QUERYING OUR DATABASE

Query 1: Para cada persona que administre más de una empresa, listar el nombre y fecha de nacimiento.

```
--Query 1:
SELECT P.NOMBRE, P.FECHA_NAC
FROM PERSONA P, EMPRESA E, ADMINISTRA A
WHERE A.CIF = E.CIF
    AND A.DNI = P.DNI
GROUP BY P.NOMBRE, P.FECHA_NAC
HAVING COUNT(*) > 1;
```

Para que sea más sencillo de entender, el enunciado podría reescribirse de esta otra forma: "Por cada persona que administre más de una empresa, listar su nombre y su fecha de nacimiento". La expresión *por cada* nos indica que tenemos que particionar los resultados de la consulta por el nombre usando la cláusula GROUP BY. Además, como también queremos mostrar la fecha de nacimiento tenemos que incluir esta en la cláusula GROUP BY. El relativo *que* nos indica la condición que tenemos que aplicar sobre cada partición, es decir, la cláusula HAVING. Con el operador de agregación COUNT(*) > 1 indicamos que queremos obtener aquellas personas que administran más de una empresa. En la cláusula FROM indicamos las tablas PERSONA, EMPRESA y ADMINISTRA que en la cláusula WHERE reunimos mediante las claves foráneas que las interrelacionan dos a dos. (También las podemos reunir en la cláusula FROM mediante la cláusula JOIN). Por último, en el SELECT mostramos solo el nombre y la fecha de nacimiento de la(s) persona(s).

Query 2: Listar el id y la deuda pendiente de los clientes cuya nota sea inferior a 5. Ordenar la consulta descendentemente por deuda pendiente.

```
--Query 2:
SELECT C.IDCLIENTE, A.DEUDA_PEND
FROM CLIENTE C, CALIFICACION A
WHERE A.IDCLIENTE = C.IDCLIENTE
    AND A.NOTA < 5
ORDER BY A.DEUDA_PEND DESC;
```

En la cláusula FROM indicamos las tablas CLIENTE y CALIFICACION que necesitamos para aplicar las condiciones. En la cláusula WHERE reunimos ambas tablas mediante la clave foránea que las relaciona y nos quedamos con los clientes cuya nota sea inferior a 5. En la cláusula ORDER BY ordenamos las tuplas por su deuda pendiente y explicitamos que lo haga de forma descendente con DESC. Por último, en el SELECT mostramos el id de cliente y su deuda pendiente.

Query 3: Listar los clientes que no tienen mecanismos de nivel 2, identificando el tipo de cliente (persona/empresa)

Para esta consulta, se utiliza la negación. Como los mecanismos de nivel 2 son aquellos que pertenecen a la tabla "MEC_TIENE" y a la tabla "MEC_ES", buscamos aquellos clientes que en la tabla "TIENE", tengan algún mecanismo de esas dos tablas. Luego, nos quedamos con todos los clientes que no estén dentro de los anteriores. Se realiza tanto para Personas como

para Empresas, y nos quedamos con la unión de ambas consultas, identificándose con un nuevo campo "TIPO_CLIENTE".

```
--Query 3:
SELECT DNI AS IDENTIFICADOR, NOMBRE, 'PERSONA' AS TIPO_CLIENTE
FROM PERSONA
WHERE IDCLIENTEP NOT IN ( -- Clientes con mecanismo TIENE
SELECT DISTINCT T.IDCLIENTE
FROM TIENE T JOIN MEC_TIENE MT ON T.NOMBREMEC = MT.NOMBREMEC
UNION
--Clientes con mecanismo ES
SELECT DISTINCT T.IDCLIENTE
FROM TIENE T JOIN MEC_ES ME ON T.NOMBREMEC = ME.NOMBREMEC)

UNION

SELECT CIF AS IDENTIFICADOR, NOMBRE, 'EMPRESA' AS TIPO_CLIENTE
FROM EMPRESA
WHERE IDCLIENTEE NOT IN (-- Clientes con mecanismo TIENE
SELECT DISTINCT T.IDCLIENTE
FROM TIENE T JOIN MEC_TIENE MT ON T.NOMBREMEC = MT.NOMBREMEC
UNION
--Clientes con mecanismo ES
SELECT DISTINCT T.IDCLIENTE
FROM TIENE T JOIN MEC_ES ME ON T.NOMBREMEC = ME.NOMBREMEC);
```

Query 4: Cantidad de operaciones del tipo "TRANSFERENCIA" validadas por mecanismos de autenticación y ordenadas de forma descendiente.

Para esta consulta, utilizamos la tabla "OPERACION", utilizando la condición de que el tipo de operación sea "TRANSFERENCIA". Agrupamos por el nombre del mecanismo, y utilizamos un count para determinar la cantidad de estas operaciones que hay en la tabla por cada nombre de mecanismo. Luego, especificamos en el order by, que ordene por la el campo 2 del select ("CANTIDAD TRANSFERENCIAS"), en orden descendiente.

```
--Query 4:
SELECT NOMBRENV2 "NOMBRE MECANISMO", COUNT(*) AS "CANTIDAD
TRANSFERENCIAS"
FROM OPERACION
WHERE UPPER(TIPO_OP) = 'TRANSFERENCIA'
GROUP BY NOMBRENV2, TIPO_OP
ORDER BY 2 DESC;
```

Query 5: Listar las parejas de miembros de la misma unidad familiar.

Para esta consulta, cruzamos la tabla persona consigo misma, para poder obtener las parejas de personas que forman parte de la misma unidad familiar, utilizando como criterio aquellas que tengan el mismo domicilio. Para esto, en la cláusula where, ponemos como condición

que el DNI de la persona 1 sea menor que el de la persona 2, y que la calle y número del domicilio sean iguales.

```
--Query 5:
SELECT P1.NOMBRE, P2.NOMBRE
FROM PERSONA P1, PERSONA P2
WHERE P1.DNI < P2.DNI AND P1.CALLE = P2.CALLE AND P1.NUMERO =
P2.NUMERO;
```

Query 6: Relación de empresas administradas por miembros de la misma unidad familiar

Está consulta es similar a la anterior.
Se realiza un join entre la tabla empresa (para obtener los datos de la empresa administrada) con la tabla administra (para saber qué persona administra esa empresa). Luego, se vuelve a hacer un join nuevamente con la tabla administra, para encontrar los pares de personas que administran una misma empresa. Para esto, se pone como condición que los CIF de las empresas sean iguales, que el DNI de la persona 1 sea menor que el de la persona 2 (para evitar pares duplicados de personas), y que además el par de personas que administran estén dentro del resultado de la subconsulta que busca personas que formen parte de la misma unidad familiar.

```
--Query 6:
SELECT E.CIF, E.NOMBRE
FROM EMPRESA E, ADMINISTRA A1, ADMINISTRA A2
WHERE E.CIF = A1.CIF AND A1.CIF = A2.CIF AND A1.DNI < A2.DNI AND
(A1.DNI, A2.DNI) IN(
    SELECT P1.DNI, P2.DNI
    FROM PERSONA P1, PERSONA P2
    WHERE P1.DNI < P2.DNI AND P1.CALLE = P2.CALLE AND P1.NUMERO =
P2.NUMERO);
```

Query 7: Tipo de canal más utilizado (en cantidad de operaciones realizadas) para cada tipo de cliente:
Para esta consulta, nos quedaremos con la unión de dos consultas, ya que la primera la realizaremos para los clientes del tipo "Persona" y la segunda para las "Empresas". Así podremos identificar qué tipo de cliente es cada uno.
Luego, realizaremos una agrupación dentro de la tabla "OPERACION", para contar cuántas operaciones hubo por cada tipo de canal. Para determinar el tipo de canal, utilizamos un case que determina cómo se llamará cada canal de acuerdo al nombre que tenga en el campo "TIPOCANAL". De esta forma, todos los tipocanal que comiencen con 'ATM', se llamarán 'ATM'. los que comienzan con 'APP' se llamarán 'APP' y los que comienzan con 'WEB' se llamarán "BANCO_WEB".
Luego, se va a realizar un join entre esta subconsulta, y otra que seleccione el número máximo de operaciones realizadas, por la condición de que las cantidades de ambas sean iguales. De esta forma, podremos obtener el tipo de canal para el número máximo de consultas.

```
--QUERY 7:
SELECT 'PERSONA' AS TIPO_CLIENTE, A.TIPOCANAL, B.CANTIDAD
FROM (SELECT TIPOCANAL, COUNT(*) AS CANTIDAD
FROM (SELECT CASE WHEN TIPOCANAL LIKE 'WEB%' THEN 'BANCO_WEB'
WHEN TIPOCANAL LIKE 'ATM%' THEN 'ATM'
WHEN TIPOCANAL LIKE 'APP%' THEN 'APP'
ELSE''END AS TIPOCANAL
FROM OPERACION
WHERE IDCLIENTE IN (SELECT IDCLIENTEP FROM PERSONA))
GROUP BY TIPOCANAL) A

JOIN (SELECT MAX(CANTIDAD) AS CANTIDAD
FROM (SELECT TIPOCANAL, COUNT(*) AS CANTIDAD
FROM (SELECT CASE WHEN TIPOCANAL LIKE 'WEB%' THEN 'BANCO_WEB'
WHEN TIPOCANAL LIKE 'ATM%' THEN 'ATM'
WHEN TIPOCANAL LIKE 'APP%' THEN 'APP'
ELSE''END AS TIPOCANAL
FROM OPERACION
WHERE IDCLIENTE IN (SELECT IDCLIENTEP FROM PERSONA))
GROUP BY TIPOCANAL)) B ON A.CANTIDAD = B.CANTIDAD

UNION

SELECT 'EMPRESA' AS TIPO_CLIENTE, A.TIPOCANAL, B.CANTIDAD
FROM (SELECT TIPOCANAL, COUNT(*) AS CANTIDAD
FROM (SELECT CASE WHEN TIPOCANAL LIKE 'WEB%' THEN 'BANCO_WEB'
WHEN TIPOCANAL LIKE 'ATM%' THEN 'ATM'
WHEN TIPOCANAL LIKE 'APP%' THEN 'APP'
ELSE''END AS TIPOCANAL
FROM OPERACION
WHERE IDCLIENTE IN (SELECT IDCLIENTEE FROM EMPRESA))
GROUP BY TIPOCANAL) A

JOIN (SELECT MAX(CANTIDAD) AS CANTIDAD
FROM (SELECT TIPOCANAL, COUNT(*) AS CANTIDAD
FROM (SELECT CASE WHEN TIPOCANAL LIKE 'WEB%' THEN 'BANCO_WEB'
WHEN TIPOCANAL LIKE 'ATM%' THEN 'ATM'
WHEN TIPOCANAL LIKE 'APP%' THEN 'APP'
ELSE''END AS TIPOCANAL
FROM OPERACION
WHERE IDCLIENTE IN (SELECT IDCLIENTEE FROM EMPRESA))
GROUP BY TIPOCANAL)) B ON A.CANTIDAD = B.CANTIDAD;
```

Query 8: Listar el número de ATM por cada zona, ordenado por cantidad de forma descendente

Para esta consulta, realizamos un join entre las tablas ATM y ZONA, para poder obtener la cantidad de cajeros que hay en cada zona. La condiciòn del join será que los campos CPRO

de ambas tablas sean iguales. Seleccionamos las distintas provincias que están en la tabla ZONA, y realizamos un count del IDCAJERO que está en la tabla ATM, agrupando por provincia y luego ordenamos por el campo 2 (cantidad de atm") de manera descendente.

```
--Query 8:
SELECT DISTINCT PROVINCIA, COUNT (DISTINCT A.IDCAJERO) AS "CANTIDAD
ATM"
FROM ATM A JOIN ZONA Z ON A.CPRO = Z.CPRO
GROUP BY PROVINCIA
ORDER BY 2 DESC;
```

Query 9: Cantidad de bloqueos por mecanismos de autenticación realizados por clientes del tipo Persona.

Para esta consulta, utilizamos la tabla "ABM_CLIENTES", ya que las operaciones de bloqueo se encuentran en ella. En el where, especificamos que solo queremos quedarnos con aquellas operaciones que sean del tipo "Bloqueo", y que hayan sido realizadas por clientes del tipo personas. Para esta última condición, utilizamos una subconsulta, especificando que el id del cliente que realiza la operación esté dentro de la tabla "Persona".
Contamos la cantidad de operaciones con un count, agrupando por el campo "nombremec", y por último ordenamos de manera descendente por el campo 2 del select (Nº de bloqueos).

```
--Query 9:
SELECT NOMBREMEC, COUNT(*) AS " N° DE BLOQUEOS"
FROM ABM_CLIENTES
WHERE UPPER(OPERACION) = 'BLOQUEO' AND IDCLIENTE IN (
    SELECT IDCLIENTEP FROM PERSONA)
GROUP BY NOMBREMEC
ORDER BY 2 DESC;
```

Query 10: Cantidad de operaciones realizadas por cada persona, clasificada por tipo, año y mecanismo utilizado.

Para esta consulta, realizamos un join entre las tablas "Persona" (ya que queremos las operaciones realizadas solo por clientes personas) y la tabla "Operacion". De esta forma obtenemos todos los datos de los clientes que realizaron operaciones. Luego, realizamos un tercer join con una subconsulta, que contiene la unión entre las tablas "MEC_TIENE" y "MEC_ES", ya que estas contienen los nombres de los mecanismos de nivel 2 que son los que se utilizan para validar operaciones.
Luego, seleccionamos el nombre del cliente, el año correspondiente a la fecha de la operación, el nombre del mecanismo utilizado y realizamos un count de las operaciones agrupando por los cuatro campos mencionados.
Por último, realizamos un order by por el campo 3 (año de la operación) de manera descendiente.

```
--Query 10:
SELECT P.NOMBRE, O.TIPO_OP, EXTRACT(YEAR FROM O.FECHA) AS AÑO,
MECS.NOMBREMEC, COUNT(*) AS CANTIDAD
FROM PERSONA P JOIN OPERACION O ON P.IDCLIENTEP = O.IDCLIENTE
```

```
JOIN (SELECT NOMBREMEC, NOMBREMECNV2 FROM MEC_TIENE UNION SELECT
NOMBREMEC, NOMBREMECNV2 FROM MEC_ES) MECS ON O.NOMBRENV2 =
MECS.NOMBREMECNV2
GROUP BY P.NOMBRE, O.TIPO_OP, EXTRACT(YEAR FROM O.FECHA),
MECS.NOMBREMEC
ORDER BY 3 DESC;
```