[1 PUNTO] CONSULTA 1

a) Forma explícita

```
SELECT CONCAT (V.nom_comp, ' , ' , S.partido) AS Senadores
FROM votante V
INNER JOIN senador S ON V.dni = S.dni
INNER JOIN pide_c P ON V.dni = P.dni_votante AND P.dia = '2025-03-09'
ORDER BY nom_comp;
```

```
+-----------------------------------+
| Senadores                         |
+-----------------------------------+
| Diana Rodríguez Fernández , DAM   |
| Javier Mangui López , PRO         |
| Yolanda Metu Sueldo , ASIR        |
+-----------------------------------+
3 rows in set (0,00 sec)
```

Utilizo la versión sql standard de Ubuntu en máquina virtual en todas las consultas. Se concatenan el nombre completo y el partido separado con una coma para que aparezcan en la consulta que cruza dos tablas senador y pide_c basadas en el DNI y la tabla pide_c y votante también basadas en el DNI con la especificación de la ficha indicada donde se muestran los resultados con el nombre de Senadores y están ordenados ORDER BY por el nombre por orden alfabético. Se hace de forma explicita como indica el ejercicio por medio de la combinación INNER JOIN. Se utilizan los alias para poder correlacionar correctamente las tablas y columnas para que no haya errores.

b) Forma implícita

```
SELECT CONCAT ( V.nom_comp, ' , ' , S.partido) AS Senadores
FROM votante V,  senador S, pide_c P
WHERE V.dni = S.dni AND V.dni = P.dni_votante AND P.dia = '2025-03-09'
ORDER BY nom_comp;
```

```
+-----------------------------------+
| Senadores                         |
+-----------------------------------+
| Diana Rodríguez Fernández , DAM   |
| Javier Mangui López , PRO         |
| Yolanda Metu Sueldo , ASIR        |
+-----------------------------------+
3 rows in set (0,00 sec)
```

En este caso de manera implícita donde se establecen las consultas por medio de clausula WHERE donde se dan los campos conectados como claves primarias de ambas tablas relacionadas y se marca la especificación de ese día de cita que han pedido, y salen ordenadas por orden alfabético, si no se especifica nada, por defecto sale en orden ascendente (alfabético en este caso). Se hace uso de alias también para el uso correcto de relaciones de tablas y columnas.

[1.25 PUNTOS] CONSULTA 2

a) GROUP BY

```
SELECT C.nombre_centro, COUNT(I.dni) AS num_interventores
FROM centro C
INNER JOIN interventor I ON C.id_centro = I.centro_asignado
GROUP BY C.id_centro, C.nombre_centro
HAVING COUNT(I.dni) > 1
ORDER BY 2 DESC, 1 ASC;
```

```
+---------------------------------------+-------------------+
| nombre_centro                         | num_interventores |
+---------------------------------------+-------------------+
| Centro de Cultura y Ocio Gran Vida    |                 3 |
| Centro de Estudios Superiores Robantes |                 2 |
+---------------------------------------+-------------------+
2 rows in set (0,02 sec)
```

En este caso es una consulta donde se unen dos tablas relacionando las claves primarias entre ambas, extrayendo la información solicitada en dos campos donde se da la condición que exista mas de un interventor relacionado directamente con el dni del interventor que esta en el centro, mostrándose la consulta en orden alfabético nombres y numero de interventores en orden descendente, por las posiciones de las columnas en la ordenación, DESC descenciente y ASC (o sin nada) ascendientes. Se utiliza la función HAVING donde se de la condición de tener un conteo de interventores mayor a 1.

b) Subconsulta

```
SELECT C.nombre_centro, COUNT(I.dni) AS num_interventores
FROM centro C, interventor I
WHERE C.Id_centro =  I.centro_asignado
AND 1<
        (SELECT COUNT(I.dni)
         FROM interventor I
         WHERE C.id_centro = I.centro_asignado)
GROUP BY C.nombre_centro
ORDER BY COUNT(I.dni) DESC, C.nombre_centro;
```

```
+---------------------------------------+-------------------+
| nombre_centro                         | num_interventores |
+---------------------------------------+-------------------+
| Centro de Cultura y Ocio Gran Vida    |                 3 |
| Centro de Estudios Superiores Robantes |                 2 |
+---------------------------------------+-------------------+
2 rows in set (0,01 sec)
```

Se seleccionan dos columnas: nombre_centro de la tabla centro y el recuento de dni de la tabla interventor, con el alias num_interventores. Se filtran los datos para incluir solo aquellos centros que tienen más de un interventor, utilizando una subconsulta desde el AND que cuenta el número de interventores asignados a cada centro y compara si es mayor que 1. Esta condición se expresa como 1< (SELECT COUNT(I.dni) FROM interventor I WHERE C.id_centro = I.centro_asignado) Los datos se agrupan por la función GROUP BY por el nombre del centro (nombre_centro), lo que significa que se calculará el recuento de interventores por centro. Los resultados se ordenan primero por el número de interventores en orden descendente (DESC) y luego alfabéticamente por el nombre del centro. Esto se logra mediante la cláusula ORDER BY COUNT(I.dni) DESC, C.nombre_centro

[1.25 PUNTOS] CONSULTA 3

a) IN

```
SELECT DISTINCT I.id, I.circunscripcion
FROM id_digital I
INNER JOIN vota_sen VS ON I.id = VS.id_voto
WHERE VS.id_voto NOT IN (SELECT DISTINCT S.num_votos FROM senador S
                                        INNER JOIN votante V ON V.dni = S.dni)
AND I.circunscripcion LIKE '%Ladron%' OR ( I.circunscripcion LIKE '_________' AND
I.circunscripcion LIKE 'I%')
ORDER BY I.circunscripcion;
```

```
40 rows in set (0,00 sec)
```

Se seleccionan dos columnas: id y circunscripción de la tabla id_digital. Se utiliza DISTINCT para asegurarse de que solo se devuelvan valores únicos. La consulta utiliza las tablas id_digital, vota_sen, senador y votante. Se establece una relación entre la tabla id_digital y la tabla vota_sen mediante la condición I.id = VS.id_voto. Se excluyen los identificadores que están presentes en la tabla vota_sen pero no están asociados con ningún voto de senador registrado en la tabla senador, utilizando la condición de la columna id_voto que no esté en NOT IN. Esto se hace utilizando una subconsulta que selecciona los números de votos de la tabla senador y los compara con los identificadores de la tabla vota_sen. Se filtran las circunscripciones por LIKE y se ordena para que aparezcan en la consulta por la circunscripción

b) EXISTS

```
SELECT DISTINCT I.id, I.circunscripcion
FROM id_digital I
WHERE  EXISTS (SELECT S.num_votos FROM senador S
                        INNER JOIN votante V ON V.dni = S.dni)
AND I.circunscripcion LIKE '%Ladron%' OR ( I.circunscripcion LIKE '_________' AND
I.circunscripcion LIKE 'I%')
ORDER BY I.circunscripcion;
```

```
40 rows in set (0,00 sec)
```

Se seleccionan dos columnas: id y circunscripcion de la tabla id_digital. Se utiliza DISTINCT para asegurar que solo se devuelvan valores únicos. La condición EXISTS se utiliza para verificar si hay al menos un registro en la subconsulta. En la subconsulta, se seleccionan los números de votos (num_votos) de la tabla senador que están asociados a votantes (votante) a través de la comparación de los números de identificación nacional (dni). Luego, se filtran los registros de la tabla id_digital donde la circunscripción cumpla los requisitos de la consulta. Los resultados se ordenan alfabéticamente por la circunscripción (circunscripcion) utilizando la cláusula ORDER BY.

[1.5 PUNTOS] CONSULTA 4

```
SELECT V.nom_comp AS nombre, I.sueldo
FROM interventor I
INNER JOIN votante V ON I.dni = V.dni
WHERE I.centro_asignado = (SELECT centro_asignado FROM interventor
                                        ORDER BY sueldo ASC
                                        LIMIT 1)
AND I.sueldo >  (SELECT MIN(I.sueldo)
                            FROM interventor I)
ORDER BY I.sueldo ASC;
```

```
+------------------------------+---------+
| nom_comp                     | sueldo  |
+------------------------------+---------+
| Fernando Comisión Permanente | 2100.00 |
+------------------------------+---------+
1 row in set (0,00 sec)
```

Se seleccionan dos columnas: nom_comp de la tabla votante y sueldo de la tabla interventor. Se utilizan las tablas interventor y votante. Se establece una relación entre las tablas utilizando la condición I.dni = V.dni, donde dni es el número de identificación nacional del votante. Se filtran mediante el WHERE los interventores basados en dos condiciones: La primera condición, selecciona solo los interventores que están asignados al mismo centro que el interventor con el sueldo más bajo. Esto se logra mediante una subconsulta que devuelve el centro_asignado del interventor con el sueldo más bajo (utilizando ORDER BY sueldo ASC LIMIT 1). La segunda condición, I.sueldo > (SELECT MIN(I.sueldo) FROM interventor I), selecciona solo los interventores cuyo sueldo es mayor que el sueldo mínimo entre todos los interventores. Esto se realiza mediante una subconsulta que devuelve el sueldo mínimo de la tabla interventor.

[2 PUNTOS] CONSULTA 5

a) 1ª Consulta

```
SELECT S.dni, S.partido, SUM(S.pago_compra_voto) AS Dinero_gastado_envotos
FROM senador S
INNER JOIN id_digital I ON S.dni =  I.senador_corrrupto
WHERE I.comprado =’1’
GROUP BY S.dni, S.partido
ORDER BY 3 DESC;
```

```
+-----------+---------+--------------------------+
| dni       | partido | dinero_gastado_en_votos |
+-----------+---------+--------------------------+
| 78901234G | PHT     |                  7999.92 |
| 89012345H | PHT     |                  4200.00 |
| 67890123F | DAW     |                  2250.00 |
| 01234567J | PRO     |                  1600.00 |
| 12345678A | ASIR    |                  1500.00 |
| 56789012E | DAW     |                  1051.50 |
| 23456789B | ASIR    |                  1000.00 |
| 90123456I | PRO     |                   300.00 |
+-----------+---------+--------------------------+
8 rows in set (0,00 sec)
```

Se seleccionan los datos de 3 columnas utilizando dos tablas senador y id_digital, estableciendo una relación entre ambas tablas mediante condición donde dni en la tabla senador se relaciona directamente con senador corrupto según el diagrama dado. Se filtran los registros de la tabla id_digital donde hemos comprobado previamente que el valor 1 indica que el voto ha sido comprado. Los datos se agrupan por dni y partido de los senadores. A continuación, se calcula la columna pago_compra_voto para cada senador, lo que la suma representará el dinero gastado en votos en la compra de votos por cada senador. Los resultados se ordenan por orden descendiente por la columna 3.

b) 2ª Consulta

```
SELECT partido, SUM(pago_compra_voto) AS Dinero_gastado_en_votos
FROM (SELECT partido, SUM(pago_compra_voto) AS A
            FROM senador S
            LEFT JOIN id_digital I ON S.dni = I.senador_corrupto AND I.comprado = '1'
            GROUP BY partido
            UNION ALL
            SELECT DISTINCT partido, NULL AS A
            FROM senador) AS B
GROUP BY partido
ORDER BY 2 ASC;
```

```
+---------+-------------------------+
| partido | Dinero_gastado_envotos |
+---------+-------------------------+
| DAM     |                    NULL |
| PRO     |                 1900.00 |
| ASIR    |                 2500.00 |
| DAW     |                 3301.50 |
| PHT     |                12199.92 |
+---------+-------------------------+
5 rows in set (0,00 sec)
```

Esta consulta me ha sido complicada de realizar para poder incluir el campo NULL donde no había votos comprados por lo que después de darle muchas vueltas lo he planteado haciendo tablas derivadas por una subconsulta desde el FROM. La subconsulta realiza dos acciones: por un lado calcula la suma del pago_compra_voto para cada partido utilizando un JOIN entre tablas senador e id_digital , agrupándolas por partido almacenándolas en A ya que tienen que llevar las consultas un alias establecido. He utilizado UNION ALL para combinar los resultados con una consulta que nos da los partidos políticos junto con un valor nulo para la columna A dentro de esta subconsulta, ya que de este modo todos los partidos políticos aparecerán en el resultado final, incluso los que no tengan datos de compra de votos asociados. Por otra parte, la consulta principal selecciona el partido político y la suma del dinero gastado en votos para cada partido, agrupando los resultados por partido políticos y los resultados por orden ascendente como indica el enunciado, podríamos haber omitido ASC ya que por defecto salen ascendentes y el numero 2 indica la columna 2 de la consulta principal. Se han utilizado alias en el JOIN para poder relacionar las tablas y en las tablas derivadas se ha utilizado el alias correspondiente que debe añadirse. En definitiva, se combinan los resultados de dos subconsultas con funciones agregadas en una de ellas y combinadas con UNION ALL para poder obtener los resultados de varias subconsultas.

[3 PUNTOS] VISTA Y CONSULTAS 6

a) Vista

```
CREATE OR REPLACE VIEW cuenta_votos AS
SELECT V.dni_senador, COUNT(*) AS votos_obtenidos
FROM vota_sen V, senador S
WHERE V.dni_senador = S.dni AND S.circ_presenta = 'Impunícia'
GROUP BY V.dni_senador;
SELECT * FROM cuenta_votos;
```

```
+-------------+-----------------+
| dni_senador | votos_obtenidos |
+-------------+-----------------+
| 01234567J   |               6 |
| 12345678A   |               7 |
| 34567890C   |               7 |
| 56789012E   |               3 |
| 89012345H   |               7 |
+-------------+-----------------+
5 rows in set (0,00 sec)
```

Se crea la vista llamada cuenta_votos donde se unen dos tablas en una consulta de forma implícita (utilizando alias correspondientes para las tablas y en las relaciones) y se agregan (agrupan) con la función group by por el dni_senador y el número de votos. Se añaden condiciones especiales antes donde especifican una circ_ presenta = 'Impunícia'. A continuación, se consulta la vista seleccionando todas las columnas de la tabla, que ahora es la vista cuenta_votos.

b) Consulta con vista

```
SELECT V.nom_comp AS nombre_senador, CV.votos_obtenidos
FROM cuenta_votos CV
INNER JOIN votante V ON CV.dni = V.dni
WHERE CV.votos_obtenidos = (SELECT MAX(votos_obtenidos)
                            FROM cuenta_votos CV);
```

```
+-------------------------+-----------------+
| nombre_senador          | votos_obtenidos |
+-------------------------+-----------------+
| David Miente Losada     |               7 |
| Carlos Martínez Gómez   |               7 |
| Óscar Gocom Prado       |               7 |
+-------------------------+-----------------+
3 rows in set (0,01 sec)
```

Se trata de una consulta con la vista donde se utilizan alias para poder determinar las tablas tanto de la vista como de la base de datos (votante V(tabla) y cuenta_votos(vista)). Se utiliza una subconsulta desde el WHERE donde se iguala la condición que se da donde haya mayor numero de votos obtenidos en todas aquellas filas donde se dé el máximo valor con la función MAX que determinará tal valor. Aquí aplican todas las condiciones explicadas en la vista anterior ya que es una consulta sobre la vista.

c) Consulta sin vista

```
SELECT V.nom_comp AS nombre_senador, COUNT(VS.id_voto) AS votos_obtenidos
FROM senador S
INNER JOIN vota_sen VS ON S.dni = VS.dni_senador
INNER JOIN votante V ON S.dni = V.dni
WHERE S.circ_presenta = 'Impunícia'
HAVING COUNT(VS.id_voto) = (SELECT MAX(votos_obtenidos)
                            FROM (SELECT COUNT(VS.id_voto) AS votos_obtenidos
                                  FROM senador S
                                  INNER JOIN vota_sen VS ON S.dni = VS.dni_senador
                                  GROUP BY S.dni) AS conteo);
```

```
+-------------------------+-----------------+
| nombre_senador          | votos_obtenidos |
+-------------------------+-----------------+
| David Miente Losada     |               7 |
| Carlos Martínez Gómez   |               7 |
| Óscar Gocom Prado       |               7 |
+-------------------------+-----------------+
3 rows in set (0,01 sec)
```

Se seleccionan dos columnas, el nombre completo del senador (nom_comp) y el recuento de votos obtenidos (votos_obtenidos). Consulta sin vista por lo cual se combinan 3 tablas en una consulta inicial, se utilizan las tablas senador, vota_sen y votante, se establece una relación entre las tablas mediante las cláusulas INNER JOIN utilizando los DNIs de los senadores y los votos, se filtran los registros para incluir solo aquellos senadores cuya circunscripción= 'Impunícia , los datos se agrupan por el dni del senador, luego se plantea una subconsulta desde el HAVING sobre otra subconsulta ambas relacionadas donde se asignan alias por necesidad en las tablas. Además se utiliza la cláusula HAVING para filtrar los resultados basados en el recuento de votos obtenidos por cada senador. Se compara el recuento de votos obtenidos por cada senador con el recuento máximo de votos obtenidos entre todos los senadores. Se hace una subconsulta para calcular el recuento máximo de votos obtenidos entre todos los senadores. Esta subconsulta cuenta los votos obtenidos para cada senador y luego encuentra el máximo de esos recuentos de votos obtenidos. Utilizamos la cláusula HAVING para filtrar los resultados basados en el recuento de votos obtenidos por cada senador. Se compara el recuento de votos obtenidos por cada senador con el recuento máximo de votos obtenidos entre todos los senadores