UNIT 5

PHYSICAL MODELING
DATA QUERY LANGUAGE (DQL)

BASES DE DATOS 2022/2023
CFGS DAW

FULL DQL WORKSHOP (SELECTS)
SOLUTIONS WORKSHOP B (MEDIUM LEVEL)

Reviewed by:
Sergio Badal
Author:
Paco Aldarias
Date: 13. March 2023

SOLUTION QUERY 21

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 y alfabéticamente en caso de empate.

```
SELECT COUNT(*) AS Num_de_clientes, ciudad

FROM clientes

GROUP BY ciudad

ORDER BY 1 DESC, ciudad;
```

SERÍAN REALMENTE 16 RESULTADOS SI HAS

CAMBIADO "Fenlabrada" por "Fuenlabrada"

```
+-----------------+--------------------------+
| Num_de_clientes | ciudad                   |
+-----------------+--------------------------+
|              11 | Madrid                   |
|               5 | Fuenlabrada              |
|               2 | Barcelona                |
|               2 | Humanes                  |
|               2 | Miami                    |
|               2 | Paris                    |
|               2 | Sydney                   |
|               1 | Canarias                 |
|               1 | Fenlabrada               |
|               1 | Getafe                   |
|               1 | London                   |
|               1 | Montornes del valles     |
|               1 | New York                 |
|               1 | San Francisco            |
|               1 | San Lorenzo del Escorial |
|               1 | Santa cruz de Tenerife   |
|               1 | Sotogrande               |
+-----------------+--------------------------+
17 rows in set (0.00 sec)
```

SOLUTION QUERY 22

Mostrar los datos de la consulta anterior excluyendo las ciudades que no empiezan por M o por S.

```
SELECT COUNT(*) AS Num_de_clientes, ciudad

FROM clientes

WHERE ciudad LIKE 'M%' OR ciudad LIKE 'S%'

GROUP BY ciudad

ORDER BY 1 DESC, ciudad;
```

```
mysql> SELECT COUNT(*) AS Num_de_clientes, ciudad
    -> FROM clientes
    -> WHERE ciudad LIKE 'M%' OR ciudad LIKE 'S%'
    -> GROUP BY ciudad
    -> ORDER BY 1 DESC, ciudad;
+-----------------+--------------------------+
| Num_de_clientes | ciudad                   |
+-----------------+--------------------------+
|              11 | Madrid                   |
|               2 | Miami                    |
|               2 | Sydney                   |
|               1 | Montornes del valles     |
|               1 | San Francisco            |
|               1 | San Lorenzo del Escorial |
|               1 | Santa cruz de Tenerife   |
|               1 | Sotogrande               |
+-----------------+--------------------------+
8 rows in set (0.00 sec)
```

SOLUTION QUERY 23

Mostrar los datos de la primera consulta, seleccionado solo las ciudades de España con más de un cliente. Ignora las ciudades con país "Spain" (es un error al FILTRAR LA ENTRADA).

```
SELECT COUNT(*) AS Num_de_clientes, ciudad

FROM clientes

WHERE pais = 'España'

GROUP BY ciudad

HAVING COUNT(*) > 1

ORDER BY 1 DESC, ciudad;
```

```
mysql> SELECT COUNT(*) AS Num_de_clientes, ciudad
    -> FROM clientes
    -> WHERE pais = 'España'
    -> GROUP BY ciudad
    -> HAVING COUNT(*) > 1
    -> ORDER BY 1 DESC, ciudad;
+-----------------+-------------+
| Num_de_clientes | ciudad      |
+-----------------+-------------+
|              10 | Madrid      |
|               3 | Fuenlabrada |
|               2 | Barcelona   |
|               2 | Humanes     |
+-----------------+-------------+
4 rows in set (0.00 sec)
```

SOLUTION QUERY 24

Mostrar cuál es el beneficio máximo (precioventa-precioproveedor) 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.

```
SELECT MAX(precioventa - precioproveedor) AS Beneficio, gama

FROM productos

WHERE cantidadenstock >0

GROUP BY gama

HAVING COUNT(*) > 1

ORDER BY 1 DESC;
```

```
mysql> SELECT MAX(precioventa - precioproveedor) AS Beneficio, gama
    -> FROM productos
    -> WHERE cantidadenstock >0
    -> GROUP BY gama
    -> HAVING COUNT(*) > 1
    -> ORDER BY 1 DESC;
+-----------+--------------+
| Beneficio | gama         |
+-----------+--------------+
|     93.00 | Ornamentales |
|     20.00 | Frutales     |
|      3.00 | Herramientas |
|      1.00 | Aromáticas   |
+-----------+--------------+
4 rows in set (0.00 sec)
```

SOLUTION QUERY 25

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

```
SELECT codigocliente, COUNT(*) AS cuantos_pedidos

FROM pedidos

GROUP BY codigocliente

ORDER BY 2 DESC;
```

```
mysql> SELECT codigocliente, COUNT(*) AS cuantos_pedidos
    -> FROM pedidos
    -> GROUP BY codigocliente
    -> ORDER BY 2 DESC;
+---------------+-----------------+
| codigocliente | cuantos_pedidos |
+---------------+-----------------+
|             1 |              11 |
|            16 |              10 |
|            30 |              10 |
|             3 |               9 |
|             4 |               5 |
|             5 |               5 |
|             7 |               5 |
|             9 |               5 |
|            13 |               5 |
|            14 |               5 |
|            15 |               5 |
|            19 |               5 |
|            23 |               5 |
|            26 |               5 |
|            27 |               5 |
|            28 |               5 |
|            35 |               5 |
|            36 |               5 |
|            38 |               5 |
+---------------+-----------------+
19 rows in set (0.00 sec)
```

SOLUTION QUERY 26

Mostrar cuántos pedidos ha rechazado cada uno de nuestros clientes junto con el nombre del cliente, ordenado por el número de pedidos rechazados y nombre del cliente.

```
SELECT C.nombrecliente AS cliente, COUNT(*) AS cuantos_rechazados

FROM pedidos P, clientes C

WHERE P.codigocliente = C.codigocliente AND P.estado = 'Rechazado'

GROUP BY P.codigocliente

ORDER BY 2 DESC;
```

```
mysql> SELECT C.nombrecliente AS cliente, COUNT(*) AS cuantos_rechazados
    -> FROM pedidos P, clientes C
    -> WHERE P.codigocliente = C.codigocliente AND P.estado = 'Rechazado'
    -> GROUP BY P.codigocliente
    -> ORDER BY 2 DESC;
+------------------------+--------------------+
| cliente                | cuantos_rechazados |
+------------------------+--------------------+
| Tendo Garden           |                  2 |
| DGPRODUCTIONS GARDEN   |                  2 |
| Gardening Associates   |                  2 |
| Camunas Jardines S.L.  |                  2 |
| Gerudo Valley          |                  2 |
| Flores Marivi          |                  2 |
| Jardineria Sara        |                  2 |
| El Jardin Viviente S.L |                  2 |
| Beragua                |                  1 |
| Naturagua              |                  1 |
| Dardena S.A.           |                  1 |
| Golf S.A.              |                  1 |
| Sotogrande             |                  1 |
| Jardin de Flores       |                  1 |
| Agrojardin             |                  1 |
| FLORES S.L.            |                  1 |
+------------------------+--------------------+
16 rows in set (0.00 sec)
```

SOLUTION QUERY 27

Mostrar el importe total (cantidad x precio unidad) del pedido número 10.

```
SELECT SUM(cantidad*preciounidad) AS total

FROM detallepedidos

WHERE codigopedido = 10;
```

```
mysql> SELECT SUM(cantidad*preciounidad) AS total
    -> FROM detallepedidos
    -> WHERE codigopedido = 10;
+---------+
| total   |
+---------+
| 2920.00 |
+---------+
1 row in set (0.00 sec)
```

SOLUTION QUERY 28

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. ¿Qué cambios harías en la consulta para mostrar, además, el nombre del producto?

```
SELECT codigoproducto, MAX(cantidad) AS cantidad_maxima

FROM detallepedidos

GROUP BY codigoproducto

HAVING MAX(cantidad) >= 100

ORDER BY 2;
```

```
mysql> SELECT codigoproducto, MAX(cantidad) AS cantidad_maxima
    -> FROM detallepedidos
    -> GROUP BY codigoproducto
    -> HAVING MAX(cantidad) >= 100
    -> ORDER BY 2;
+----------------+-----------------+
| codigoproducto | cantidad_maxima |
+----------------+-----------------+
| 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 (0.00 sec)
```

```
SELECT D.codigoproducto, P.nombre AS producto,

MAX(D.cantidad) AS cantidad_maxima

FROM detallepedidos D INNER JOIN productos P

ON D.codigoproducto = P.codigoproducto

GROUP BY D.codigoproducto

HAVING MAX(D.cantidad) >= 100

ORDER BY 3;
```

```
mysql> SELECT D.codigoproducto, P.nombre AS producto,
    -> MAX(D.cantidad) AS cantidad_maxima
    -> FROM detallepedidos D INNER JOIN productos P
    -> ON D.codigoproducto = P.codigoproducto
    -> GROUP BY D.codigoproducto
    -> HAVING MAX(D.cantidad) >= 100
    -> ORDER BY 3;
+----------------+-----------------------------------------+-----------------+
| codigoproducto | producto                                | cantidad_maxima |
+----------------+-----------------------------------------+-----------------+
| AR-002         | Lavándula Dentata                       |             110 |
| OR-157         | Acer Pseudoplatanus                     |             113 |
| FR-48          | Nogal Común                             |             120 |
| FR-29          | Cerezo Napoleón                         |             120 |
| 30310          | Azadón                                  |             143 |
| OR-247         | Trachycarpus Fortunei                   |             150 |
| OR-177         | Robinia Pseudoacacia Casque Rouge       |             150 |
| AR-006         | Petrosilium Hortense (Peregil)          |             180 |
| FR-57          | Kaki Rojo Brillante                     |             203 |
| OR-214         | Brahea Armata                           |             212 |
| AR-009         | Thymus Vulgaris                         |             290 |
| FR-17          | Rosal bajo 1Âª -En maceta-inicio brotación |             423 |
| AR-008         | Thymus Citriodra (Tomillo limón)        |             450 |
+----------------+-----------------------------------------+-----------------+
13 rows in set (0.00 sec)
```

SOLUTION QUERY 29.1

Mostrar el código del producto, nombre y el importe total pedido de cada producto cuyo importe total esté entre los 800 y los 1000 euros ordenado por el total obtenido. No puedes usar JOINS.

```
SELECT D.codigoproducto, P.nombre AS producto,

SUM(D.cantidad*D.preciounidad) AS total_producto

FROM detallepedidos D, productos P

WHERE D.codigoproducto = P.codigoproducto

GROUP BY D.codigoproducto

HAVING SUM(D.cantidad*D.preciounidad) BETWEEN 800 AND 1000

ORDER BY 3;
```

```
mysql> SELECT D.codigoproducto, P.nombre AS producto,
    -> SUM(D.cantidad*D.preciounidad) AS total_producto
    -> FROM detallepedidos D, productos P
    -> WHERE D.codigoproducto = P.codigoproducto
    -> GROUP BY D.codigoproducto
    -> HAVING SUM(D.cantidad*D.preciounidad) BETWEEN 800 AND 1000
    -> ORDER BY 3;
+----------------+---------------------------------------------+----------------+
| codigoproducto | producto                                    | total_producto |
+----------------+---------------------------------------------+----------------+
| OR-225         | Chamaerops Humilis                          |         840.00 |
| FR-17          | Rosal bajo 1Âª -En maceta-inicio brotación |         846.00 |
| OR-208         | Tuja orientalis "Aurea nana"                |         884.00 |
| FR-79          | Higuera                                     |         946.00 |
| OR-218         | Butia Capitata                              |         950.00 |
| OR-237         | Livistonia Australis                        |         950.00 |
| FR-29          | Cerezo Napoleón                             |         960.00 |
| OR-217         | Brahea Edulis                               |         975.00 |
| FR-82          | Higuera                                     |         980.00 |
| AR-009         | Thymus Vulgaris                             |         986.00 |
| 22225          | Rastrillo de Jardín                         |         996.00 |
+----------------+---------------------------------------------+----------------+
11 rows in set (0.00 sec)
```

SOLUTION QUERY 29.2

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.

```
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 1;
```

```
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 1;
+----------------+----------------+
| codigoproducto | total_producto |
+----------------+----------------+
| FR-82          |         980.00 |
| OR-217         |         975.00 |
+----------------+----------------+
2 rows in set (0.00 sec)
```

SOLUTION QUERY 29.3

Mostrar el código del cliente, su nombre y los números de los pedidos que han realizado los clientes del empleado que ejerce de representante cuyo nombre es Emmanuel ***(independientemente de cuál sea el puesto de ese empleado).***

```
-- Asumimos que un empleado puede ejercer de representante aunque no sea ese su puesto

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 1;
```

Si observas con detenimiento la base de datos, verás que cada EMPLEADO puede tener un puesto asociado como "director de oficina", "representante", "director de marketing" ... o no tener puesto asignado (nulo). Por otro lado, cada CLIENTE puede tener asociado un EMPLEADO (FOREIGN KEY => REFERENCES Empleados) que ejerce, para ese cliente, como representante de ventas, pudiendo ser ese empleado un "representante de ventas" o no.

Como el enunciado de la consulta puede dar a confusión, si se diera en la empresa, tendríamos que hablar con el cliente (o con quien nos haya pedido la consulta) y pedirle que nos solucione la ambigüedad (o el error de la base de datos) y, como dijimos al principio del curso, si no obtuviéramos respuesta deberíamos incluir un comentario del tipo "Asumimos que un empleado puede ejercer de representante con un cliente aunque no sea ese su puesto".

```
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 1;
+---------------+---------------+--------------+
| 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)
```

SOLUTION QUERY 29.4

Mostrar el nombre de los empleados y el número de pedidos realizados por todos sus clientes ordenado por el número de pedidos y nombre del empleado. Haz dos consultas, una con JOIN y otra con cartesianos.

```
SELECT E.nombre, COUNT(P.codigopedido) AS cuantos_pedidos

FROM clientes C, pedidos P, empleados E

-- FROM clientes C INNER JOIN pedidos P INNER JOIN empleados E

WHERE E.codigoempleado = C.codigoempleadorepventas

-- ON E.codigoempleado = C.codigoempleadorepventas

AND C.codigocliente = P.codigocliente

GROUP BY E.nombre

ORDER BY 2,1;
```

```
mysql> SELECT E.nombre, COUNT(P.codigopedido) AS cuantos_pedidos
    -> FROM clientes C, pedidos P, empleados E
    -> WHERE E.codigoempleado = C.codigoempleadorepventas
    -> AND C.codigocliente = P.codigocliente
    -> GROUP BY E.nombre
    -> ORDER BY 2,1;
+-----------------+-----------------+
| nombre          | cuantos_pedidos |
+-----------------+-----------------+
| Michael         |               5 |
| Emmanuel        |              10 |
| José Manuel     |              10 |
| Julian          |              10 |
| Lorena          |              10 |
| Lucio           |              10 |
| Mariano         |              10 |
| Mariko          |              10 |
| Felipe          |              20 |
| Walter Santiago |              20 |
+-----------------+-----------------+
10 rows in set (0.00 sec)
```

SOLUTION QUERY 29.5 (CONT. 16 BASIC)

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.

a) PRIMERA OPCIÓN

*La primera opción que nos viene a la cabeza es esta, consistente en pedir el nombre y un valor agregado **sin usar group by**, que nos da este resultado:*

```
mysql> select nombre, max(precioventa-precioproveedor) as Beneficio from productos
    -> where cantidadEnStock > 0;
+----------------------+-----------+
| nombre               | Beneficio |
+----------------------+-----------+
| Sierra de Poda 400MM |     93.00 |
+----------------------+-----------+
1 row in set (0.00 sec)
```

(*) Nota: Si te da error en mysql, ve al el anexo 1 (al final de este documento)

Esta opción nos da un resultado IMPREDECIBLE (como explican en este enlace), ya que nos está devolviendo el máximo y un valor arbitrario para "nombre" ya que la "Sierra de Poda" no es el producto que tiene ese beneficio como podemos ver si hacemos esta otra consulta:

```
mysql> select nombre, precioventa, precioproveedor, (precioventa-precioproveedor) as beneficio from Productos where
 cantidadenstock>0 order by beneficio desc;
+-----------------------------------------------------------+-------------+-----------------+-----------+
| nombre                                                    | precioventa | precioproveedor | beneficio |
+-----------------------------------------------------------+-------------+-----------------+-----------+
| Trachycarpus Fortunei                                     |      462.00 |          369.00 |     93.00 |
| Bismarckia Nobilis                                        |      266.00 |          212.00 |     54.00 |
```

Pocos SGBD nos permiten usar agregados y campos en el select sin requerir un GROUP BY. MySQL es uno de ellos, arrojando resultados impredecibles.

b) SEGUNDA OPCIÓN

```
mysql> select nombre, max(precioventa-precioproveedor) as beneficio from Productos where cantidadenstock>0
group by nombre order by beneficio desc limit 1;
+----------------------+-----------+
| nombre               | beneficio |
+----------------------+-----------+
| Trachycarpus Fortunei |     93.00 |
+----------------------+-----------+
1 row in set (0,00 sec)
```

*La opción correcta sería la siguiente, consistente en aplicar un **GROUP BY** y la cláusula **LIMIT 1**, que ofrece solo el primer resultado de todos los posibles.*

ANNEX 1. AGGREGATE FUNCTIONS AND GROUP BY

El sql estándar podemos usar funciones agregadas (max,avg, ..) sin tener que poner group by pero no todos los SGBD las aceptan.

Con mysql podemos usar funciones agregadas sin el group by pero nos puede devolver resultados impredecibles (como explican en este enlace).

Si aún así quieres ejecutar una consulta de este tipo, sigue estos pasos:

```
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 SELECT list contains nonaggregated column
'jardineria2.productos.Nombre'; this is incompatible with sql_mode=only_full_group_by
mysql>
```

Podemos ver que está a on la variable ONLY_FULL_GROUP_BY de mysql con:

select @@sql_mode;

```
mysql> select @@sql_mode;
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
| @@sql_mode
                      |
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
| ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,
NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set (0.00 sec)
```

Vamos a desactivar ONLY_FULL_GROUP_BY con:

SET sql_mode=(SELECT REPLACE(@@sql_mode, 'ONLY_FULL_GROUP_BY', ''));

```
mysql> SET sql_mode=(SELECT REPLACE(@@sql_mode, 'ONLY_FULL_GROUP_BY', ''));
Query OK, 0 rows affected (0.00 sec)
```

Vemos que ya no aparece 'ONLY_FULL_GROUP_BY

```
mysql> select @@sql_mode;
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
| @@sql_mode
                      |
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
| STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set (0.00 sec)
```

Ahora no da error (AUNQUE EL RESULTADO NO ES CORRECTO):

```
mysql> select nombre, max(precioventa-precioproveedor) as Beneficio from productos
    -> where cantidadEnStock > 0;
+-----------------------+-----------+
| nombre                | Beneficio |
+-----------------------+-----------+
| Sierra de Poda 400MM  |     93.00 |
+-----------------------+-----------+
1 row in set (0.00 sec)
```

Más info en este enlace.