

DAW/DAM. UD 6. MODELO FÍSICO DQL. ACTIVIDADES NO EVALUABLES. BOLETÍN B (SOLUCIONADO)

DAW/DAM. Bases de datos (BD)

UD 6. MODELO FÍSICO DQL

Boletín B. Prácticas no evaluables (solucionado)

Abelardo Martínez y Pau Miñana

Basado y modificado de Sergio Badal (www.sergiobadal.com) y Raquel Torres.

Curso 2023-2024

Aspectos a tener en cuenta

Importante

Estas actividades son opcionales y no evaluables pero es recomendable hacerlas para un mejor aprendizaje de la asignatura.

Si buscas las soluciones por Internet o preguntas al oráculo de ChatGPT, te estarás engañando a ti mismo. Ten en cuenta que **ChatGPT no es infalible ni todopoderoso.**

Es una gran herramienta para agilizar el trabajo una vez se domina una materia, pero usarlo como atajo en el momento de adquirir habilidades y conocimientos básicos perjudica gravemente tu aprendizaje. Si lo utilizas para obtener soluciones o asesoramiento respecto a las tuyas, revisa cuidadosamente las soluciones propuestas igualmente. Intenta resolver las actividades utilizando los recursos que hemos visto y la documentación extendida que encontrarás en el “Aula Virtual”.

Recomendaciones

Importante

- **No uses NUNCA tildes, ni eñes, ni espacios, ni caracteres no alfanuméricos** (salvo el guión bajo) **en los metadatos** (nombres de elementos de una base de datos).
- Sé coherente con el uso de mayúsculas/minúsculas.

1. BD Jardinería. Esquema

Disponemos del siguiente esquema de la BD o diseño físico, en el que se muestran las tablas que lo forman y cómo están relacionadas entre sí:



Dedícale unos minutos a revisar los nombres de las tablas, los campos que las forman y cómo están relacionadas entre ellas. Es fundamental conocer las tablas para realizar después las consultas de forma adecuada.

2. Consultas. Nivel medio

Actividad no evaluable

Realiza las siguientes consultas en **MySQL** sobre la base de datos anterior.

2.1. Ejercicio

Mostrar el número de clientes que tenemos en cada ciudad en una columna denominada "Num_de_Clientes", ordenado por el número de clientes de mayor a menor. Recordad que no debemos ordenar nunca por el alias de la columna, sino por la expresión o función agregada.

Solución

```
mysql> SELECT COUNT(*) AS Num_de_Clientes, Ciudad
-> FROM Clientes
-> GROUP BY Ciudad
-> ORDER BY COUNT(*) DESC;
```

Num_de_Clientes	Ciudad
11	Madrid
6	Fuenlabrada
2	Miami
2	Barcelona
2	Humanes
2	Paris
2	Sydney
1	San Francisco
1	New York
1	San Lorenzo del Escorial
1	Montornes del valles
1	Santa cruz de Tenerife
1	Canarias
1	Sotogrande

```
|          1 | Getafe          |
|          1 | London          |
+-----+-----+
16 rows in set (0,00 sec)
```

2.2. Ejercicio

Mostrar el número de clientes que tenemos en cada ciudad de España en una columna denominada "Num_de_Clientes", ordenado por la ciudad.

Solución

```
mysql> SELECT COUNT(*) AS Num_de_Clientes, Ciudad
-> FROM Clientes
-> WHERE País = 'España'
-> GROUP BY Ciudad
-> ORDER BY Ciudad;
```

```
+-----+-----+
| Num_de_Clientes | Ciudad          |
+-----+-----+
|          2 | Barcelona      |
|          1 | Canarias       |
|          4 | Fuenlabrada    |
|          1 | Getafe         |
|          2 | Humanes        |
|         10 | Madrid         |
|          1 | Montornes del valles |
|          1 | Santa cruz de Tenerife |
|          1 | Sotogrande     |
+-----+-----+
9 rows in set (0,01 sec)
```


2.3. Ejercicio

Mostrar el número de clientes que tenemos en cada ciudad de España con más de un cliente en una columna denominada "Num_de_Clientes", ordenado de mayor a menor por el número de clientes.

Solución

Analicemos el enunciado:

- Mostrar el número de clientes → COUNT(*)
- En cada ciudad → agrupado por ciudad
- De España → donde el país es España
- Con más de 1 cliente → número de clientes > 1
- Ordenado de mayor a menor → por número de clientes descendente

Ahora creamos la instrucción SQL que tiene en cuenta todo esto.

```
mysql> SELECT COUNT(*) AS Num_de_Clientes, Ciudad
-> FROM Clientes
-> WHERE País = 'España'
-> GROUP BY Ciudad
-> HAVING COUNT(*) > 1
-> ORDER BY COUNT(*) DESC;
```

Num_de_Clientes	Ciudad
10	Madrid
4	Fuenlabrada
2	Barcelona
2	Humanes

```
+-----+-----+
4 rows in set (0,00 sec)
```

En la cláusula de ordenación ORDER BY, también podemos hacer referencia al número de orden de la columna en el SELECT. De este modo, esta sentencia sería equivalente:

```
SELECT COUNT(*) AS Num_de_Clientes, Ciudad
FROM Clientes
WHERE Pais = 'España'
GROUP BY Ciudad
HAVING COUNT(*) > 1
ORDER BY 1 DESC;
```

2.4. Ejercicio

Mostrar cuál es el beneficio máximo que se puede obtener con la venta de un producto de los que tenemos en *stock* en cada una de las gamas que tenemos. Ordena el resultado por el beneficio de mayor a menor.

Solución

```
mysql> SELECT MAX(PrecioVenta-PrecioProveedor) AS Beneficio, Gama
-> FROM Productos
-> WHERE CantidadEnStock > 0
-> GROUP BY Gama
-> ORDER BY MAX(PrecioVenta-PrecioProveedor) DESC;
```

```
+-----+-----+
| Beneficio | Gama          |
+-----+-----+
|      93.00 | Ornamentales |
|      20.00 | Frutales      |
|       3.00 | Herramientas  |
|       1.00 | Aromáticas    |
+-----+-----+
4 rows in set (0,01 sec)
```

También podemos poner ORDER BY 1 DESC.

2.5. Ejercicio

Obtener cuántos pedidos ha realizado cada cliente, ordenado por el número de pedidos, de mayor a menor número de pedidos.

Solución

```
mysql> SELECT CodigoCliente, COUNT(*) AS Total_Num_Pedidos
-> FROM Pedidos
-> GROUP BY CodigoCliente
-> ORDER BY COUNT(*) DESC;
```

CodigoCliente	Total_Num_Pedidos
1	11
30	10
16	10
3	9
7	5
9	5
13	5
14	5
15	5
5	5
19	5
23	5
26	5
27	5
28	5
4	5

```
|          35 |          5 |  
|          36 |          5 |  
|          38 |          5 |  
+-----+-----+  
19 rows in set (0,00 sec)
```

También podemos poner ORDER BY 2 DESC.

2.6. Ejercicio

Mostrar cuántos pedidos ha rechazado cada uno de nuestros clientes, ordenado por el número total de rechazos.

Solución

```
mysql> SELECT CodigoCliente, COUNT(*) AS Total_Rechazados  
-> FROM Pedidos  
-> WHERE Estado = 'Rechazado'  
-> GROUP BY CodigoCliente  
-> ORDER BY COUNT(*) DESC;
```

CodigoCliente	Total_Rechazados
1	2
3	2
4	2
5	2
13	2
16	2
30	2
38	2
7	1
9	1
14	1
15	1
19	1
23	1
28	1

```
|          36 |          1 |
+-----+
16 rows in set (0,01 sec)
```

También podemos poner ORDER BY 2 DESC.

2.7. Ejercicio

Mostrar el importe total del pedido número 10.

Solución

```
mysql> SELECT SUM(Cantidad*PrecioUnidad) AS Total_Pedido_10
-> FROM DetallePedidos
-> WHERE CódigoPedido = 10;
+-----+
| Total_Pedido_10 |
+-----+
|          2920.00 |
+-----+
1 row in set (0,00 sec)
```

2.8. Ejercicio

Obtener la máxima cantidad de un producto solicitada en un pedido siempre que ésta sea mayor o igual a 100. Mostrar el resultado ordenado por la cantidad pedida.

Solución


```
mysql> SELECT CódigoProducto, MAX(Cantidad) AS Maximo_Cant_Pedida
-> FROM DetallePedidos
-> GROUP BY CódigoProducto
-> HAVING MAX(Cantidad) >= 100
-> ORDER BY MAX(Cantidad);
```

```
+-----+-----+
| CódigoProducto | Maximo_Cant_Pedida |
+-----+-----+
| AR-002        | 110                |
| OR-157        | 113                |
| FR-29         | 120                |
| FR-48         | 120                |
| 30310         | 143                |
| OR-177        | 150                |
| OR-247        | 150                |
| AR-006        | 180                |
| FR-57         | 203                |
| OR-214        | 212                |
| AR-009        | 290                |
| FR-17         | 423                |
| AR-008        | 450                |
+-----+-----+
13 rows in set (0,01 sec)
```

En la cláusula de ordenación ORDER BY, también podemos hacer referencia al número de orden de la columna en el SELECT. De este modo, esta sentencia sería equivalente:

```
SELECT CódigoProducto, MAX(Cantidad) AS Maximo_Cant_Pedida
FROM DetallePedidos
GROUP BY CódigoProducto
HAVING MAX(Cantidad) >= 100
ORDER BY 2;
```


2.9. Ejercicio

Mostrar el código del producto y el importe total pedido de cada producto cuyo importe total esté situado entre los 800 y los 1000 euros, ordenado por el total obtenido.

Solución

```
mysql> SELECT CodigoProducto, SUM(Cantidad*PrecioUnidad) AS Total_Producto
-> FROM DetallePedidos
-> GROUP BY CodigoProducto
-> HAVING SUM(Cantidad*PrecioUnidad) BETWEEN 800 AND 1000
-> ORDER BY SUM(Cantidad*PrecioUnidad);
```

```
+-----+-----+
| CodigoProducto | Total_Producto |
+-----+-----+
| OR-225         | 840.00         |
| FR-17          | 846.00         |
| OR-208         | 884.00         |
| FR-79          | 946.00         |
| OR-218         | 950.00         |
| OR-237         | 950.00         |
| FR-29          | 960.00         |
| OR-217         | 975.00         |
| FR-82          | 980.00         |
| AR-009         | 986.00         |
| 22225         | 996.00         |
+-----+-----+
11 rows in set (0,01 sec)
```

También podemos poner ORDER BY 2.

2.10. Ejercicio

Mostrar el código del producto y el importe total pedido de cada producto, de los productos con un precio mayor o igual a 50 euros y menor o igual a 100 y cuyo importe total esté situado entre los 800 y los 1000 euros, ordenado por el código del producto.

Solución

```
mysql> SELECT CodigoProducto, SUM(Cantidad*PrecioUnidad) AS Total_Producto
-> FROM DetallePedidos
-> WHERE PrecioUnidad BETWEEN 50 AND 100
-> GROUP BY CodigoProducto
-> HAVING SUM(Cantidad*PrecioUnidad) BETWEEN 800 AND 1000
-> ORDER BY CodigoProducto;

+-----+-----+
| CodigoProducto | Total_Producto |
+-----+-----+
| FR-82          |          980.00 |
| OR-217         |          975.00 |
+-----+-----+
2 rows in set (0,00 sec)
```

También podemos poner ORDER BY 1.

2.11. Ejercicio

Mostrar el código del cliente, su nombre y los números de los pedidos que han realizado los clientes del representante cuyo nombre es Emmanuel.

Solución

a) OPCIÓN 1

Forma implícita.

```
mysql> SELECT C.CodigoCliente, C.NombreCliente, P.CodigoPedido
-> FROM Empleados E, Clientes C, Pedidos P
-> WHERE E.CodigoEmpleado = C.CodigoEmpleadoRepVentas
->      AND C.CodigoCliente = P.CodigoCliente
->      AND E.Nombre = 'Emmanuel'
-> ORDER BY C.CodigoCliente;
```

CodigoCliente	NombreCliente	CodigoPedido
7	Beragua	13
7	Beragua	14
7	Beragua	15
7	Beragua	16
7	Beragua	17
9	Naturagua	18
9	Naturagua	19
9	Naturagua	20
9	Naturagua	21
9	Naturagua	22

10 rows in set (0,01 sec)

b) OPCIÓN 2

Forma explícita.

```
SELECT C.CodigoCliente, C.NombreCliente, P.CodigoPedido
FROM Empleados E
INNER JOIN Clientes C ON E.CodigoEmpleado = C.CodigoEmpleadoRepVentas
INNER JOIN Pedidos P ON C.CodigoCliente = P.CodigoCliente
WHERE E.Nombre = 'Emmanuel'
ORDER BY C.CodigoCliente;
```

2.12. Ejercicio

Mostrar el nombre de los empleados y el número de pedidos realizados por todos sus clientes ordenado de menor a mayor por el número de pedidos.

Solución

a) OPCIÓN 1

Forma implícita.

```
mysql> SELECT E.Nombre, COUNT(P.CodigoPedido) AS Total_Pedidos_Clientes
-> FROM Empleados E, Clientes C, Pedidos P
-> WHERE C.CodigoCliente = P.CodigoCliente
->      AND E.CodigoEmpleado = C.CodigoEmpleadoRepVentas
-> GROUP BY E.Nombre
-> ORDER BY COUNT(P.CodigoPedido);
```

Nombre	Total_Pedidos_Clientes
Michael	5
Mariano	10
Lucio	10
Emmanuel	10
José Manuel	10
Lorena	10
Julian	10
Mariko	10
Felipe	20
Walter Santiago	20

10 rows in set (0,00 sec)

b) OPCIÓN 2

Forma explícita.

```
SELECT E.Nombre, COUNT(P.CodigoPedido) AS Total_Pedidos_Clientes
FROM Empleados E
INNER JOIN Clientes C ON E.CodigoEmpleado = C.CodigoEmpleadoRepVentas
INNER JOIN Pedidos P ON C.CodigoCliente = P.CodigoCliente
GROUP BY E.Nombre
ORDER BY 2;
```


2.13. Ejercicio

Mostrar cuál es el beneficio máximo (en una columna denominada Beneficio) que se puede obtener con la venta de un producto de los que tenemos en *stock* (si no tiene *stock* no cuenta). Necesitamos saber también a qué producto pertenece ese beneficio.

Solución

a) PRIMERA OPCIÓN

La primera opción que nos viene a la cabeza es ésta, consistente en pedir el nombre y un valor agregado sin usar GROUP BY, que nos da error:

```
mysql> SELECT Nombre, MAX(PrecioVenta-PrecioProveedor) AS Beneficio
-> FROM Productos
-> WHERE CantidadEnStock > 0;
ERROR 1140 (42000): In aggregated query without GROUP BY, expression #1 of SEL
```

Algunos SGBD permiten deshabilitar esta restricción, aunque no debemos hacerlo, puesto que nos devolverá resultados impredecibles (como explican en [este enlace](#)).

b) SEGUNDA OPCIÓN

Otra opción que se podría pensar es aplicar un GROUP BY, ordenar y usar la cláusula LIMIT 1, que ofrece solo el primer resultado de todos los posibles.

```
mysql> SELECT Nombre, MAX(PrecioVenta-PrecioProveedor) AS Beneficio
-> FROM Productos
-> WHERE CantidadEnStock > 0
-> GROUP BY Nombre
-> ORDER BY MAX(PrecioVenta-PrecioProveedor) DESC LIMIT 1;
```

```
+-----+-----+
| Nombre          | Beneficio |
+-----+-----+
| Trachycarpus Fortunei |    93.00 |
+-----+-----+
1 row in set (0,00 sec)
```

Sin embargo, como hemos comentado en el tema de DQL básico, la opción LIMIT es específica de MySQL y **no ofrece un resultado real en el caso de que hayan varios valores repetidos.**

c) TERCERA OPCIÓN

La opción correcta es aplicar subconsultas, que veremos en el tema siguiente.

3. Bibliografía

- MySQL 8.0 Reference Manual. <https://dev.mysql.com/doc/refman/8.0/en/>
- Oracle Database Documentation. <https://docs.oracle.com/en/database/oracle/oracle-database/index.html>
- MySQL Tutorial. <https://www.w3schools.com/mysql/>
- GURU99. Tutorial de MySQL para principiantes Aprende en 7 días. <https://guru99.es/sql/>
- SQL Tutorial - Learn SQL. <https://www.sqltutorial.net/>



Obra publicada con [Licencia Creative Commons Reconocimiento Compartir igual 4.0](https://creativecommons.org/licenses/by-sa/4.0/)