM NombreDeLaTabla
WHERE condición
```

Registro con valor mínimo:

```
SELECT MIN(NombreDeColumna)
FROM NombreDeLaTabla
WHERE condición
```

Operaciones numéricas

Ejemplos

```
SELECT SUM(Precio)
FROM Productos
```

```
SELECT AVG(Precio)
FROM Productos
WHERE Precio > 10
```

```
SELECT MAX(Precio)
FROM Productos
WHERE Nombre LIKE "A%"
```

Productos
SKU (PK) (FK)
Nombre
Precio

Ejercitación 1 - Importar la Base de Datos

- Descargar del campus la base de datos SQL Avanzado - Ejercitacion1.sqlite
- Abrir SQLite Manager en Firefox.
- Importar la base de datos.

Ejercitación 1 - Resolver las siguientes consultas

1. Contabilizar la cantidad de artistas [Tabla Artista] registrados en la base de datos.
2. Contabilizar la cantidad de facturas [Tabla Factura] asociadas al código postal 00530.
3. Contabilizar la cantidad de facturas [Tabla Factura] asociadas a la ciudad de Berlín. GO LIMA
4. Seleccionar la factura [Tabla Factura] de monto total más alto vendido en Alemania.
5. Seleccionar al primer empleado de la empresa.
6. Seleccionar los Ingresos de la empresa en el año 2010.

Agrupación de registros

GROUP BY

El **GROUP BY** separa la tabla en diferentes subgrupos y a cada subgrupo le aplica la función de agregación que indicamos en el **SELECT**. Las columnas indicadas en el **GROUP BY** se denominan columnas de agrupación.

```
SELECT Funcion(ColumnaX), ColumnaY  
FROM NombreDeLaTabla  
GROUP BY ColumnaY
```

Agrupación de registros

Ejemplo:

```
SELECT DNI, SUM(Precio)
FROM Compras
GROUP BY DNI
```

Compras
SKU (PK) (FK)
DNI (FK)
Fecha
Precio

¿Cómo funciona el GROUP BY?

Como estamos agrupando por DNI, se crean grupos diferentes por cada DNI que exista en la tabla. En este ejemplo, existen dos grupos.

SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40

Agrupación

SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40

SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
3	33.241.677	03/01/2017	70

SKU	DNI	Fecha	Precio
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	40

Aplicación de la función de agregación

Sobre cada **grupo**, se aplica la función de agregación que se indicó en el SELECT.
En este caso, se aplica la función suma sobre la columna precio

SUM			
SKU	DNI	Fecha	Precio
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	40

SUM			
SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
3	33.241.677	03/01/2017	70

Resultado

El resultado de la consulta, es una tabla que contiene el resultado de cada grupo.

DNI	SUM(PRECIO)
35.186.928	100
33.241.677	120

Agrupación de registros con condiciones

HAVING permite eliminar los grupos que no cumplan con la condición indicada. La condición será una función de agregación, ya que se aplica al grupo entero. Es similar al **WHERE**, pero se aplica al grupo entero y no a cada fila.

```
SELECT Funcion(ColumnaX), ColumnaY  
FROM NombreDeLaTabla  
GROUP BY ColumnaY  
HAVING condicion
```

Agrupación de registros

Ejemplo:

```
SELECT DNI, SUM(Precio)
FROM Compras
GROUP BY DNI
HAVING SUM(Precio) > 110
```

Compras
SKU (PK) (FK)
DNI (FK)
Fecha
Precio

¿Cómo funciona el **GROUP BY** con **HAVING**?

Como estamos agrupando por DNI, se crean grupos diferentes por cada DNI que exista en la tabla. En este ejemplo, existen dos grupos.

SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40

Agrupación

SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40



SKU	DNI	Fecha	Precio
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	40



SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
3	33.241.677	03/01/2017	70

Aplicación de la función de Agregación

Sobre cada **grupo**, se aplica la función de agregación que se indicó en el SELECT. En este caso, se aplica la función suma sobre la columna precio.

SUM			
SKU	DNI	Fecha	Precio
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	40

SUM			
SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
3	33.241.677	03/01/2017	70

Aplicación de la condición del HAVING

Se eliminan los grupos que no cumplan la condición indicada. En este ejemplo, se eliminan todos los grupos cuya suma es menor a 110.

DNI	SUM(PRECIO)
35.186.928	100
33.241.677	120

Resultado

El resultado de la consulta, es una tabla que contiene el resultado de cada grupo que cumple la condición.

DNI	SUM(PRECIO)
33.241.677	120

Agrupación de registros

Ejemplo Avanzado:

```
SELECT DNI, SUM(Precio)
FROM Compras
WHERE Fecha > "02/01/2017"
GROUP BY DNI
HAVING SUM(Precio) >= 70
```

Compras
SKU (PK) (FK)
DNI (FK)
Fecha
Precio

¿Cómo funciona el **GROUP BY** con **HAVING** y con **WHERE**?

Lo primero que se ejecuta en esta consulta es la cláusula **WHERE**. Se eliminan todas las filas que no cumplan la condición.

SKU	DNI	Fecha	Precio
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40

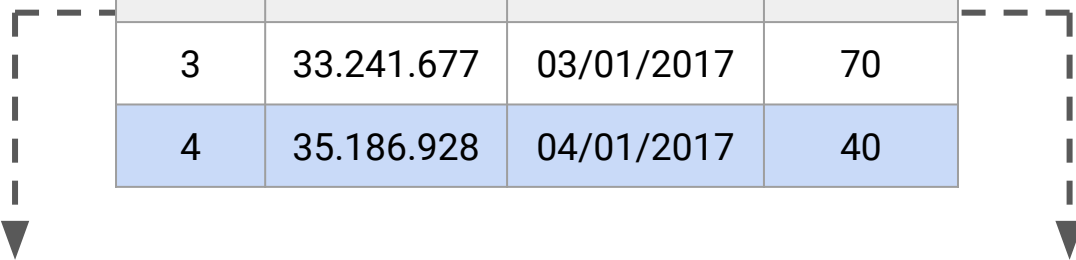
Agrupación

Luego, se continúa con la agrupación. Como estamos agrupando por DNI, se crean grupos diferentes por cada DNI que exista en la tabla. En este ejemplo, existen dos grupos.

SKU	DNI	Fecha	Precio
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40

SKU	DNI	Fecha	Precio
4	35.186.928	04/01/2017	40

SKU	DNI	Fecha	Precio
3	33.241.677	03/01/2017	70



Aplicación de la función de Agregación

Sobre cada **grupo**, se aplica la función de agregación que se indicó en el SELECT.
En este caso, se aplica la función suma sobre la columna precio

SUM			
SKU	DNI	Fecha	Precio
4	35.186.928	04/01/2017	40

SUM			
SKU	DNI	Fecha	Precio
3	33.241.677	03/01/2017	70

Aplicación de la condición del HAVING

Se eliminan los grupos que no cumplan la condición indicada. En este ejemplo, se eliminan todos los grupos cuya suma es menor a 70.

DNI	SUM(PRECIO)
35.186.928	40
33.241.677	70

Resultado

El resultado de la consulta, es una tabla que contiene el resultado de cada grupo que cumple la condición.

DNI	SUM(PRECIO)
33.241.677	70

Junta de tablas

JOIN

JOIN permite combinar registros de diferentes tablas. En el **WHERE** debemos establecer la condición por la cual queremos unir la tabla, que generalmente es algo que tengan ambas en común.

```
SELECT Columna1, Columna2, ...  
FROM NombreDeLaTabla1, NombreDeLaTabla2, ...  
WHERE condicionDeJunta
```

Junta de tablas

Ejemplo

```
SELECT C.SKU, C.DNI, Cli.Apellido  
FROM Compras AS C, Clientes AS Cli  
WHERE C.DNI == Cli.DNI
```

Compras
SKU (PK) (FK)
DNI (FK)
Fecha

Clientes
DNI (PK)
Nombre
Apellido

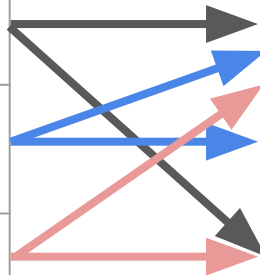
¿Cómo funciona el JOIN?

Al seleccionar dos tablas en el FROM, SQL lo que hace es **devolver una nueva tabla con todos los registros combinados de una con todas las posibles combinaciones de la otra** (producto cartesiano)

¿Cómo funciona el JOIN?

```
SELECT C.SKU, C.DNI, Cli.Apellido  
FROM Compras AS C, Clientes AS Cli
```

SKU	DNI	Fecha	Precio
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40
5	33.241.677	03/01/2017	50



DNI	Nombre	Apellido
33.241.677	Cesar	Carroza
35.186.928	Mariano	Firefox

¿Cómo funciona el JOIN?

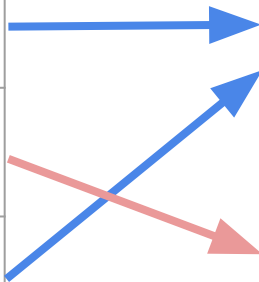
Para eliminar las combinaciones que no queremos, ponemos la **condición por la cual queremos que quede la unión.**

Esta restricción va en el **where**.

¿Cómo funciona el JOIN?

```
SELECT C.SKU, C.DNI, Cli.Apellido
FROM Compras AS C, Clientes AS Cli
WHERE C.DNI == Cli.DNI
```

SKU	DNI	Fecha	Precio
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40
5	33.241.677	03/01/2017	50



DNI	Nombre	Apellido
33.241.677	Cesar	Carroza
35.186.928	Mariano	Firefox

Alias de tablas y columnas

Se puede utilizar un alias para abreviar el nombre de una tabla o una columna, utilizando la instrucción **AS**

```
SELECT T1.Columna1, T2.Columna2 as C2  
FROM NombreDeLaTabla1 AS T1, NombreDeLaTabla2 AS T2
```

Alias de tablas y columnas

Ejemplos:

```
SELECT C.SKU, C.DNI, Cli.Apellido  
FROM Compras AS C, Clientes AS Cli  
WHERE C.DNI == Cli.DNI
```

```
SELECT DNI, SUM(Precio) AS PrecioTotal  
FROM Compras  
WHERE Fecha > "02/01/2017"  
GROUP BY DNI  
HAVING PrecioTotal >= 70
```

Ejercitación 1 - Ejercicio 2 - Resolver las siguientes consultas

1. Listar los nombres de bandas y la cantidad de álbumes publicados.
2. Listar todos los clientes y el total de dinero que han gastado.
3. Listar los artistas del género Metal.
4. Listar los nombres de clientes que compraron en el 2009.
5. Listar las Facturas en las que se compraron más de 5 canciones.
6. Listar los nombres de las canciones ordenadas “de más compradas a menos compradas”.