Guía Práctica - Unidad V: Consultas en SQL

Juannie

09/06/2024

CONSULTAS SQL

- Las consultas serán probadas en el SGDB MariaDB versión 15.1 Distrib 10.6.16-MariaDB, for debian-linux-gnu (x86_64).
- La relación *catalog* está encerrada entre tildes inversas o *«backticks»* (‘) para evitar errores, ya que dicha palabra figura como reservada.

1. Dado el siguiente esquema relacional:

- suppliers(sid: integer, sname: string, address: string)
- parts(pid: integer, pname: string, color: string)
- catalog(sid: integer, pid: integer, cost: real)

1. **Encuentre los `snombres` de proveedores que provean alguna parte roja.**

```
SELECT DISTINCT s.sname
FROM `catalog` AS c NATURAL JOIN parts AS p NATURAL JOIN suppliers AS s
WHERE p.color LIKE "%red%";
```

2. **Encuentre los `sids` de proveedores que provean alguna parte roja o alguna parte verde.**

Opción 1:

```
SELECT c.sid
FROM `catalog` AS c NATURAL JOIN parts AS p
WHERE p.color LIKE "%RED%"

UNION

SELECT c.sid
FROM `catalog` AS c NATURAL JOIN parts AS p
WHERE p.color LIKE "%GREEN%";
```

Nota: Cuando se utiliza `UNION` o `INTERSECT` se descarta por defecto los valores repetidos, ergo, no hace falta utilizar la cláusula `DISTINCT`.

Opción 2:

```sql
SELECT DISTINCT c.sid
FROM `catalog` AS c NATURAL JOIN parts AS p
WHERE p.color LIKE "%red%" or p.color LIKE "%green%";
```

En esta última opción sí es necesario utilizar la cláusula `DISTINCT`, ya que si no, por ejemplo, el valor de `sid = 2` aparece tres veces, ya que el proveedor con dicho `sid` provee tres partes rojas.

3. **Encuentre los `sids` de proveedores que provean alguna parte roja o vivan en "2 Groom Lake, Rachel, NV 51902".**

Opción 1:

```sql
SELECT DISTINCT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid
WHERE p.color LIKE "%red%" OR EXISTS (
   SELECT 1
   FROM suppliers AS s
   WHERE s.address LIKE "%2 Groom Lake, Rachel, NV 51902%"
);
```

Opción 2:

```sql
SELECT DISTINCT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid INNER JOIN suppliers
↪  AS s ON c.sid = s.sid
WHERE p.color LIKE "%red%" OR s.address LIKE "%2 Groom Lake, Rachel, NV 51902%";
```

Opción 3:

```sql
SELECT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid
WHERE p.color LIKE "%red%"

UNION

SELECT s.sid
FROM suppliers AS s
WHERE s.address LIKE "%2 Groom Lake, Rachel, NV 51902%"
```

4. **Encuentre los `sids` de proveedores que provean alguna parte roja y alguna parte verde.**

Opción 1:

```sql
SELECT DISTINCT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid
WHERE p.color LIKE "%red%" AND EXISTS (
   SELECT *
   FROM catalog AS c2 INNER JOIN parts AS p2 ON c2.pid = p2. pid
   WHERE c.sid = c2.sid AND p2.color LIKE "%green%"
);
```

Opción 2:

```
SELECT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid
WHERE p.color LIKE "%red%"

INTERSECT

SELECT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid
WHERE p.color LIKE "%green%";
```

Opción 3:

```
SELECT DISTINCT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid
WHERE p.color LIKE "%red%" AND c.sid IN (
   SELECT c2.sid
   FROM `catalog` AS c2 INNER JOIN parts AS p2 ON c2.pid = p2.pid
   WHERE p2.color LIKE "%green%"
);
```

5. **Encuentre los sids de proveedores que provean cada parte.**

   Nota: En esta consulta aplicaremos lo equivalente a la división del Álgebra Relacional.

Opción 1:

```
SELECT DISTINCT c.sid
FROM `catalog` AS c
WHERE NOT EXISTS (
   SELECT 1
   FROM parts AS p
   WHERE NOT EXISTS (
      SELECT 1
      FROM `catalog` AS c2 NATURAL JOIN parts AS p2
      WHERE c.sid = c2.sid AND p.pid = p2.pid
   )
);
```

Opción 2:

```
SELECT DISTINCT c.sid
FROM `catalog` AS c
WHERE NOT EXISTS (
   SELECT p2.pid
   FROM parts AS p2

   EXCEPT

   SELECT c2.pid
   FROM `catalog` AS c2
```

```
    WHERE c.sid = c2.sid
);
```

Opción 3:

```
SELECT c.sid
FROM `catalog` AS c
GROUP BY c.sid
HAVING COUNT(DISTINCT c.pid) = (SELECT COUNT(DISTINCT p.pid) FROM parts AS p);
```

> *Importante: No olvidar la cláusula* `DISTINCT` *dentro del operador de agregación* `COUNT`*, ya que sin ella, en los casos donde aparezcan repetidos los valores, la consulta no funcionará. En este caso en particular no es necesario porque* `pid` *es parte de la clave primaria de la tabla* catalog*, pero lo agregué igualmente.*

6. **Encuentre los sids de proveedores que provean cada parte roja.**

Opción 1:

```
SELECT DISTINCT c.sid
FROM `catalog` AS c
WHERE NOT EXISTS (
    SELECT 1
    FROM parts AS p
    WHERE p.color LIKE "%red%" AND NOT EXISTS (
        SELECT 1
        FROM `catalog` AS c2 NATURAL JOIN parts AS p2
        WHERE c.sid = c2.sid AND p.pid = p2.pid
    )
);
```

Opción 2:

```
SELECT DISTINCT c.sid
FROM `catalog` AS c
WHERE NOT EXISTS (
    SELECT p2.pid
    FROM parts AS p2
    WHERE p2.color LIKE "%RED%"

    EXCEPT

    SELECT c2.pid
    FROM `catalog` AS c2
    WHERE c.sid = c2.sid
);
```

Opción 3:

```
SELECT c.sid
FROM `catalog` AS c NATURAL JOIN parts AS p
```

```
WHERE p.color LIKE "%red%"
GROUP BY c.sid
HAVING COUNT(DISTINCT c.pid) = (SELECT COUNT(DISTINCT p.pid) FROM parts AS p
↪  WHERE p.color LIKE "%red%");
```

7. **Encuentre los sids de proveedores que provean cada parte verde o alguna roja.**

```
SELECT DISTINCT c.sid
FROM `catalog` AS c
WHERE NOT EXISTS (
   SELECT p2.pid
   FROM parts AS p2
   WHERE p2.color LIKE "%GREEN%"

   EXCEPT

   SELECT c2.pid
   FROM `catalog` AS c2
   WHERE c.sid = c2.sid
)

UNION

SELECT DISTINCT c.sid
FROM `catalog` AS c INNER JOIN parts AS p ON c.pid = p.pid
WHERE p.color LIKE "%RED%";
```

8. **Encuentre los sids de proveedores que provean cada parte roja o provean cada parte verde.**

```
SELECT DISTINCT c.sid
FROM `catalog` AS c
WHERE NOT EXISTS (
   SELECT *
   FROM parts AS p
   WHERE p.color LIKE 'red' AND NOT EXISTS (
      SELECT *
      FROM `catalog` AS c2 NATURAL JOIN parts AS p2
      WHERE c.sid = c2.sid AND p.pid = p2.pid
   )
)

UNION

SELECT DISTINCT c.sid
FROM `catalog` AS c
WHERE NOT EXISTS (
   SELECT *
   FROM parts AS p
   WHERE p.color LIKE 'green' AND NOT EXISTS (
      SELECT *
      FROM `catalog` AS c2 NATURAL JOIN parts AS p2
      WHERE c.sid = c2.sid AND p.pid = p2.pid
   )
```

```
);
```

9. **Encuentre los pares de `sids` tal que el proveedor con el primer `sid` tenga la misma pieza pero más costosa que el proveedor del segundo `sid`.**

```
SELECT c1.sid AS sid_más_costoso, c2.sid AS sid_menos_costoso, c1.pid AS
↪  pid_pieza_en_cuestión
FROM `catalog` AS c1, `catalog` AS c2
WHERE c1.sid <> c2.sid AND c1.pid = c2.pid AND c1.cost > c2.cost;
```

10. **Encuentre los `pids` de partes provistas por al menos dos proveedores diferentes.**

Opción 1:

```
SELECT DISTINCT c.pid
FROM `catalog` AS c
WHERE EXISTS (
   SELECT 1
   FROM `catalog` AS c2
   WHERE c.pid = c2.pid AND c.sid <> c2.sid
);
```

Opción 2:

```
SELECT DISTINCT c1.pid
FROM `catalog` AS c1, `catalog` AS c2
WHERE c1.sid != c2.sid AND c1.pid = c2.pid;
```

Los operadores de desigualdad `<>` *y* `!=` *son equivalentes, al menos en MaríaDB.*

11. **Encuentre los `pids` de las partes mas caras provistas por el proveedor llamado "Alien Aircaft Inc.".**

```
SELECT p.pid
FROM suppliers AS s NATURAL JOIN `catalog` AS c NATURAL JOIN parts AS p
WHERE s.sname LIKE "Alien%" AND c.cost = (
   SELECT MAX(c2.cost)
   FROM `catalog` AS c2
);
```

12. **Encuentre los `pids` de partes provistas por cada proveedor a menos que $200 (si algún proveedor no provee la parte o cuesta igual o más de $200, dicha parte no deberá ser listada).**

```
SELECT p.pid
FROM parts AS p
WHERE NOT EXISTS (
   SELECT 1
   FROM `catalog` AS c
   WHERE c.cost < 200 AND NOT EXISTS (
      SELECT 1
      FROM parts AS p2 INNER JOIN `catalog` AS c2 ON p2.pid = c2.pid
      WHERE p.pid = p2.pid AND c.sid = c2.sid
```

```
    )
);
```

2. Dado el siguiente esquema relacional:

- flights(flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time)
- aircraft(aid: integer, aname: string, cruisingrange: integer)
- certified(eid: integer, aid: integer)
- employees(eid: integer, ename: string, salary: integer)

1. **Encuentre los eids de pilotos certificados para algún avión Boeing.**

```sql
SELECT DISTINCT c.eid
FROM certified AS c INNER JOIN aircraft AS a ON c.aid = a.aid
WHERE a.aname LIKE "Boeing%";
```

2. **Encuentre los nombres de pilotos certificados para algún avión Boeing.**

```sql
SELECT DISTINCT e.ename
FROM employees AS e NATURAL JOIN certified AS c NATURAL JOIN aircraft AS a
WHERE a.aname LIKE "Boeing%";
```

3. **Encuentre los aids de todos los aviones que pueden ser usados para vuelos sin escalas desde Los Ángeles hasta Sydney.**

```sql
SELECT a.aid
FROM aircraft AS a, flights AS f
WHERE a.cruisingrange > f.distance AND f.origin LIKE "Los Ángeles" AND
↪  f.destination LIKE "Sydney";
```

4. **Identifique los vuelos que pueden ser pilotados por cada piloto cuyo salario sea mayor a $200.000.**

   Opción 1:

```sql
SELECT DISTINCT f.flno
FROM flights AS f, certified AS c INNER JOIN employees AS e ON c.eid = e.eid
↪  INNER JOIN aircraft AS a ON c.aid = a.aid
WHERE a.cruisingrange > f.distance AND e.salary > 200000;
```

   Opción 2:

```sql
SELECT DISTINCT f.flno
FROM flights AS f CROSS JOIN certified AS c INNER JOIN employees AS e ON c.eid =
↪  e.eid INNER JOIN aircraft AS a ON c.aid = a.aid
WHERE a.cruisingrange > f.distance AND e.salary > 200000;
```

5. **Encuentre los nombres de pilotos que pueden operar aviones con un rango mayor a 3,000 millas pero que no estén certificados para los aviones Boeing.**

```sql
SELECT DISTINCT e.ename
FROM employees AS e NATURAl JOIN certified AS c NATURAL JOIN aircraft AS a
WHERE a.cruisingrange > 3000 AND NOT EXISTS (
   SELECT 1
   FROM certified AS c2 NATURAL JOIN aircraft AS a2
   WHERE c.eid = c2.eid AND a2.aname LIKE "Boeing%"
);
```

6. **Encuentre los eids de empleados que ganan el mayor salario.**

```sql
SELECT e.eid
FROM employees AS e
WHERE e.salary = (
   SELECT MAX(e2.salary)
   FROM employees AS e2
);
```

7. **Encuentre los eids de empleados que ganen el segundo mayor salario.**

```sql
SELECT e.eid
FROM employees AS e
WHERE e.salary = (
SELECT MAX(e.salary)
FROM employees AS e
WHERE e.salary < (
   SELECT MAX(e.salary)
   FROM employees AS e
)
);
```

8. **Encuentre los eids de empleados que están certificados para el mayor número de aviones.**

```sql
-- newt es una tabla temporal que tiene los campos eid y núm_aviones. Por cada
↪  eid (o sea, piloto) indica el número de aviones que puede pilotar.
WITH newt AS (
   SELECT c.eid, COUNT(c.aid) AS núm_aviones
   FROM certified AS c
   GROUP BY c.eid
)
-- Y aquí de newt, hallamos quién puede volar más aviones.
SELECT n.eid
FROM newt AS n
WHERE n.núm_aviones = (
   SELECT MAX(n2.núm_aviones)
   FROM newt AS n2
);
```

9. **Encuentre los eids de los empleados que están certificados para exactamente 3 (tres) aviones.**

```sql
SELECT c.eid
FROM certified AS c
GROUP BY c.eid
HAVING COUNT(c.aid) = 3;
```

10. **Encuentre la cantidad total de dinero pagado en concepto de salario.**

```sql
SELECT SUM(e.salary)
FROM employees AS e;
```

3. Dado el siguiente esquema relacional:

- student(snum: integer, sname: string, major: string, level: string, age: integer)
- class(name: string, meets at: string, room: string, fid: integer)
- enrolled(snum: integer, cname: string)
- faculty(fid: integer, fname: string, deptid: integer)

1. **Encuentre los nombres de todos los estudiantes Juniors (level = JR) que estén enrolados en una clase dictada por Ivana Teach.**

```sql
SELECT DISTINCT s.sname
FROM student AS s
INNER JOIN enrolled AS e ON s.snum = e.snum
INNER JOIN class AS c ON c.name = e.cname
INNER JOIN faculty AS f ON f.fid = c.fid
WHERE s.standing LIKE 'JR' AND f.fname LIKE 'I%Teach';
```

2. **Encuentre los nombres de todas las clases que cumplan estar en el aula R128 o tenga más de 5 estudiantes inscriptos.**

```sql
SELECT c.name
FROM class AS c
WHERE c.room LIKE 'r128'

UNION

SELECT c.name
FROM class AS c INNER JOIN enrolled AS e ON c.name = e.cname
GROUP BY c.name
HAVING COUNT(e.snum) > 5;
```

Consulta extra:

Encuentre los nombres de todas las clases que cumplan estar en el aula R128 y tenga más de 5 estudiantes inscriptos.

```sql
SELECT c.name
FROM class AS c INNER JOIN enrolled AS e ON c.name = e.cname
WHERE c.room LIKE 'R128'
GROUP BY c.name
HAVING COUNT(e.snum) > 5;
```

3. **Encuentre los nombres de todos estudiantes que están inscriptos en dos clases que comiencen a la misma hora.**

```sql
SELECT DISTINCT s.sname
FROM student AS s INNER JOIN enrolled AS e ON s.snum = e.snum INNER JOIN class AS
↪  c ON e.cname = c.name
WHERE EXISTS (
```

```
    SELECT *
    FROM enrolled AS e2 INNER JOIN class AS c2 ON e2.cname = c2.name
    WHERE s.snum = e2.snum AND c.name != c2.name AND c.meets_at = c2.meets_at
);
```

4. **Encuentre los nombres de miembros de la facultad que hay en cada aula en la cual se dicta alguna clase.**

   Nota: Este punto se refiere tanto a estudiantes como profesores.

```
(SELECT f.fname AS Miembro, c.room AS Aula
FROM faculty AS f INNER JOIN class AS c ON f.fid = c.fid

UNION

SELECT s.sname AS Miembro, c.room AS Aula
FROM student AS s INNER JOIN enrolled AS e ON s.snum = e.snum INNER JOIN class AS
↪  c ON c.name = e.cname)

-- Este ORDER BY ordena la tabla que resulta de la unión anterior, por eso la
↪  remarqué con paréntesis, aunque éstos no hagan falta.

ORDER BY Aula;
```

5. **Encuentre los nombres de los miembros de la facultad para quienes la inscripción combinada de los cursos que enseñan es menos de cinco.**

```
SELECT f.fname
FROM faculty AS f INNER JOIN class AS c ON f.fid = c.fid
GROUP BY f.fid
HAVING COUNT(f.fid) < 5;
```

6. **Para cada nivel, liste los niveles y los promedios de edad de los estudiantes para ese nivel.**

   Nota: Entiendo que me pide la cantidad de estudiantes que hay por nivel y el promedio de edad en cada uno de dichos niveles.

```
SELECT s.standing, COUNT(s.standing), AVG(s.age) AS prom_edad
FROM student AS s
GROUP BY s.standing;
```

7. **Para todos los niveles excepto JR, liste el nivel y los promedios de edad de estudiantes para dicho nivel.**

```
SELECT s.standing, AVG(s.age)
FROM student AS s
WHERE s.standing NOT LIKE 'JR'
GROUP BY s.standing;
```

8. **Para cada miembro de la facultad que toma clases en el aula R128, liste los nombres de los miembros de la facultad y el total de clases que toma.**

   ¡Atención! La siguiente consulta no funciona, porque la cláusula `WHERE` descarta todas las clases que no sean en el aula 'r128', por lo que no puede contar la totalidad de clases que toma el estudiante si el mismo tiene clases en otras aulas.

```sql
SELECT s.sname, COUNT(e.cname) AS clases_que_toma
FROM student AS s INNER JOIN enrolled AS e ON s.snum = e.snum INNER JOIN class AS
↪  c ON e.cname = c.name
WHERE c.room LIKE 'r128'
GROUP BY s.snum;
```

*Nota: Con la cláusula **EXISTS** nos aseguramos que el estudiante curse al menos una materia en el aula 'r128'*

```sql
SELECT s.sname, COUNT(e.cname) AS total_de_clases
FROM student s INNER JOIN enrolled e ON s.snum = e.snum
GROUP BY s.snum, s.sname
HAVING EXISTS (
   SELECT 1
   FROM enrolled e2 INNER JOIN class c ON e2.cname = c.name
   WHERE e2.snum = s.snum AND c.room = 'R128'
);
```

9. **Encuentre los nombres de estudiantes inscriptos en para el máximo número de clases.**

```sql
-- Esta tabla temporal `newt` tiene el número (algo así como el ID), nombre y
↪  cantidad de clases por estudiante.

WITH newt AS (
   SELECT s.snum, s.sname, COUNT(e.cname) AS nro_de_clases_que_toma
   FROM student s NATURAL JOIN enrolled e
   GROUP BY s.snum
)
SELECT n.sname
FROM newt AS n
WHERE n.nro_de_clases_que_toma IN (
   SELECT MAX(n2.nro_de_clases_que_toma)
   FROM newt AS n2
);
```

10. **Encuentre los nombres de estudiantes no inscriptos en ninguna clase.**

Opción 1:

```sql
SELECT s.sname
FROM student AS s
WHERE NOT EXISTS (
   SELECT 1
   FROM student AS s2 INNER JOIN enrolled AS e ON s2.snum = e.snum
   WHERE s.snum = s2.snum
);
```

Opción 2

```sql
SELECT s.sname
FROM student AS s
WHERE s.snum NOT IN (
   SELECT s2.snum
```

```
    FROM student AS s2 INNER JOIN enrolled AS e ON s2.snum = e.snum
);
```

11. **Para cada valor de la edad que aparezca en estudiantes, encuentre el valor del nivel que aparece lo más a menudo posible. Por ejemplo, si hay más estudiantes del nivel del FR con 18 años que en los niveles SR, JR, o SO, usted debe imprimir el par (18, FR).**

```
WITH AgeStandingCounts AS (
   SELECT s.age, s.standing, COUNT(s.standing) AS standing_count
   FROM student AS s
   GROUP BY s.age, s.standing
),
MaxStandingCounts AS (
   SELECT A.age, MAX(standing_count) AS max_count
   FROM AgeStandingCounts AS A
   GROUP BY A.age
)
SELECT A.age, A.standing
FROM AgeStandingCounts AS A INNER JOIN MaxStandingCounts AS M ON A.age = M.age AND
↪  A.standing_count = M.max_count;
```

Análisis de la consulta

1. **AgeStandingCounts**: Esta es una subconsulta que selecciona la edad y el nivel (standing) de cada estudiante, y cuenta cuántos estudiantes hay en cada combinación de edad y nivel. El resultado se agrupa por edad y nivel.
2. **MaxStandingCounts**: Esta es otra subconsulta que selecciona la edad y el recuento máximo de estudiantes en cada nivel para cada edad. El resultado se agrupa por edad.
3. La consulta principal luego selecciona la edad y el nivel de la subconsulta `AgeStandingCounts` donde la edad y el recuento de estudiantes en ese nivel coinciden con la edad y el recuento máximo de estudiantes en ese nivel en la subconsulta `MaxStandingCounts`.

En resumen, esta consulta devuelve la edad y el nivel que tiene la mayor cantidad de estudiantes para cada edad.

4. Dado el siguiente esquema relacional:

- emp(eid: integer, ename: string, age: integer, salary: real)
- works(eid: integer, did: integer, pct time: integer)
- dept(did: integer, dname: string, budget: real, managerid: integer)

¡Atención! Tener en cuenta que un empleado puede trabajar en varios departamentos simultáneamente y como así también puede ser el gerente de varios departamentos al mismo tiempo.

1. **Liste los nombres y las edades de cada empleado que trabajan en los departamentos de Hardware y los departamentos de Software.**

   Opción 1:

```
SELECT e.ename, e.age
FROM emp AS e INNER JOIN works AS w ON e.eid = w.eid INNER JOIN dept AS d ON
↪  w.did = d.did
WHERE d.dname LIKE 'hardware' AND EXISTS (
```

```
    SELECT 1
    FROM works AS w2 INNER JOIN dept AS d2 ON w2.did = d2.did
    WHERE e.eid = w2.eid AND d2.dname LIKE 'software'
);
```

Opción 2:

```
SELECT e.ename, e.age
FROM emp AS e NATURAL JOIN works AS w NATURAL JOIN dept AS d
WHERE d.dname LIKE 'HARDWARE'

INTERSECT

SELECT e.ename, e.age
FROM emp AS e NATURAL JOIN works AS w NATURAL JOIN dept AS d
WHERE d.dname LIKE 'SOFTWARE';
```

2. **Encuentre los managerids de los gerentes (o "managers") que administran solo departamentos con presupuesto mayor a $1.000.000.**

   Nota: Queremos únicamente a los que administren departamentos con un presupuesto mayor al millón. Si un manager administra un departamento con un presupuesto mayor al millón pero al mismo tiempo administra otro con un presupuesto menor, dicho manager queda descartado de los resultados.

Opción 1:

```
SELECT d.managerid
FROM dept AS d

EXCEPT

SELECT d.managerid
FROM dept AS d
WHERE d.budget < 1000000;
```

Opción 2:

```
SELECT DISTINCT d.managerid
FROM dept AS d
WHERE d.budget > 1000000
AND d.managerid NOT IN (
    SELECT d.managerid
    FROM dept AS d
    WHERE d.budget <= 1000000
);
```

Opción 3:

```
  SELECT d.managerid
  FROM dept AS d INNER JOIN emp AS e ON d.managerid = e.eid
```

```
    GROUP BY d.managerid
    HAVING MIN(d.budget) > 1000000
```

3. **Encuentre los enames de los gerentes que administran los departamentos con los mayores presupuestos. Si un gerente administra más de un departamento se debe sumar sus presupuestos.**

   Opción 1:

```
WITH PresupuestoGerente AS (
   SELECT d.managerid, SUM(d.budget) as presupuesto
   FROM dept AS d
   GROUP BY d.managerid
), PresupuestoMayor AS (
   SELECT MAX(pg.presupuesto) AS presupuesto_mayor
   FROM PresupuestoGerente AS pg
)
SELECT e.ename
FROM emp AS e INNER JOIN PresupuestoGerente AS pg ON e.eid = pg.managerid INNER
↪  JOIN PresupuestoMayor AS pm ON pg.presupuesto = pm.presupuesto_mayor;
```

   Opción 2:

```
WITH PresupuestoPorGerente AS (
   SELECT d.managerid, e.ename, SUM(d.budget) as presupuesto
   FROM dept AS d INNER JOIN emp AS e ON d.managerid = e.eid
   GROUP BY d.managerid, e.ename
)
SELECT ppg.ename
FROM PresupuestoPorGerente AS ppg
WHERE ppg.presupuesto = (
   SELECT MAX(ppg2.presupuesto)
   FROM PresupuestoPorGerente AS ppg2
);
```

   Nota: Tuve que cambiar los datos de la tablas porque con la definición que tenía los operadores de agregación no funcionaban bien.

   Recordar que en una consulta SQL que utiliza una función de agregación como `SUM()` o `MAX()`, si deseás seleccionar otras columnas junto con la columna agregada, tenés que agrupar esas otras columnas utilizando la cláusula `GROUP BY`.

4. **Encuentre los managerids de los gerentes que controlan mas de $5.000.000.**

   Opción 1:

```
WITH newt AS (
   SELECT d.managerid, SUM(d.budget) AS presupuesto
   FROM dept AS d
   GROUP BY d.managerid
)
SELECT n.managerid
FROM newt AS n
```

```sql
WHERE EXISTS (
   SELECT 1
   FROM newt AS n2
   WHERE n.managerid = n2.managerid AND n2.presupuesto > 5000000
);
```

Opción 2:

```sql
WITH newt AS (
   SELECT d.managerid, SUM(d.budget) AS presupuesto
   FROM dept AS d
   GROUP BY d.managerid
)
SELECT n.managerid
FROM newt AS n
WHERE n.presupuesto > 5000000;
```

5. **Encuentre los `managerids` de los gerentes que controlan los mayores presupuestos.**

Opción 1:

```sql
WITH newt AS (
   SELECT d.managerid, SUM(d.budget) AS presupuesto
   FROM dept AS d
   GROUP BY d.managerid
)
SELECT n.managerid
FROM newt AS n
WHERE n.presupuesto IN (
   SELECT MAX(n2.presupuesto)
   FROM newt AS n2
);
```

Opción 2:

```sql
WITH PresupuestoGerente AS (
   SELECT d.managerid, SUM(d.budget) AS presupuesto
   FROM dept AS d
   GROUP BY d.managerid
),
PresupuestoMayor AS (
   SELECT MAX(pg.presupuesto) AS presupuesto_mayor
   FROM PresupuestoGerente AS pg
)
SELECT pg.managerid
FROM PresupuestoGerente AS pg, PresupuestoMayor AS pm
WHERE pg.presupuesto = pm.presupuesto_mayor;
```

6. **Encuentre los `enames` de los gerentes que administran solo departamentos con presupuestos de más de $1.000.000, pero al menos un departamento tiene presupuesto menor a $5.000.000.**

Opción 1:

```
-- Quisiera estar en la radio, para ganar mi primer millón.
WITH `más_de_un_millón` AS (
   SELECT d.managerid
   FROM dept AS d INNER JOIN emp AS e ON d.managerid = e.eid
   GROUP BY d.managerid
   HAVING MIN(d.budget) > 1000000
),
menos_de_cinco_millones AS (
   SELECT d.managerid
   FROM dept AS d INNER JOIN emp AS e ON d.managerid = e.eid
   GROUP BY d.managerid
   HAVING MAX(d.budget) < 5000000
)
SELECT e.ename
FROM emp AS e INNER JOIN `más_de_un_millón` AS M1 ON e.eid = M1.managerid INNER
↪  JOIN menos_de_cinco_millones AS m5 ON M1.managerid = m5.managerid;
```

Opción 2:

```
WITH newt AS (
   SELECT d.managerid
   FROM dept AS d INNER JOIN emp AS e ON d.managerid = e.eid
   GROUP BY d.managerid
   HAVING MIN(d.budget) > 1000000
)
SELECT e.ename
FROM emp AS e INNER JOIN newt AS n ON e.eid = n.managerid
WHERE NOT EXISTS (
   SELECT 1
   FROM dept AS d
   WHERE n.managerid = d.managerid AND d.budget > 5000000
);
```

Opción 3:

```
WITH newt AS (
   SELECT d.managerid
   FROM dept AS d INNER JOIN emp AS e ON d.managerid = e.eid
   GROUP BY d.managerid
   HAVING MIN(d.budget) > 1000000
)
SELECT DISTINCT e.ename
FROM emp AS e INNER JOIN newt AS n ON e.eid = n.managerid INNER JOIN dept AS d ON
↪  n.managerid = d.managerid
WHERE d.budget < 5000000;
```