CONSULTAS Y SUBCONSULTAS

AUTOR:

JAVIER NICOLAS SALAS

YESI ESTEBAN PANTOJA CUELLAR

ENTREGADO A:

ING. BRAYAN IGNACIO ARCOS BURBANO

INSTITUTO TECNOLÓGICO DEL PUTUMAYO

TECNOLOGÍA EN DESARROLLO DE SOFTWARE

QUINTO SEMESTRE

DESARROLLO DE BASE DE DATOS

MOCOA – PUTUMAYO

2024

CONSULTAS

1) Consulta #1: Listar los usuarios junto con su rol correspondiente

Descripción: Esta consulta muestra el nombre de los usuarios junto con el nombre del rol que tiene cada uno.

Consulta SQL:

```
#consulta 1 Listar los usuarios junto con su rol correspondiente
SELECT p.name AS Usuario, r.name AS Rol
FROM people p
INNER JOIN usersRoles ur ON p.idUser = ur.idUser
INNER JOIN roles r ON ur.idRole = r.id;
```

Explicación:

1. **Tablas involucradas**:
   - `people` (`p`): contiene los datos de los usuarios, incluyendo el campo `name`.
   - `usersRoles` (`ur`): actúa como una tabla de relación entre usuarios y roles, con los campos `idUser` y `idRole`.
   - `roles` (`r`): contiene información sobre los roles, incluyendo el campo `name` para el nombre del rol.
2. **Cláusula `INNER JOIN`**:
   - Se usa `INNER JOIN` para combinar las tablas `people`, `usersRoles`, y `roles`:
     - `people` y `usersRoles` se combinan a través del campo `idUser`.
     - `usersRoles` y `roles` se combinan a través del campo `idRole`.
3. **Selección de columnas**:
   - `p.name AS Usuario`: selecciona el nombre del usuario de la tabla `people`.
   - `r.name AS Rol`: selecciona el nombre del rol de la tabla `roles`.

Resultado: Una lista con los nombres de los usuarios y el rol asignado a cada uno.

Usuario	Rol
Camilo	Admin
Luis	Customer
Sofía	Customer
Pablo	Customer
Verónica	Customer
Diego	Customer
Laura	Customer
Paola	Customer
Andrés	Customer
Carolina	Customer
Felipe	Employees

2) Consulta 2: Listar las reservas realizadas por cada usuario, junto con el estado de la reserva

Descripción: Esta consulta muestra las reservas realizadas por cada usuario, incluyendo detalles como la fecha, la hora de la reserva y su estado actual.

Consulta SQL:

```sql
#consulta 2 Listar las reservas realizadas por cada usuario, junto con el estado de la reserva
SELECT
    p.name AS Usuario,
    r.id AS ReservaID,
    r.time AS HoraReserva,
    r.dateReserve AS FechaReserva,
    rs.name AS Estado
FROM
    reserves r
INNER JOIN
    users u ON r.idUser = u.id
INNER JOIN
    people p ON u.id = p.idUser
INNER JOIN
    reserveStates rs ON r.idState = rs.id;
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): almacena las reservas, con detalles como `time` (hora), `dateReserve` (fecha), y `idState` (estado de la reserva).
   - `users` (`u`): contiene información de los usuarios, relacionada con las reservas mediante `idUser`.
   - `people` (`p`): almacena información detallada de las personas, incluyendo el nombre (`name`).
   - `reserveStates` (`rs`): contiene los distintos estados posibles de una reserva.
2. **Cláusula `INNER JOIN`**:
   - Se utiliza `INNER JOIN` para conectar las tablas:
     - `reserves` y `users` se enlazan mediante el campo `idUser`.
     - `users` y `people` se enlazan mediante `idUser`.
     - `reserves` y `reserveStates` se enlazan mediante el campo `idState`.
3. **Selección de columnas**:
   - `p.name AS Usuario`: muestra el nombre del usuario de la tabla `people`.
   - `r.id AS ReservaID`: identifica cada reserva con su ID.
   - `r.time AS HoraReserva`: especifica la hora en que se realizó la reserva.
   - `r.dateReserve AS FechaReserva`: especifica la fecha en que se realizó la reserva.

- `rs.name AS Estado`: muestra el estado actual de la reserva.

Resultado: Devuelve una lista de reservas con la información del usuario, fecha, hora y estado de cada reserva.

	Usuario	ReservaID	HoraReserva	FechaReserva	Estado
▶	Luis	1	10:00:00	2024-11-01	Confirmado
	Sofía	2	11:00:00	2024-11-01	Confirmado
	Verónica	4	13:00:00	2024-11-01	Confirmado
	Diego	5	14:00:00	2024-11-01	Confirmado
	Paola	7	16:00:00	2024-11-01	Confirmado
	Andrés	8	17:00:00	2024-11-01	Confirmado
	Luis	10	19:00:00	2024-11-02	Confirmado
	Pablo	3	12:00:00	2024-11-01	Pendiente
	Laura	6	15:00:00	2024-11-01	Pendiente
	Carolina	9	18:00:00	2024-11-01	Pendiente

3) Consulta 3: Consultar los productos reservados junto con los detalles de la reserva y el nombre del cliente

Descripción: Esta consulta muestra los productos reservados por cada cliente, incluyendo detalles como la cantidad reservada, el nombre del producto, el precio, la hora y la fecha de la reserva.

Consulta SQL:

```sql
#consulta 3 Consultar los productos reservados junto con los detalles de la reserva y el nombre del cliente
SELECT
    p.name AS Cliente,
    r.id AS ReservaID,
    r.time AS HoraReserva,
    r.dateReserve AS FechaReserva,
    rp.quantity AS Cantidad,
    prod.name AS Producto,
    prod.price AS Precio
FROM
    reserveProducts rp
INNER JOIN
    reserves r ON rp.idReserve = r.id
INNER JOIN
    users u ON r.idUser = u.id
INNER JOIN
    people p ON u.id = p.idUser
INNER JOIN
    products prod ON rp.idProduct = prod.id;
```

Explicación:

1. **Tablas involucradas**:
    - `reserveProducts` (`rp`): contiene las relaciones entre las reservas y los productos, con campos como `idReserve` (ID de reserva) y `idProduct` (ID del producto) además de `quantity` (cantidad reservada).
    - `reserves` (`r`): almacena la información de cada reserva, incluyendo la fecha (`dateReserve`) y la hora (`time`).
    - `users` (`u`): almacena información de usuarios, relacionada con `reserves` mediante `idUser`.
    - `people` (`p`): almacena los detalles de las personas, incluyendo el nombre del cliente.
    - `products` (`prod`): contiene la información de cada producto, incluyendo el nombre (`name`) y el precio (`price`).
2. **Cláusula `INNER JOIN`**:
    - Se utiliza `INNER JOIN` para unir las tablas:
        - `reserveProducts` y `reserves` se enlazan mediante `idReserve`.
        - `reserves` y `users` se enlazan mediante `idUser`.
        - `users` y `people` se enlazan mediante `idUser`.
        - `reserveProducts` y `products` se enlazan mediante `idProduct`.
3. **Selección de columnas**:
    - `p.name AS Cliente`: muestra el nombre del cliente de la tabla `people`.
    - `r.id AS ReservaID`: identifica cada reserva con su ID.
    - `r.time AS HoraReserva`: especifica la hora en que se realizó la reserva.
    - `r.dateReserve AS FechaReserva`: especifica la fecha en que se realizó la reserva.
    - `rp.quantity AS Cantidad`: muestra la cantidad de cada producto reservado.
    - `prod.name AS Producto`: muestra el nombre del producto reservado.
    - `prod.price AS Precio`: muestra el precio del producto reservado.

Resultado: Genera una lista de productos reservados por cada cliente, incluyendo el nombre del producto, su cantidad, el precio, y los detalles de la reserva.

Cliente	ReservaID	HoraReserva	FechaReserva	Cantidad	Producto	Precio
Luis	1	10:00:00	2024-11-01	1	Cancha de Fútbol 11 - Estadio	1000
Laura	6	15:00:00	2024-11-01	1	Cancha de Fútbol 11 - Estadio	1000
Sofía	2	11:00:00	2024-11-01	1	Cancha de Fútbol 7 - Polideportivo	800
Paola	7	16:00:00	2024-11-01	1	Cancha de Fútbol 7 - Polideportivo	800
Pablo	3	12:00:00	2024-11-01	1	Cancha de Fútbol 6 - Complejo Deportivo	600
Andrés	8	17:00:00	2024-11-01	1	Cancha de Fútbol 6 - Complejo Deportivo	600
Verónica	4	13:00:00	2024-11-01	1	Cancha de Fútbol 5 - Mini Estadio	500
Carolina	9	18:00:00	2024-11-01	1	Cancha de Fútbol 5 - Mini Estadio	500
Diego	5	14:00:00	2024-11-01	1	Cancha de Voleibol - Arena	300
Luis	10	19:00:00	2024-11-02	1	Cancha de Voleibol - Arena	300

4) Consulta 4: Obtener el nombre de los clientes que han reservado productos específicos (Cancha de Fútbol 5 - Mini Estadio)

Descripción: Esta consulta obtiene el nombre de los clientes que han reservado un producto específico, en este caso, la "Cancha de Fútbol 5 - Mini Estadio".

Consulta SQL:

```sql
#consulta 4 Obtener el nombre de los clientes que han reservado productos específicos (Cancha de Fútbol 5 - Mini Estadio)
SELECT
    p.name AS Cliente
FROM
    reserveProducts rp
INNER JOIN
    reserves r ON rp.idReserve = r.id
INNER JOIN
    users u ON r.idUser = u.id
INNER JOIN
    people p ON u.id = p.idUser
INNER JOIN
    products prod ON rp.idProduct = prod.id
WHERE
    prod.name = 'Cancha de Fútbol 5 - Mini Estadio';
```

Explicación:

1. **Tablas involucradas**:
   - `reserveProducts` (`rp`): contiene la relación entre las reservas y los productos, vinculando cada reserva con los productos específicos reservados.
   - `reserves` (`r`): almacena los datos de cada reserva.
   - `users` (`u`): almacena información sobre los usuarios, conectada con la reserva mediante `idUser`.
   - `people` (`p`): contiene los datos de cada cliente, incluyendo su nombre.
   - `products` (`prod`): almacena la información de cada producto, como su nombre (`name`).
2. **Cláusula `INNER JOIN`**:
   - Se emplea `INNER JOIN` para combinar las tablas y obtener solo los registros que coinciden:
     - `reserveProducts` y `reserves` se enlazan mediante `idReserve`.
     - `reserves` y `users` se enlazan mediante `idUser`.
     - `users` y `people` se enlazan mediante `idUser`.
     - `reserveProducts` y `products` se enlazan mediante `idProduct`.
3. **Condición `WHERE`**:
   - `WHERE prod.name = 'Cancha de Fútbol 5 - Mini Estadio'`: filtra los resultados para obtener solo los clientes que han reservado la "Cancha de Fútbol 5 - Mini Estadio".
4. **Selección de columnas**:

- `p.name AS Cliente`: muestra el nombre del cliente de la tabla `people` que ha reservado el producto especificado.

Resultado: Devuelve una lista con los nombres de los clientes que han reservado la "Cancha de Fútbol 5 - Mini Estadio".

	Cliente
▶	Verónica
	Carolina

5) Consulta 5: Ver los métodos de pago utilizados por los clientes en sus reservas

Descripción: Esta consulta muestra los métodos de pago que los clientes han utilizado en sus reservas.

Consulta SQL:

```sql
#consulta 5 Ver los métodos de pago utilizados por los clientes en sus reservas
SELECT
    p.name AS Cliente,
    pm.name AS MetodoPago
FROM
    paymentTransactionInformation pt
INNER JOIN
    reserves r ON pt.idCustomers = r.idUser
INNER JOIN
    users u ON pt.idCustomers = u.id
INNER JOIN
    people p ON u.id = p.idUser
INNER JOIN
    paymentMethods pm ON pt.idPayMethods = pm.id;
```

Explicación:

1. **Tablas involucradas**:
   - `paymentTransactionInformation` (`pt`): contiene la información de transacciones de pago, con campos que identifican al cliente (`idCustomers`) y al método de pago (`idPayMethods`).
   - `reserves` (`r`): almacena las reservas realizadas por los usuarios, enlazada con los clientes mediante `idUser`.
   - `users` (`u`): almacena la información de cada usuario.
   - `people` (`p`): contiene los datos de cada cliente, incluyendo el nombre.

- `paymentMethods` (`pm`): almacena los métodos de pago disponibles, como `name` (nombre del método de pago).

2. **Cláusula `INNER JOIN`**:
   - `INNER JOIN` combina las tablas y muestra solo los registros con coincidencias:
     - `paymentTransactionInformation` y `reserves` se enlazan mediante `idCustomers`.
     - `paymentTransactionInformation` y `users` también se enlazan mediante `idCustomers`.
     - `users` y `people` se enlazan mediante `idUser`.
     - `paymentTransactionInformation` y `paymentMethods` se enlazan mediante `idPayMethods`.
3. **Selección de columnas**:
   - `p.name AS Cliente`: muestra el nombre del cliente de la tabla `people`.
   - `pm.name AS MetodoPago`: muestra el nombre del método de pago utilizado por el cliente.

Resultado: Devuelve una lista de clientes junto con los métodos de pago que han utilizado en sus reservas.

Cliente	MetodoPago
Luis	Tarjeta de Crédito
Luis	Tarjeta de Crédito
Laura	Tarjeta de Crédito
Sofía	Tarjeta de Débito
Paola	Tarjeta de Débito
Pablo	Transferencia Bancaria
Andrés	Transferencia Bancaria
Verónica	Efectivo
Carolina	Efectivo
Diego	PayPal

6) Consulta 6: Mostrar todos los productos reservados por fecha de reserva

Descripción: Esta consulta lista todos los productos reservados, ordenados por la fecha de la reserva, e incluye la cantidad de cada producto reservado.

Consulta SQL:

```sql
# consulta 6 Mostrar todos los productos reservados por fecha de reserva
SELECT
    p.name AS Producto,
    r.dateReserve AS FechaReserva,
    rp.quantity AS CantidadReservada
FROM
    reserveProducts rp
INNER JOIN
    reserves r ON rp.idReserve = r.id
INNER JOIN
    products p ON rp.idProduct = p.id
ORDER BY
    r.dateReserve;
```

Explicación:

1. **Tablas involucradas**:
   - `reserveProducts` (`rp`): relaciona cada reserva con los productos reservados, incluyendo la cantidad (`quantity`) de cada producto.
   - `reserves` (`r`): almacena las reservas realizadas, con el campo `dateReserve` que indica la fecha de reserva.
   - `products` (`p`): contiene la información de cada producto, como su nombre (`name`).
2. **Cláusula `INNER JOIN`**:
   - `INNER JOIN` conecta las tablas para obtener solo registros con coincidencias:
     - `reserveProducts` y `reserves` se enlazan mediante `idReserve`.
     - `reserveProducts` y `products` se enlazan mediante `idProduct`.
3. **Selección de columnas**:
   - `p.name AS Producto`: muestra el nombre del producto reservado.
   - `r.dateReserve AS FechaReserva`: muestra la fecha de la reserva.
   - `rp.quantity AS CantidadReservada`: muestra la cantidad reservada del producto.
4. **Ordenación (`ORDER BY`)**:
   - `ORDER BY r.dateReserve`: ordena los resultados por la fecha de reserva en orden ascendente, mostrando primero las reservas más antiguas.

Resultado: Devuelve una lista de productos reservados con su cantidad y la fecha de reserva, ordenada cronológicamente.

	Producto	FechaReserva	CantidadReservada
▶	Cancha de Fútbol 11 - Estadio	2024-11-01	1
	Cancha de Fútbol 11 - Estadio	2024-11-01	1
	Cancha de Fútbol 7 - Polideportivo	2024-11-01	1
	Cancha de Fútbol 7 - Polideportivo	2024-11-01	1
	Cancha de Fútbol 6 - Complejo Deportivo	2024-11-01	1
	Cancha de Fútbol 6 - Complejo Deportivo	2024-11-01	1
	Cancha de Fútbol 5 - Mini Estadio	2024-11-01	1
	Cancha de Fútbol 5 - Mini Estadio	2024-11-01	1
	Cancha de Voleibol - Arena	2024-11-01	1
	Cancha de Voleibol - Arena	2024-11-02	1

7) Consulta 7: Mostrar las reservas confirmadas con sus respectivas canchas

Descripción: Esta consulta obtiene las reservas que tienen el estado "Confirmado", junto con la fecha, hora de la reserva y el nombre de la cancha reservada.

Consulta SQL:

```sql
#  consulta 7 Mostrar las reservas confirmadas con sus respectivas canchas
SELECT
    r.id AS ReservaID,
    r.dateReserve AS FechaReserva,
    r.time AS HoraReserva,
    p.name AS Cancha,
    rs.name AS EstadoReserva
FROM
    reserves r
INNER JOIN
    reserveStates rs ON r.idState = rs.id
INNER JOIN
    reserveProducts rp ON r.id = rp.idReserve
INNER JOIN
    products p ON rp.idProduct = p.id
WHERE
    rs.name = 'Confirmado'
ORDER BY
    r.dateReserve, r.time;
```

Explicación:

1. **Tablas involucradas**:

- `reserves` (`r`): contiene la información de cada reserva, incluyendo la fecha (`dateReserve`) y la hora (`time`).
- `reserveStates` (`rs`): define los diferentes estados de las reservas, enlazado con `reserves` mediante `idState`.
- `reserveProducts` (`rp`): relaciona las reservas con los productos reservados, enlazado con `reserves` mediante `idReserve`.
- `products` (`p`): contiene la información de cada producto (cancha) reservada, identificada por su nombre (`name`).

2. **Cláusula `INNER JOIN`**:
   - Se usa `INNER JOIN` para combinar las tablas y obtener solo los registros con coincidencias:
     - `reserves` y `reserveStates` se enlazan mediante `idState`.
     - `reserves` y `reserveProducts` se enlazan mediante `idReserve`.
     - `reserveProducts` y `products` se enlazan mediante `idProduct`.
3. **Condición `WHERE`**:
   - `WHERE rs.name = 'Confirmado'`: filtra los resultados para mostrar solo las reservas que están en estado "Confirmado".
4. **Selección de columnas**:
   - `r.id AS ReservaID`: muestra el identificador de la reserva.
   - `r.dateReserve AS FechaReserva`: muestra la fecha en la que se realizó la reserva.
   - `r.time AS HoraReserva`: muestra la hora en la que se realizó la reserva.
   - `p.name AS Cancha`: muestra el nombre de la cancha reservada.
   - `rs.name AS EstadoReserva`: muestra el estado de la reserva, que en este caso siempre será "Confirmado" por la condición `WHERE`.
5. **Ordenación (`ORDER BY`)**:
   - `ORDER BY r.dateReserve, r.time`: organiza las reservas confirmadas en orden cronológico, primero por fecha de reserva y luego por hora.

Resultado: Devuelve una lista de reservas confirmadas con la fecha, hora y la cancha reservada.

ReservaID	FechaReserva	HoraReserva	Cancha	EstadoReserva
1	2024-11-01	10:00:00	Cancha de Fútbol 11 - Estadio	Confirmado
2	2024-11-01	11:00:00	Cancha de Fútbol 7 - Polideportivo	Confirmado
4	2024-11-01	13:00:00	Cancha de Fútbol 5 - Mini Estadio	Confirmado
5	2024-11-01	14:00:00	Cancha de Voleibol - Arena	Confirmado
7	2024-11-01	16:00:00	Cancha de Fútbol 7 - Polideportivo	Confirmado
8	2024-11-01	17:00:00	Cancha de Fútbol 6 - Complejo Deportivo	Confirmado
10	2024-11-02	19:00:00	Cancha de Voleibol - Arena	Confirmado

8) Consulta 8: Listar los usuarios que han utilizado un descuento en sus pagos

Descripción: Esta consulta muestra una lista de usuarios que han aplicado un descuento en sus pagos, junto con el monto del descuento utilizado.

Consulta SQL:

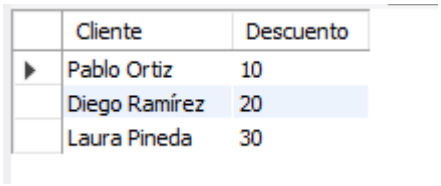
```sql
# consulta 8 Listar los usuarios que han utilizado un descuento en sus pagos
SELECT
    CONCAT(pe.name, ' ', pe.lastname) AS Cliente,
    d.total AS Descuento
FROM
    payment p
INNER JOIN
    paymentTransactionInformation pti ON p.idTransaction = pti.id
INNER JOIN
    users u ON pti.idCustomers = u.id
INNER JOIN
    people pe ON u.id = pe.idUser
INNER JOIN
    discount d ON p.idDiscount = d.id
WHERE
    p.idDiscount IS NOT NULL;
```

Explicación:

1. **Tablas involucradas**:
    - `payment` (`p`): almacena la información de los pagos, incluyendo el identificador del descuento (`idDiscount`).
    - `paymentTransactionInformation` (`pti`): contiene información detallada sobre las transacciones de pago, enlazada a la tabla `payment` mediante `idTransaction`.
    - `users` (`u`): guarda datos sobre los usuarios, enlazados mediante el campo `idCustomers` de `paymentTransactionInformation`.
    - `people` (`pe`): contiene información sobre los clientes, incluyendo su nombre (`name`) y apellido (`lastname`).
    - `discount` (`d`): almacena detalles de los descuentos, enlazada a la tabla `payment` mediante `idDiscount`.
2. **Cláusula `INNER JOIN`**:
    - Se utiliza `INNER JOIN` para conectar las tablas y mostrar solo los registros con coincidencias:
        - `payment` y `paymentTransactionInformation` se enlazan mediante `idTransaction`.
        - `paymentTransactionInformation` y `users` se enlazan mediante `idCustomers`.
        - `users` y `people` se enlazan mediante `idUser`.

        - `payment` y `discount` se enlazan mediante `idDiscount`.
3. **Condición `WHERE`**:
    - `WHERE p.idDiscount IS NOT NULL`: filtra los resultados para incluir solo aquellos registros donde se ha aplicado un descuento.
4. **Selección de columnas**:
    - `CONCAT(pe.name, ' ', pe.lastname) AS Cliente`: combina el nombre y apellido del cliente en un solo campo, mostrando el nombre completo.
    - `d.total AS Descuento`: muestra el total del descuento aplicado.

Resultado: Devuelve una lista de clientes que han utilizado un descuento en sus pagos, junto con el monto del descuento aplicado.



Cliente	Descuento
Pablo Ortiz	10
Diego Ramírez	20
Laura Pineda	30

9) Consulta 9: Ver los detalles de los pagos realizados, incluyendo descuentos y métodos de pago

Descripción: Esta consulta proporciona información detallada sobre los pagos realizados, incluyendo el cliente, el total pagado, los descuentos aplicados, los métodos de pago y el estado del pago.

Consulta SQL:

```sql
# consulta 9 Ver los detalles de los pagos realizados, incluyendo descuentos y métodos de pago
SELECT
    pti.id AS ID_Transaccion,
    CONCAT(pe.name, ' ', pe.lastname) AS Cliente,
    p.totalPay AS Total_Pagado,
    d.total AS Descuento_Aplicado,
    pm.name AS Metodo_Pago,
    sp.name AS Estado_Pago
FROM
    payment p
INNER JOIN
    paymentTransactionInformation pti ON p.idTransaction = pti.id
INNER JOIN
    users u ON pti.idCustomers = u.id
INNER JOIN
    people pe ON u.id = pe.idUser
LEFT JOIN
    discount d ON p.idDiscount = d.id
INNER JOIN
    paymentMethods pm ON pti.idPayMethods = pm.id
INNER JOIN
    statusPayment sp ON pti.idStatus = sp.id
ORDER BY
    pti.id;
```

Explicación:

1. **Tablas involucradas**:
    - `payment` (`p`): almacena la información sobre los pagos, incluyendo el total pagado (`totalPay`) y el identificador del descuento (`idDiscount`).
    - `paymentTransactionInformation` (`pti`): contiene detalles sobre las transacciones de pago, enlazada a `payment` mediante `idTransaction`.
    - `users` (`u`): guarda datos de los usuarios, enlazados a `paymentTransactionInformation` mediante `idCustomers`.
    - `people` (`pe`): contiene información del cliente, como su nombre (`name`) y apellido (`lastname`).
    - `discount` (`d`): almacena los descuentos, enlazada a `payment` mediante `idDiscount` (se utiliza `LEFT JOIN` para incluir pagos sin descuento).
    - `paymentMethods` (`pm`): contiene los métodos de pago, enlazada a `paymentTransactionInformation` mediante `idPayMethods`.
    - `statusPayment` (`sp`): define los estados de los pagos, enlazada a `paymentTransactionInformation` mediante `idStatus`.
2. **Cláusula `INNER JOIN` y `LEFT JOIN`**:
    - Se utiliza `INNER JOIN` para conectar las tablas donde se requiere que haya coincidencias.

- `LEFT JOIN` se usa para la tabla `discount` para incluir registros de pagos que no tengan un descuento asociado.

3. **Selección de columnas**:
   - `pti.id AS ID_Transaccion`: muestra el identificador de la transacción.
   - `CONCAT(pe.name, ' ', pe.lastname) AS Cliente`: combina el nombre y apellido del cliente.
   - `p.totalPay AS Total_Pagado`: muestra el total pagado por el cliente.
   - `d.total AS Descuento_Aplicado`: muestra el total del descuento aplicado (puede ser `NULL` si no se aplicó descuento).
   - `pm.name AS Metodo_Pago`: muestra el nombre del método de pago utilizado.
   - `sp.name AS Estado_Pago`: muestra el estado del pago.
4. **Ordenación (`ORDER BY`)**:
   - `ORDER BY pti.id`: organiza los resultados por el identificador de la transacción.

Resultado: Devuelve una lista de detalles de los pagos realizados, incluyendo el cliente, el total pagado, descuentos, métodos de pago y estado de cada pago.

ID_Transaccion	Cliente	Total_Pagado	Descuento_Aplicado	Metodo_Pago	Estado_Pago
1	Luis Moreno	50.00	NULL	Tarjeta de Crédito	Completado
2	Sofía Herrera	30.00	NULL	Tarjeta de Débito	Pendiente
3	Pablo Ortiz	70.00	10	Transferencia Bancaria	Completado
4	Verónica Castro	100.00	NULL	Efectivo	Completado
5	Diego Ramírez	60.00	20	PayPal	Completado
6	Laura Pineda	90.00	30	Tarjeta de Crédito	Completado
7	Paola Mendoza	45.00	NULL	Tarjeta de Débito	Pendiente
8	Andrés Torres	25.00	NULL	Transferencia Bancaria	Completado
9	Carolina Barrera	55.00	NULL	Efectivo	Completado

10) Consulta 10: Mostrar los productos de todas las categorías

Descripción: Esta consulta lista todos los productos junto con sus respectivas categorías y precios.

Consulta SQL:

```sql
# consulta 10 Mostrar los productos de todas las categorías
SELECT c.name AS Categoria, pr.name AS Producto, pr.price AS Precio
FROM productCategories c
INNER JOIN products pr ON pr.idProductCategory = c.id;
```

Explicación:

1. **Tablas involucradas**:

- `productCategories` (`c`): almacena información sobre las categorías de productos, incluyendo el nombre de la categoría (`name`).
- `products` (`pr`): contiene información sobre los productos, incluyendo el nombre del producto (`name`) y su precio (`price`). Además, está enlazada a `productCategories` a través del campo `idProductCategory`.

2. **Cláusula `INNER JOIN`**:
   - Se utiliza `INNER JOIN` para conectar las tablas, asegurando que se muestren solo aquellos registros de productos que tienen una categoría asociada.
3. **Selección de columnas**:
   - `c.name AS Categoria`: muestra el nombre de la categoría del producto.
   - `pr.name AS Producto`: muestra el nombre del producto.
   - `pr.price AS Precio`: muestra el precio del producto.

Resultado: Devuelve una lista de productos con su respectiva categoría y precio, permitiendo ver la relación entre productos y categorías.

	Categoria	Producto	Precio
▶	Cancha de Fútbol 11	Cancha de Fútbol 11 - Estadio	1000
	Cancha de Fútbol 5	Cancha de Fútbol 5 - Mini Estadio	500
	Cancha de Fútbol 6	Cancha de Fútbol 6 - Complejo Deportivo	600
	Cancha de Fútbol 7	Cancha de Fútbol 7 - Polideportivo	800
	Cancha de Voleibol	Cancha de Voleibol - Arena	300

11)Consulta 11: Consultar las reservas pendientes y sus productos

Descripción: Esta consulta proporciona una lista de las reservas que están pendientes, junto con los productos asociados y el nombre del cliente.

Consulta SQL:

```sql
# consulta 11 Consultar las reservas pendientes y sus productos
SELECT p.name AS Cliente, pr.name AS Producto, r.dateReserve AS Fecha
FROM reserves r
INNER JOIN reserveStates rs ON r.idState = rs.id
INNER JOIN reserveProducts rp ON r.id = rp.idReserve
INNER JOIN products pr ON rp.idProduct = pr.id
INNER JOIN people p ON r.idUser = p.id
WHERE rs.name = 'Pendiente';
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): almacena la información sobre las reservas, incluyendo el identificador del estado de la reserva (`idState`) y la fecha de reserva (`dateReserve`).

- `reserveStates` (`rs`): contiene los estados de las reservas, como "Pendiente", enlazada a `reserves` mediante `id`.
- `reserveProducts` (`rp`): almacena la relación entre reservas y productos, enlazada a `reserves` mediante `idReserve`.
- `products` (`pr`): contiene información sobre los productos reservados, incluyendo su nombre (`name`).
- `people` (`p`): almacena información del cliente, incluyendo su nombre (`name`), enlazada a `reserves` mediante `idUser`.

2. **Cláusula `INNER JOIN`**:
   - Se utilizan `INNER JOIN` para conectar las tablas, asegurando que solo se muestren registros donde exista una relación válida entre las reservas, estados, productos y clientes.
3. **Condición `WHERE`**:
   - `WHERE rs.name = 'Pendiente'`: filtra los resultados para mostrar solo aquellas reservas que tienen el estado "Pendiente".
4. **Selección de columnas**:
   - `p.name AS Cliente`: muestra el nombre del cliente que realizó la reserva.
   - `pr.name AS Producto`: muestra el nombre del producto reservado.
   - `r.dateReserve AS Fecha`: muestra la fecha en que se realizó la reserva.

Resultado: Devuelve una lista de clientes, los productos que han reservado y las fechas de las reservas que están pendientes.

Cliente	Producto	Fecha
Pablo	Cancha de Fútbol 6 - Complejo Deportivo	2024-11-01
Laura	Cancha de Fútbol 11 - Estadio	2024-11-01
Carolina	Cancha de Fútbol 5 - Mini Estadio	2024-11-01

12)Consulta 12: Ver todas las canchas reservadas junto con la cantidad de horas reservadas

Descripción: Esta consulta muestra una lista de todas las canchas que han sido reservadas, junto con la suma total de horas que han sido reservadas para cada cancha.

Consulta SQL:

```sql
# consulta 12 Ver todas las canchas reservadas junto con la cantidad de horas reservadas
SELECT
    p.name AS Cancha,
    SUM(rp.quantity) AS Horas_Reservadas
FROM
    reserveProducts rp
INNER JOIN
    reserves r ON rp.idReserve = r.id
INNER JOIN
    products p ON rp.idProduct = p.id
WHERE
    p.name LIKE '%Cancha%'
GROUP BY
    p.name;
```

Explicación:

1. **Tablas involucradas**:
   - `reserveProducts` (`rp`): almacena la relación entre reservas y productos, incluyendo la cantidad de horas reservadas (`quantity`).
   - `reserves` (`r`): contiene información sobre las reservas, enlazada a `reserveProducts` mediante `idReserve`.
   - `products` (`p`): almacena información sobre los productos, incluyendo el nombre de la cancha (`name`).
2. **Cláusula `INNER JOIN`**:
   - Se utilizan `INNER JOIN` para conectar las tablas, asegurando que se muestren solo aquellos registros donde existe una relación válida entre reservas, productos y la cantidad de horas reservadas.
3. **Condición `WHERE`**:
   - `WHERE p.name LIKE '%Cancha%'`: filtra los resultados para incluir solo aquellos productos cuyo nombre contenga la palabra "Cancha". Esto asegura que solo se muestren las canchas reservadas.
4. **Función de agregación `SUM`**:
   - `SUM(rp.quantity) AS Horas_Reservadas`: suma la cantidad total de horas reservadas para cada cancha.
5. **Agrupación `GROUP BY`**:
   - `GROUP BY p.name`: agrupa los resultados por el nombre de la cancha para obtener la suma de horas reservadas de manera individual para cada cancha.

Resultado: Devuelve una lista de canchas junto con la cantidad total de horas que han sido reservadas.

	Cancha	Horas_Reservadas
▶	Cancha de Fútbol 11 - Estadio	2
	Cancha de Fútbol 7 - Polideportivo	2
	Cancha de Fútbol 6 - Complejo Deportivo	2
	Cancha de Fútbol 5 - Mini Estadio	2
	Cancha de Voleibol - Arena	2

13)Consulta 13: Listar los métodos de pago utilizados para las reservas confirmadas

Descripción: Esta consulta muestra los métodos de pago utilizados en reservas que han sido confirmadas.

Consulta SQL:

```sql
# consulta 13 Listar los métodos de pago utilizados para las reservas confirmadas
SELECT
    r.id AS Reserva,
    pm.name AS Metodo_Pago
FROM
    reserves r
INNER JOIN
    paymentTransactionInformation pti ON r.idUser = pti.idCustomers
INNER JOIN
    paymentMethods pm ON pti.idPayMethods = pm.id
INNER JOIN
    reserveStates rs ON r.idState = rs.id
WHERE
    rs.name = 'Confirmado';
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): contiene la información de las reservas, incluyendo el identificador del usuario que realizó la reserva (`idUser`).
   - `paymentTransactionInformation` (`pti`): almacena la información de las transacciones de pago, enlazada a `reserves` mediante `idCustomers`.
   - `paymentMethods` (`pm`): contiene los métodos de pago disponibles, enlazada a `paymentTransactionInformation` mediante `idPayMethods`.
   - `reserveStates` (`rs`): almacena los estados de las reservas, enlazada a `reserves` mediante `idState`.
2. **Cláusula `INNER JOIN`**:
   - Se utilizan `INNER JOIN` para conectar las tablas y obtener solo las reservas que tienen métodos de pago válidos y que están confirmadas.
3. **Condición `WHERE`**:

- `WHERE rs.name = 'Confirmado'`: filtra los resultados para incluir solo reservas que han sido confirmadas.

4. **Selección de columnas**:
   - `r.id AS Reserva`: muestra el identificador de la reserva.
   - `pm.name AS Metodo_Pago`: muestra el nombre del método de pago utilizado.

Resultado: Devuelve una lista de reservas confirmadas junto con los métodos de pago utilizados.

Reserva	Metodo_Pago
1	Tarjeta de Crédito
10	Tarjeta de Crédito
2	Tarjeta de Débito
7	Tarjeta de Débito
8	Transferencia Bancaria
4	Efectivo
5	PayPal

14)Consulta 14: Listar los usuarios con más reservas realizadas

Descripción: Esta consulta lista a los usuarios junto con la cantidad total de reservas que han realizado, ordenados de mayor a menor.

Consulta SQL:

```sql
# consulta 14 Listar los usuarios con más reservas realizadas
SELECT p.name AS Cliente, COUNT(r.id) AS TotalReservas
FROM reserves r
INNER JOIN people p ON r.idUser = p.id
GROUP BY p.name
ORDER BY TotalReservas DESC;
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): contiene la información de las reservas, incluyendo el identificador del usuario (`idUser`).
   - `people` (`p`): almacena información sobre los usuarios, enlazada a `reserves` mediante `id`.
2. **Cláusula `INNER JOIN`**:
   - Se utiliza `INNER JOIN` para conectar las tablas y asegurar que solo se muestren registros válidos.
3. **Función de agregación `COUNT`**:

- `COUNT(r.id) AS TotalReservas`: cuenta la cantidad total de reservas por cada usuario.

4. **Agrupación `GROUP BY`**:
   - `GROUP BY p.name`: agrupa los resultados por el nombre del usuario.
5. **Ordenación `ORDER BY`**:
   - `ORDER BY TotalReservas DESC`: ordena los resultados de manera descendente según la cantidad total de reservas.

Resultado: Devuelve una lista de clientes con el total de reservas que han realizado, ordenados de mayor a menor.

Cliente	TotalReservas
Luis	2
Sofía	1
Pablo	1
Verónica	1
Diego	1
Laura	1
Paola	1
Andrés	1
Carolina	1

15)Consulta 15: Ver los productos más reservados

Descripción: Esta consulta muestra los productos que han sido más reservados, junto con la cantidad total de reservas por producto.

Consulta SQL:

```sql
#consulta 15 Ver los productos más reservados
SELECT pr.name AS Producto, COUNT(rp.id) AS TotalReservas
FROM reserveProducts rp
INNER JOIN products pr ON rp.idProduct = pr.id
GROUP BY pr.name
ORDER BY TotalReservas DESC;
```

Explicación:

1. **Tablas involucradas**:
   - `reserveProducts` (`rp`): almacena la relación entre reservas y productos, incluyendo el identificador del producto (`idProduct`).
   - `products` (`pr`): contiene información sobre los productos, enlazada a `reserveProducts` mediante `id`.
2. **Cláusula `INNER JOIN`**:

- Se utiliza `INNER JOIN` para conectar las tablas y asegurarse de que solo se muestren productos válidos.

3. **Función de agregación `COUNT`**:
   - `COUNT(rp.id) AS TotalReservas`: cuenta la cantidad total de reservas por cada producto.
4. **Agrupación `GROUP BY`**:
   - `GROUP BY pr.name`: agrupa los resultados por el nombre del producto.
5. **Ordenación `ORDER BY`**:
   - `ORDER BY TotalReservas DESC`: ordena los resultados de manera descendente según la cantidad total de reservas.

Resultado: Devuelve una lista de productos junto con la cantidad total de reservas que han tenido, ordenados de mayor a menor.

Producto	TotalReservas
Cancha de Fútbol 11 - Estadio	2
Cancha de Fútbol 7 - Polideportivo	2
Cancha de Fútbol 6 - Complejo Deportivo	2
Cancha de Fútbol 5 - Mini Estadio	2
Cancha de Voleibol - Arena	2

16) Consulta 16: Ingresos totales por fecha de reserva

Descripción: Esta consulta muestra los ingresos totales generados por reservas, agrupados por fecha de reserva.

Consulta SQL:

```sql
#consulta 16 Ingresos totales por fecha de reserva
SELECT
    r.dateReserve AS Fecha_Reserva,
    SUM(p.totalPay) AS Ingresos_Totales
FROM
    reserves r
INNER JOIN
    paymentTransactionInformation pti ON r.idUser = pti.idCustomers
INNER JOIN
    payment p ON pti.id = p.idTransaction
GROUP BY
    r.dateReserve;
```

Explicación:

1. **Tablas involucradas**:

- `reserves` (`r`): contiene la información de las reservas, incluyendo la fecha de la reserva (`dateReserve`).
- `paymentTransactionInformation` (`pti`): almacena la información de las transacciones de pago, enlazada a `reserves` mediante `idCustomers`.
- `payment` (`p`): contiene los detalles de los pagos, enlazada a `paymentTransactionInformation` mediante `idTransaction`.

2. **Cláusula `INNER JOIN`**:
   - Se utilizan `INNER JOIN` para conectar las tablas y obtener solo las reservas que tienen pagos válidos.
3. **Función de agregación `SUM`**:
   - `SUM(p.totalPay) AS Ingresos_Totales`: suma los ingresos de los pagos realizados por cada fecha de reserva.
4. **Agrupación `GROUP BY`**:
   - `GROUP BY r.dateReserve`: agrupa los resultados por la fecha de la reserva.

Resultado: Devuelve una lista de fechas de reserva junto con los ingresos totales generados en cada fecha.

	Fecha_Reserva	Ingresos_Totales
▶	2024-11-01	525.00
	2024-11-02	50.00

17) Consulta 17: Usuarios que nunca han hecho una reserva

Descripción: Esta consulta lista a los usuarios que no tienen ninguna reserva registrada.

Consulta SQL:

```sql
# consulta 17 Usuarios que nunca han hecho una reserva
SELECT p.name AS Usuario
FROM people p
LEFT JOIN reserves r ON p.id = r.idUser
WHERE r.id IS NULL;
```

Explicación:

1. **Tablas involucradas**:
   - `people` (`p`): contiene la información de los usuarios.
   - `reserves` (`r`): almacena la información de las reservas, enlazada a `people` mediante `idUser`.
2. **Cláusula `LEFT JOIN`**:
   - Se utiliza `LEFT JOIN` para incluir todos los usuarios, incluso aquellos que no tienen reservas.
3. **Condición `WHERE`**:

- `WHERE r.id IS NULL`: filtra los resultados para incluir solo aquellos usuarios que no tienen ninguna reserva registrada.

Resultado: Devuelve una lista de usuarios que nunca han realizado una reserva.

	Usuario
▶	Felipe
	Camilo

18) Consulta 18: Listar productos reservados en un rango de fechas

Descripción: Esta consulta muestra los productos que han sido reservados dentro de un rango de fechas específico.

Consulta SQL:

```sql
# consulta 18 Listar productos reservados en un rango de fechas
SELECT pr.name AS Producto, r.dateReserve AS Fecha
FROM reserveProducts rp
INNER JOIN products pr ON rp.idProduct = pr.id
INNER JOIN reserves r ON rp.idReserve = r.id
WHERE r.dateReserve BETWEEN '2024-01-01' AND '2024-12-31';
```

Explicación:

1. **Tablas involucradas**:
   - `reserveProducts` (`rp`): almacena la relación entre reservas y productos.
   - `products` (`pr`): contiene información sobre los productos.
   - `reserves` (`r`): almacena la información de las reservas, incluyendo la fecha de reserva.
2. **Cláusula `INNER JOIN`**:
   - Se utilizan `INNER JOIN` para conectar las tablas y asegurar que solo se muestren registros válidos.
3. **Condición `WHERE`**:
   - `WHERE r.dateReserve BETWEEN '2024-01-01' AND '2024-12-31'`: filtra los resultados para incluir solo aquellas reservas que están dentro del rango de fechas especificado.

Resultado: Devuelve una lista de productos junto con las fechas en las que fueron reservados, solo si están dentro del rango indicado.

	Producto	Fecha
▶	Cancha de Fútbol 11 - Estadio	2024-11-01
	Cancha de Fútbol 7 - Polideportivo	2024-11-01
	Cancha de Fútbol 6 - Complejo Deportivo	2024-11-01
	Cancha de Fútbol 5 - Mini Estadio	2024-11-01
	Cancha de Voleibol - Arena	2024-11-01
	Cancha de Fútbol 11 - Estadio	2024-11-01
	Cancha de Fútbol 7 - Polideportivo	2024-11-01
	Cancha de Fútbol 6 - Complejo Deportivo	2024-11-01
	Cancha de Fútbol 5 - Mini Estadio	2024-11-01
	Cancha de Voleibol - Arena	2024-11-02

19) Consulta 19: Promedio de pagos realizados por método de pago

Descripción: Esta consulta muestra el promedio de los pagos realizados, agrupados por método de pago.

Consulta SQL:

```sql
# consulta 19 Promedio de pagos realizados por método de pago
SELECT
    pm.name AS Metodo_Pago,
    ROUND(AVG(p.totalPay), 2) AS Promedio_Pago
FROM
    payment p
INNER JOIN
    paymentTransactionInformation pti ON p.idTransaction = pti.id
INNER JOIN
    paymentMethods pm ON pti.idPayMethods = pm.id
GROUP BY
    pm.name;
```

Explicación:

1. **Tablas involucradas**:
   - `payment` (`p`): contiene los detalles de los pagos.
   - `paymentTransactionInformation` (`pti`): almacena la información de las transacciones de pago, enlazada a `payment` mediante `idTransaction`.
   - `paymentMethods` (`pm`): contiene los métodos de pago disponibles, enlazada a `paymentTransactionInformation` mediante `idPayMethods`.
2. **Cláusula `INNER JOIN`**:
   - Se utilizan `INNER JOIN` para conectar las tablas y obtener solo las transacciones de pago válidas.
3. **Función de agregación `AVG`**:

- `ROUND(AVG(p.totalPay), 2) AS Promedio_Pago`: calcula el promedio de los pagos realizados por cada método de pago y redondea el resultado a dos decimales.

4. **Agrupación `GROUP BY`**:
   - `GROUP BY pm.name`: agrupa los resultados por el nombre del método de pago.

Resultado: Devuelve una lista de métodos de pago junto con el promedio de pagos realizados utilizando cada método.

	Metodo_Pago	Promedio_Pago
▶	Tarjeta de Crédito	70.00
	Tarjeta de Débito	37.50
	Transferencia Bancaria	47.50
	Efectivo	77.50
	PayPal	60.00

20) Consulta 20: Ver las reservas que incluyen descuentos mayores a un valor específico (15)

Descripción: Esta consulta muestra las reservas que tienen descuentos aplicados que superan un valor específico (15).

Consulta SQL:

```sql
# consulta 20 Ver las reservas que incluyen descuentos mayores a un valor específico 15
SELECT
    r.id AS Reserva_ID,
    p.totalPay AS Pago_Total,
    d.total AS Descuento_Aplicado
FROM
    reserves r
INNER JOIN
    paymentTransactionInformation pti ON r.idUser = pti.idCustomers
INNER JOIN
    payment p ON pti.id = p.idTransaction
INNER JOIN
    discount d ON p.idDiscount = d.id
WHERE
    d.total > 15;
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): contiene información sobre las reservas.

- `paymentTransactionInformation` (`pti`): almacena información sobre las transacciones de pago, enlazada a `reserves` mediante `idCustomers`.
- `payment` (`p`): contiene detalles sobre los pagos realizados.
- `discount` (`d`): almacena los detalles de los descuentos aplicados a los pagos.

2. **Cláusulas `INNER JOIN`**:
   - Se utilizan para conectar las tablas y garantizar que solo se muestren reservas que tienen pagos con descuentos.
3. **Condición `WHERE`**:
   - `WHERE d.total > 15`: filtra las reservas para mostrar solo aquellas con descuentos mayores a 15.

Resultado: Devuelve una lista de reservas con el ID de reserva, el total pagado y el descuento aplicado, siempre que el descuento supere 15.

	Reserva_ID	Pago_Total	Descuento_Aplicado
▶	5	60.00	20
	6	90.00	30

21) Consulta 21: Listar los 3 productos con el precio más alto

Descripción: Esta consulta obtiene los tres productos que tienen el precio más alto.

Consulta SQL:

```sql
# consulta 21 Listar los 3 productos con el precio más alto
SELECT pr.name AS Producto, pr.price AS Precio
FROM products pr
ORDER BY pr.price DESC
LIMIT 3;
```

Explicación:

1. **Tabla involucrada**:
   - `products` (`pr`): almacena información sobre los productos, incluyendo su nombre y precio.
2. **Ordenación**:
   - `ORDER BY pr.price DESC`: ordena los productos de mayor a menor precio.
3. **Límite**:
   - `LIMIT 3`: restringe los resultados a los tres productos más caros.

Resultado: Devuelve una lista de los tres productos con el precio más alto, mostrando el nombre del producto y su precio.

	Producto	Precio
▶	Cancha de Fútbol 11 - Estadio	1000
	Cancha de Fútbol 7 - Polideportivo	800
	Cancha de Fútbol 6 - Complejo Deportivo	600

22) Consulta 22: Historial de reservas de un usuario específico

Descripción: Esta consulta muestra el historial de reservas de un usuario específico, en este caso, "Sofía".

Consulta SQL:

```sql
#consulta 22 Historial de reservas de un usuario específico
SELECT r.dateReserve AS Fecha, s.name AS Estado, pr.name AS Producto
FROM reserves r
INNER JOIN reserveStates s ON r.idState = s.id
INNER JOIN reserveProducts rp ON r.id = rp.idReserve
INNER JOIN products pr ON rp.idProduct = pr.id
INNER JOIN people p ON r.idUser = p.id
WHERE p.name = 'Sofía';
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): contiene información sobre las reservas, incluyendo la fecha y el estado.
   - `reserveStates` (`s`): almacena los posibles estados de las reservas.
   - `reserveProducts` (`rp`): conecta las reservas con los productos.
   - `products` (`pr`): información sobre los productos reservados.
   - `people` (`p`): contiene los datos de los usuarios.
2. **Cláusula `INNER JOIN`**:
   - Conecta las tablas para obtener información completa sobre las reservas de "Sofía".
3. **Condición `WHERE`**:
   - `WHERE p.name = 'Sofía'`: filtra los resultados para mostrar solo las reservas realizadas por "Sofía".

Resultado: Devuelve una lista de reservas de "Sofía" con la fecha de reserva, el estado de la reserva y el nombre del producto reservado.

	Fecha	Estado	Producto
▶	2024-11-01	Confirmado	Cancha de Fútbol 7 - Polideportivo

23) Consulta 23: Métodos de pago más usados

Descripción: Esta consulta lista los métodos de pago utilizados, junto con la cantidad de veces que cada uno ha sido utilizado.

Consulta SQL:

```sql
# consulta 23 Métodos de pago más usados
SELECT
    pm.name AS Metodo_Pago,
    COUNT(pti.idPayMethods) AS Cantidad_Usos
FROM
    paymentTransactionInformation pti
INNER JOIN
    paymentMethods pm ON pti.idPayMethods = pm.id
GROUP BY
    pm.name
ORDER BY
    Cantidad_Usos DESC;
```

Explicación:

1. **Tablas involucradas**:
   - `paymentTransactionInformation` (`pti`): contiene información sobre las transacciones de pago.
   - `paymentMethods` (`pm`): almacena los métodos de pago disponibles.
2. **Cláusula `INNER JOIN`**:
   - Se utiliza para conectar las tablas y asociar cada transacción con su método de pago correspondiente.
3. **Función de agregación `COUNT`**:
   - `COUNT(pti.idPayMethods) AS Cantidad_Usos`: cuenta cuántas veces se ha utilizado cada método de pago.
4. **Agrupación `GROUP BY`**:
   - `GROUP BY pm.name`: agrupa los resultados por el nombre del método de pago.
5. **Ordenación**:
   - `ORDER BY Cantidad_Usos DESC`: ordena los resultados de mayor a menor uso.

Resultado: Devuelve una lista de métodos de pago junto con la cantidad de veces que cada uno ha sido utilizado.

Metodo_Pago	Cantidad_Usos
Tarjeta de Crédito	2
Tarjeta de Débito	2
Transferencia Bancaria	2
Efectivo	2
PayPal	1

24) Consulta 24: Ver todas las reservas realizadas en el último mes

Descripción: Esta consulta muestra todas las reservas que se han realizado en el último mes, incluyendo el nombre del cliente, la fecha de la reserva y el estado de la misma.

Consulta SQL:

```sql
# consulta 24 Ver todas las reservas realizadas en el último mes
SELECT p.name AS Cliente, r.dateReserve AS Fecha, s.name AS Estado
FROM reserves r
INNER JOIN people p ON r.idUser = p.id
INNER JOIN reserveStates s ON r.idState = s.id
WHERE r.dateReserve >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH);
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): contiene información sobre las reservas, incluyendo la fecha y el estado.
   - `people` (`p`): almacena datos de los clientes que han realizado las reservas.
   - `reserveStates` (`s`): almacena los diferentes estados de las reservas.
2. **Cláusulas `INNER JOIN`**:
   - Conectan las tablas para obtener detalles completos sobre las reservas.
3. **Condición `WHERE`**:
   - `WHERE r.dateReserve >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)`: filtra las reservas para mostrar solo aquellas realizadas en el último mes.

Resultado: Devuelve una lista de reservas realizadas en el último mes con el nombre del cliente, la fecha de la reserva y el estado correspondiente.

Cliente	Fecha	Estado
Luis	2024-11-01	Confirmado
Sofía	2024-11-01	Confirmado
Verónica	2024-11-01	Confirmado
Diego	2024-11-01	Confirmado
Paola	2024-11-01	Confirmado
Andrés	2024-11-01	Confirmado
Luis	2024-11-02	Confirmado
Pablo	2024-11-01	Pendiente
Laura	2024-11-01	Pendiente
Carolina	2024-11-01	Pendiente

25)Consulta 25: Cantidad total de productos reservados por categoría

Descripción: Esta consulta obtiene la cantidad total de productos reservados, agrupada por categoría.

Consulta SQL:

```sql
# consulta 25 Cantidad total de productos reservados por categoría
SELECT c.name AS Categoria, COUNT(rp.id) AS TotalReservados
FROM reserveProducts rp
INNER JOIN products pr ON rp.idProduct = pr.id
INNER JOIN productCategories c ON pr.idProductCategory = c.id
GROUP BY c.name
ORDER BY TotalReservados DESC;
```

Explicación:

1. **Tablas involucradas**:
   - `reserveProducts` (`rp`): conecta las reservas con los productos.
   - `products` (`pr`): contiene información sobre los productos.
   - `productCategories` (`c`): almacena las categorías de los productos.
2. **Cláusulas `INNER JOIN`**:
   - Se utilizan para conectar las tablas y obtener la relación entre productos, sus categorías y las reservas.
3. **Función de agregación `COUNT`**:
   - `COUNT(rp.id) AS TotalReservados`: cuenta cuántos productos han sido reservados por categoría.
4. **Agrupación `GROUP BY`**:
   - `GROUP BY c.name`: agrupa los resultados por el nombre de la categoría.
5. **Ordenación**:
   - `ORDER BY TotalReservados DESC`: ordena los resultados de mayor a menor cantidad reservada.

Resultado: Devuelve una lista de categorías junto con la cantidad total de productos que han sido reservados en cada una.

Categoria	TotalReservados
Cancha de Fútbol 11	2
Cancha de Fútbol 7	2
Cancha de Fútbol 6	2
Cancha de Fútbol 5	2
Cancha de Voleibol	2

26) Consulta 26: Obtener todos los usuarios y sus roles (si los tienen)

Descripción: Esta consulta obtiene una lista de todos los usuarios junto con sus roles, si tienen alguno asignado.

Consulta SQL:

```sql
# consulta 26 Obtener todos los usuarios y sus roles (si los tienen):
SELECT
    p.name AS Usuario,
    r.name AS Rol
FROM
    users u
LEFT JOIN
    usersRoles ur ON u.id = ur.idUser
INNER JOIN
    people p ON u.id  = p.idUser
LEFT JOIN
    roles r ON ur.idRole = r.id
ORDER BY Rol ASC;
```

Explicación:

1. **Tablas involucradas**:
    - `users` (`u`): almacena información sobre los usuarios del sistema.
    - `usersRoles` (`ur`): conecta usuarios con sus roles.
    - `people` (`p`): contiene datos personales de los usuarios.
    - `roles` (`r`): almacena los diferentes roles disponibles.
2. **Cláusulas `LEFT JOIN`**:
    - Se utilizan para asegurar que todos los usuarios se incluyan en el resultado, incluso si no tienen un rol asignado.
3. **Cláusula `INNER JOIN`**:
    - Conecta la tabla de `people` con `users` para obtener el nombre del usuario.
4. **Ordenación**:
    - `ORDER BY Rol ASC`: ordena los resultados alfabéticamente por el nombre del rol.

Resultado: Devuelve una lista de todos los usuarios y sus roles (si los tienen), mostrando el nombre del usuario y el rol asociado.

Usuario	Rol
Camilo	Admin
Andrés	Customer
Carolina	Customer
Diego	Customer
Laura	Customer
Luis	Customer
Pablo	Customer
Paola	Customer
Sofía	Customer
Verónica	Customer
Felipe	Employees

27) Consulta 27: Listar las reservas junto con el estado de cada reserva

Descripción: Esta consulta muestra todas las reservas junto con su estado correspondiente.

Consulta SQL:

```sql
#consulta 27 Listar las reservas junto con el estado de cada reserva:
SELECT
    r.id AS reserve_id,
    r.time,
    r.dateReserve,
    s.name AS state_name
FROM
    reserves r
LEFT JOIN
    reserveStates s ON r.idState = s.id;
```

Explicación:

1. **Tablas involucradas**:
   - `reserves` (`r`): contiene información sobre las reservas, como el tiempo y la fecha.
   - `reserveStates` (`s`): almacena los diferentes estados de las reservas.
2. **Cláusula `LEFT JOIN`**:
   - Se utiliza para incluir todas las reservas, incluso aquellas que pueden no tener un estado asociado.
3. **Selección de columnas**:
   - Se seleccionan el ID de la reserva, el tiempo de la reserva, la fecha y el nombre del estado.

Resultado: Devuelve una lista de reservas junto con el estado de cada una, incluso si algunas reservas no tienen un estado asignado.

reserve_id	time	dateReserve	state_name
1	10:00:00	2024-11-01	Confirmado
2	11:00:00	2024-11-01	Confirmado
3	12:00:00	2024-11-01	Pendiente
4	13:00:00	2024-11-01	Confirmado
5	14:00:00	2024-11-01	Confirmado
6	15:00:00	2024-11-01	Pendiente
7	16:00:00	2024-11-01	Confirmado
8	17:00:00	2024-11-01	Confirmado
9	18:00:00	2024-11-01	Pendiente
10	19:00:00	2024-11-02	Confirmado

28) Consulta 28: Mostrar todos los productos junto con su categoría

Descripción: Esta consulta obtiene una lista de todos los productos junto con su categoría correspondiente.

Consulta SQL:

```sql
#consulta 28 Mostrar todos los productos junto con su categoría:
SELECT
    p.id AS product_id,
    p.name AS product_name,
    c.name AS category_name
FROM
    products p
LEFT JOIN
    productCategories c ON p.idProductCategory = c.id;
```

Explicación:

1. **Tablas involucradas**:
   - `products` (`p`): almacena información sobre los productos.
   - `productCategories` (`c`): contiene las categorías de los productos.
2. **Cláusula `LEFT JOIN`**:
   - Se utiliza para asegurar que todos los productos se incluyan en el resultado, incluso si no tienen una categoría asignada.
3. **Selección de columnas**:
   - Se seleccionan el ID del producto, el nombre del producto y el nombre de la categoría.

Resultado: Devuelve una lista de todos los productos junto con su categoría, incluso si algunos productos no están asignados a ninguna categoría.

product_id	product_name	category_name
1	Cancha de Fútbol 11 - Estadio	Cancha de Fútbol 11
2	Cancha de Fútbol 7 - Polideportivo	Cancha de Fútbol 7
3	Cancha de Fútbol 6 - Complejo Deportivo	Cancha de Fútbol 6
4	Cancha de Fútbol 5 - Mini Estadio	Cancha de Fútbol 5
5	Cancha de Voleibol - Arena	Cancha de Voleibol

29) Consulta 29: Listar los pagos realizados junto con los métodos de pago y descuentos

Descripción: Esta consulta muestra todos los pagos realizados, incluyendo los métodos de pago utilizados y los descuentos aplicados.

Consulta SQL:

```sql
#consulta 29  Listar los pagos realizados junto con los métodos de pago y descuentos:
SELECT
    p.id AS PagoID,
    pm.name AS MetodoPago,
    d.total AS Descuento,
    p.totalPay AS TotalPago
FROM
    payment p
LEFT JOIN
    discount d ON p.idDiscount = d.id
LEFT JOIN
    paymentTransactionInformation pti ON p.idTransaction = pti.id
LEFT JOIN
    paymentMethods pm ON pti.idPayMethods = pm.id;
```

Explicación:

1. **Tablas involucradas**:
   - `payment` (`p`): almacena información sobre los pagos realizados.
   - `discount` (`d`): contiene información sobre los descuentos aplicados a los pagos.
   - `paymentTransactionInformation` (`pti`): conecta los pagos con los métodos de pago utilizados.
   - `paymentMethods` (`pm`): almacena los diferentes métodos de pago disponibles.
2. **Cláusulas `LEFT JOIN`**:
   - Se utilizan para incluir todos los pagos, incluso aquellos que pueden no tener un descuento o un método de pago asignado.
3. **Selección de columnas**:
   - Se seleccionan el ID del pago, el método de pago utilizado, el descuento aplicado y el total pagado.

Resultado: Devuelve una lista de todos los pagos realizados junto con el método de pago y el descuento aplicados, incluso si algunos pagos no tienen un descuento o método de pago asociado.

PagoID	MetodoPago	Descuento	TotalPago
1	Tarjeta de Crédito	NULL	50.00
2	Tarjeta de Débito	NULL	30.00
3	Transferencia Bancaria	10	70.00
4	Efectivo	NULL	100.00
5	PayPal	20	60.00
6	Tarjeta de Crédito	30	90.00
7	Tarjeta de Débito	NULL	45.00
8	Transferencia Bancaria	NULL	25.00
9	Efectivo	NULL	55.00

30) Consulta 30: Obtener información de personas junto con su tipo de documento

Descripción: Esta consulta obtiene información sobre las personas, incluyendo su tipo de documento.

Consulta SQL:

```sql
#consulta 30 Obtener información de personas junto con su tipo de documento:
SELECT
    p.id AS person_id,
    p.name AS person_name,
    p.lastName,
    td.name AS document_type
FROM
    people p
LEFT JOIN
    type_document td ON p.idTypeDocument = td.id;
```

Explicación:

1. **Tablas involucradas**:
   - `people` (`p`): almacena información sobre las personas.
   - `type_document` (`td`): contiene los diferentes tipos de documentos.
2. **Cláusula `LEFT JOIN`**:
   - Se utiliza para asegurar que se incluyan todas las personas, incluso si no tienen un tipo de documento asociado.
3. **Selección de columnas**:
   - Se seleccionan el ID de la persona, el nombre, el apellido y el tipo de documento.

Resultado: Devuelve una lista de personas junto con su tipo de documento, incluso si algunas personas no tienen un tipo de documento asignado.

person_id	person_name	lastName	document_type
1	Luis	Moreno	CC
2	Sofía	Herrera	CC
3	Pablo	Ortiz	TI
4	Verónica	Castro	CC
5	Diego	Ramírez	TI
6	Laura	Pineda	CC
7	Felipe	Córdoba	TI
8	Paola	Mendoza	CC
9	Andrés	Torres	TI
10	Carolina	Barrera	CC
11	Camilo	Vargas	CE

31) Consulta 31: Productos reservados y sus reservas

Descripción: Esta consulta muestra todos los productos que han sido reservados junto con la información de sus reservas.

Consulta SQL:

```sql
#consulta 31 Productos reservados y sus reservas:
SELECT reserves.id AS reserve_id, products.name AS product_name, reserveProducts.quantity
FROM reserveProducts
RIGHT JOIN reserves ON reserveProducts.idReserve = reserves.id
RIGHT JOIN products ON reserveProducts.idProduct = products.id;
```

Explicación:

1. **Tablas involucradas**:
   - `reserveProducts`: almacena información sobre los productos reservados.
   - `reserves`: contiene información sobre las reservas.
   - `products`: almacena información sobre los productos.
2. **Cláusulas `RIGHT JOIN`**:
   - Se utilizan para asegurar que se incluyan todas las reservas, incluso aquellas que pueden no tener productos asociados.
3. **Selección de columnas**:
   - Se seleccionan el ID de la reserva, el nombre del producto y la cantidad reservada.

Resultado: Devuelve una lista de productos reservados junto con la información de sus reservas, incluso si algunas reservas no tienen productos asociados.

reserve_id	product_name	quantity
1	Cancha de Fútbol 11 - Estadio	1
6	Cancha de Fútbol 11 - Estadio	1
2	Cancha de Fútbol 7 - Polideportivo	1
7	Cancha de Fútbol 7 - Polideportivo	1
3	Cancha de Fútbol 6 - Complejo Deportivo	1
8	Cancha de Fútbol 6 - Complejo Deportivo	1
4	Cancha de Fútbol 5 - Mini Estadio	1
9	Cancha de Fútbol 5 - Mini Estadio	1
5	Cancha de Voleibol - Arena	1
10	Cancha de Voleibol - Arena	1

32) Consulta 32: Listar todos los estados de reserva y las reservas correspondientes

Descripción: Esta consulta muestra todos los estados de las reservas y las reservas correspondientes.

Consulta SQL:

```sql
#consulta 32 Listar todos los estados de reserva y las reservas correspondientes:
SELECT
    rs.name AS EstadoReserva,
    r.id AS ReservaID,
    r.time AS HoraReserva,
    r.dateReserve AS FechaReserva
FROM
    reserveStates rs
RIGHT JOIN
    reserves r ON rs.id = r.idState;
```

Explicación:

1. **Tablas involucradas**:
    - `reserveStates` (`rs`): contiene los diferentes estados de las reservas.
    - `reserves` (`r`): almacena información sobre las reservas.
2. **Cláusula `RIGHT JOIN`**:
    - Se utiliza para incluir todas las reservas, incluso aquellas que pueden no tener un estado asignado.
3. **Selección de columnas**:
    - Se seleccionan el nombre del estado de reserva, el ID de la reserva, la hora de la reserva y la fecha de la reserva.

Resultado: Devuelve una lista de todos los estados de reserva y las reservas correspondientes, incluso si algunas reservas no tienen un estado asignado.

EstadoReserva	ReservaID	HoraReserva	FechaReserva
Confirmado	1	10:00:00	2024-11-01
Confirmado	2	11:00:00	2024-11-01
Pendiente	3	12:00:00	2024-11-01
Confirmado	4	13:00:00	2024-11-01
Confirmado	5	14:00:00	2024-11-01
Pendiente	6	15:00:00	2024-11-01
Confirmado	7	16:00:00	2024-11-01
Confirmado	8	17:00:00	2024-11-01
Pendiente	9	18:00:00	2024-11-01
Confirmado	10	19:00:00	2024-11-02

33) Consulta 33: Mostrar todas las categorías de productos y los productos correspondientes

Descripción: Esta consulta obtiene una lista de todas las categorías de productos junto con los productos que pertenecen a cada categoría.

Consulta SQL:

```sql
#consulta 33 Mostrar todas las categorías de productos y los productos correspondientes:
SELECT
    c.id AS category_id,
    c.name AS category_name,
    p.name AS product_name
FROM
    productCategories c
RIGHT JOIN
    products p ON c.id = p.idProductCategory;
```

Explicación:

1. **Tablas involucradas**:
   - `productCategories` (`c`): almacena información sobre las categorías de productos.
   - `products` (`p`): contiene información sobre los productos.
2. **Cláusula `RIGHT JOIN`**:
   - Se utiliza para incluir todos los productos, incluso si no tienen una categoría asignada.
3. **Selección de columnas**:
   - Se seleccionan el ID de la categoría, el nombre de la categoría y el nombre del producto.

Resultado: Devuelve una lista de todas las categorías de productos junto con los productos correspondientes, incluso si algunos productos no están asignados a ninguna categoría.

	category_id	category_name	product_name
▶	1	Cancha de Fútbol 11	Cancha de Fútbol 11 - Estadio
	2	Cancha de Fútbol 7	Cancha de Fútbol 7 - Polideportivo
	3	Cancha de Fútbol 6	Cancha de Fútbol 6 - Complejo Deportivo
	4	Cancha de Fútbol 5	Cancha de Fútbol 5 - Mini Estadio
	5	Cancha de Voleibol	Cancha de Voleibol - Arena

34) Consulta 34: Listar todos los métodos de pago y los pagos realizados con esos métodos

Descripción: Esta consulta obtiene una lista de todos los métodos de pago junto con los pagos realizados utilizando esos métodos.

Consulta SQL:

```sql
#consulta 34 Listar todos los métodos de pago y los pagos realizados con esos métodos:
SELECT
    pm.id AS method_id,
    pm.name AS payment_method,
    pay.totalPay AS total_paid
FROM
    paymentMethods pm
RIGHT JOIN
    paymentTransactionInformation pti ON pm.id = pti.idPayMethods
RIGHT JOIN
    payment pay ON pti.id = pay.idTransaction;
```

Explicación:

1. **Tablas involucradas**:
   - `paymentMethods` (`pm`): almacena información sobre los métodos de pago.
   - `paymentTransactionInformation` (`pti`): vincula métodos de pago con transacciones.
   - `payment` (`pay`): contiene información sobre los pagos realizados.
2. **Cláusulas `RIGHT JOIN`**:
   - Se utilizan para incluir todos los métodos de pago, incluso aquellos que pueden no tener pagos asociados.
3. **Selección de columnas**:
   - Se seleccionan el ID del método de pago, el nombre del método y el total pagado.

Resultado: Devuelve una lista de métodos de pago y el total pagado con cada uno, incluyendo métodos que no han sido utilizados.

method_id	payment_method	total_paid
1	Tarjeta de Crédito	50.00
2	Tarjeta de Débito	30.00
3	Transferencia Bancaria	70.00
4	Efectivo	100.00
5	PayPal	60.00
1	Tarjeta de Crédito	90.00
2	Tarjeta de Débito	45.00
3	Transferencia Bancaria	25.00
4	Efectivo	55.00

35) Consulta 35: Obtener descuentos aplicados a pagos

Descripción: Esta consulta muestra los descuentos que se han aplicado a los pagos realizados.

Consulta SQL:

```sql
#consulta 35 Obtener descuentos aplicados a pagos:
SELECT
    pay.id AS payment_id,
    pay.totalPay AS total_paid,
    d.total AS discount_amount
FROM
    payment pay
RIGHT JOIN
    discount d ON pay.idDiscount = d.id;
```

Explicación:

1. **Tablas involucradas**:
   - `payment` (`pay`): almacena información sobre los pagos realizados.
   - `discount` (`d`): contiene información sobre los descuentos aplicados.
2. **Cláusula `RIGHT JOIN`**:
   - Se utiliza para incluir todos los pagos, incluso aquellos que no tienen descuentos asociados.
3. **Selección de columnas**:
   - Se seleccionan el ID del pago, el total pagado y el monto del descuento aplicado.

Resultado: Devuelve una lista de pagos junto con los descuentos aplicados, incluyendo pagos que no tienen descuentos.

payment_id	total_paid	discount_amount
3	70.00	10
5	60.00	20
6	90.00	30

36) Consulta 36: Listar todas las posibles combinaciones de usuarios y roles

Descripción: Esta consulta obtiene todas las combinaciones posibles de usuarios y roles en el sistema.

Consulta SQL:

```sql
#consulta 36  Listar todos los posibles combinaciones de usuarios y roles:
SELECT
    u.id AS user_id,
    p.name AS user_name,
    r.id AS role_id,
    r.name AS role_name
FROM
    users u
INNER JOIN
    people p ON u.id = p.idUser
CROSS JOIN
    roles r;
```

Explicación:

1. **Tablas involucradas**:
   - `users` (`u`): almacena información sobre los usuarios.
   - `people` (`p`): contiene información sobre las personas asociadas a los usuarios.
   - `roles` (`r`): almacena información sobre los roles disponibles.
2. **Cláusulas `INNER JOIN` y `CROSS JOIN`**:
   - `INNER JOIN` se utiliza para unir usuarios con sus personas correspondientes.
   - `CROSS JOIN` se utiliza para generar todas las combinaciones posibles de usuarios y roles.
3. **Selección de columnas**:
   - Se seleccionan el ID del usuario, el nombre del usuario, el ID del rol y el nombre del rol.

Resultado: Devuelve todas las combinaciones posibles de usuarios y roles, mostrando cada usuario con cada rol.

user_id	user_name	role_id	role_name
1	Luis	2	Employees
1	Luis	1	Customer
1	Luis	3	Admin
2	Sofía	2	Employees
2	Sofía	1	Customer
2	Sofía	3	Admin
3	Pablo	2	Employees
3	Pablo	1	Customer
3	Pablo	3	Admin
4	Verónica	2	Employees
4	Verónica	1	Customer
4	Verónica	3	Admin
5	Diego	2	Employees
5	Diego	1	Customer
5	Diego	3	Admin

user_id	user_name	role_id	role_name
6	Laura	2	Employees
6	Laura	1	Customer
6	Laura	3	Admin
7	Felipe	2	Employees
7	Felipe	1	Customer
7	Felipe	3	Admin
8	Paola	2	Employees
8	Paola	1	Customer
8	Paola	3	Admin
9	Andrés	2	Employees
9	Andrés	1	Customer
9	Andrés	3	Admin
10	Carolina	2	Employees
10	Carolina	1	Customer
10	Carolina	3	Admin

37) Consulta 37: Obtener todas las combinaciones de tipos de documentos y personas

Descripción: Esta consulta muestra todas las combinaciones posibles entre tipos de documentos y personas.

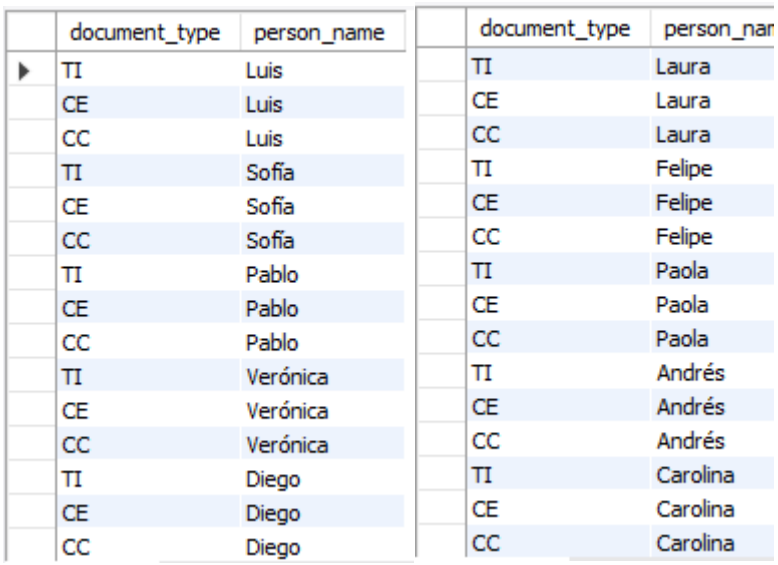
Consulta SQL:

```sql
#consulta 37  Obtener todas las combinaciones de tipos de documentos y personas:
SELECT
    td.name AS document_type,
    p.name AS person_name
FROM
    type_document td
CROSS JOIN
    people p;
```

Explicación:

1. **Tablas involucradas**:
   - `type_document` (`td`): contiene información sobre los tipos de documentos.
   - `people` (`p`): almacena información sobre las personas.
2. **Cláusula `CROSS JOIN`**:
   - Se utiliza para generar todas las combinaciones posibles de tipos de documentos y personas.
3. **Selección de columnas**:
   - Se seleccionan el tipo de documento y el nombre de la persona.

Resultado: Devuelve todas las combinaciones posibles entre tipos de documentos y personas.



document_type	person_name
TI	Luis
CE	Luis
CC	Luis
TI	Sofía
CE	Sofía
CC	Sofía
TI	Pablo
CE	Pablo
CC	Pablo
TI	Verónica
CE	Verónica
CC	Verónica
TI	Diego
CE	Diego
CC	Diego

document_type	person_nar
TI	Laura
CE	Laura
CC	Laura
TI	Felipe
CE	Felipe
CC	Felipe
TI	Paola
CE	Paola
CC	Paola
TI	Andrés
CE	Andrés
CC	Andrés
TI	Carolina
CE	Carolina
CC	Carolina

38) Consulta 38: Mostrar todas las combinaciones de productos y categorías

Descripción: Esta consulta obtiene todas las combinaciones posibles entre productos y categorías de productos.

Consulta SQL:

```sql
#consulta 38 Mostrar todas las combinaciones de productos y categorías:
SELECT
    p.name AS product_name,
    c.name AS category_name
FROM
    products p
CROSS JOIN
    productCategories c;
```

Explicación:

1. **Tablas involucradas**:
   - `products` (`p`): almacena información sobre los productos.
   - `productCategories` (`c`): contiene información sobre las categorías de productos.
2. **Cláusula `CROSS JOIN`**:
   - Se utiliza para generar todas las combinaciones posibles de productos y categorías.
3. **Selección de columnas**:
   - Se seleccionan el nombre del producto y el nombre de la categoría.

Resultado: Devuelve todas las combinaciones posibles entre productos y categorías.

product_name	category_name
Cancha de Voleibol - Arena	Cancha de Fútbol 11
Cancha de Fútbol 5 - Mini Estadio	Cancha de Fútbol 11
Cancha de Fútbol 6 - Complejo Deportivo	Cancha de Fútbol 11
Cancha de Fútbol 7 - Polideportivo	Cancha de Fútbol 11
Cancha de Fútbol 11 - Estadio	Cancha de Fútbol 11
Cancha de Voleibol - Arena	Cancha de Fútbol 5
Cancha de Fútbol 5 - Mini Estadio	Cancha de Fútbol 5
Cancha de Fútbol 6 - Complejo Deportivo	Cancha de Fútbol 5
Cancha de Fútbol 7 - Polideportivo	Cancha de Fútbol 5
Cancha de Fútbol 11 - Estadio	Cancha de Fútbol 5
Cancha de Voleibol - Arena	Cancha de Fútbol 6
Cancha de Fútbol 5 - Mini Estadio	Cancha de Fútbol 6
Cancha de Fútbol 6 - Complejo Deportivo	Cancha de Fútbol 6
Cancha de Fútbol 7 - Polideportivo	Cancha de Fútbol 6
Cancha de Fútbol 11 - Estadio	Cancha de Fútbol 6

product_name	category_name
Cancha de Fútbol 5 - Mini Estadio	Cancha de Fútbol 6
Cancha de Fútbol 6 - Complejo Deportivo	Cancha de Fútbol 6
Cancha de Fútbol 7 - Polideportivo	Cancha de Fútbol 6
Cancha de Fútbol 11 - Estadio	Cancha de Fútbol 6
Cancha de Voleibol - Arena	Cancha de Fútbol 7
Cancha de Fútbol 5 - Mini Estadio	Cancha de Fútbol 7
Cancha de Fútbol 6 - Complejo Deportivo	Cancha de Fútbol 7
Cancha de Fútbol 7 - Polideportivo	Cancha de Fútbol 7
Cancha de Fútbol 11 - Estadio	Cancha de Fútbol 7
Cancha de Voleibol - Arena	Cancha de Voleibol
Cancha de Fútbol 5 - Mini Estadio	Cancha de Voleibol
Cancha de Fútbol 6 - Complejo Deportivo	Cancha de Voleibol
Cancha de Fútbol 7 - Polideportivo	Cancha de Voleibol
Cancha de Fútbol 11 - Estadio	Cancha de Voleibol

39) Consulta 39: Listar todos los métodos de pago y las reservas

Descripción: Esta consulta obtiene una lista de todos los métodos de pago y sus respectivas reservas, generando todas las combinaciones posibles entre ambos.

Consulta SQL:

```
#consulta 39 Listar todos los métodos de pago y las reservas:
SELECT
    pm.name AS payment_method,
    r.dateReserve
FROM
    paymentMethods pm
CROSS JOIN
    reserves r;
```

Explicación:

1. **Tablas involucradas**:
   - `paymentMethods` (`pm`): almacena información sobre los métodos de pago.
   - `reserves` (`r`): contiene información sobre las reservas realizadas.
2. **Cláusula `CROSS JOIN`**:
   - Se utiliza para generar todas las combinaciones posibles de métodos de pago y reservas.
3. **Selección de columnas**:
   - Se selecciona el nombre del método de pago y la fecha de reserva.

Resultado: Devuelve una lista de combinaciones entre métodos de pago y fechas de reserva.

payment_method	dateReserve
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01

payment_method	dateReserve
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01

payment_method	dateReserve
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01
PayPal	2024-11-01
Efectivo	2024-11-01
Transferencia Bancaria	2024-11-01
Tarjeta de Débito	2024-11-01
Tarjeta de Crédito	2024-11-01
PayPal	2024-11-02
Efectivo	2024-11-02
Transferencia Bancaria	2024-11-02
Tarjeta de Débito	2024-11-02
Tarjeta de Crédito	2024-11-02

40) Consulta 40: Obtener todas las combinaciones de descuentos y pagos

Descripción: Esta consulta muestra todas las combinaciones posibles entre descuentos aplicados y pagos realizados.

Consulta SQL:

```
#consulta 40 Obtener todas las combinaciones de descuentos y pagos:
SELECT
    d.total AS discount_total,
    pay.totalPay
FROM
    discount d
CROSS JOIN
    payment pay;
```

Explicación:

1. **Tablas involucradas**:
   - `discount` (`d`): contiene información sobre los descuentos disponibles.
   - `payment` (`pay`): almacena información sobre los pagos realizados.
2. **Cláusula `CROSS JOIN`**:
   - Se utiliza para generar todas las combinaciones posibles de descuentos y pagos.
3. **Selección de columnas**:
   - Se seleccionan el total del descuento y el total pagado.

Resultado: Devuelve una lista de combinaciones entre descuentos y montos de pagos.

discount_total	totalPay
30	50.00
20	50.00
10	50.00
30	30.00
20	30.00
10	30.00
30	70.00
20	70.00
10	70.00
30	100.00
20	100.00
10	100.00
30	60.00
20	60.00
10	60.00

discount_total	totalPay
20	60.00
10	60.00
30	90.00
20	90.00
10	90.00
30	45.00
20	45.00
10	45.00
30	25.00
20	25.00
10	25.00
30	55.00
20	55.00
10	55.00

SUBCONSULTAS

41) Subconsulta 1: Obtener los clientes y su nombre con promedio mayor a 80 en alquilar las canchas

Descripción: Esta subconsulta obtiene los nombres de los clientes que tienen un promedio de pago mayor a 80 por las reservas de canchas.

Consulta SQL:

```sql
#subconsultas 1 Obtener los clientes y su nombre con promedio mayor a 80 en alquilar las canchas
SELECT
    people.name AS client_name,
    ROUND(
        (SELECT AVG(payment.totalPay)
         FROM payment
         INNER JOIN paymentTransactionInformation pti ON payment.idTransaction = pti.id
         INNER JOIN reserves r ON pti.idCustomers = r.idUser
         WHERE r.idUser = people.id
        ), 2) AS average_paid
FROM
    people
INNER JOIN users ON people.idUser = users.id
INNER JOIN usersRoles ON users.id = usersRoles.idUser
INNER JOIN roles ON usersRoles.idRole = roles.id
WHERE
    roles.id = 1  -- Filtra por el rol 1 (clientes)
GROUP BY
    people.id, people.name
HAVING
    average_paid > 80;
```

Explicación:

1. **Tablas involucradas**:
   - `people`: almacena información sobre los clientes.
   - `payment`: contiene información sobre los pagos.
   - `paymentTransactionInformation`: vincula pagos con reservas.
   - `reserves`: contiene información sobre las reservas.
   - `users`, `usersRoles`, `roles`: para filtrar clientes.
2. **Subconsulta**:
   - Calcula el promedio de pagos por cliente y filtra aquellos con un promedio mayor a 80.

Resultado: Devuelve una lista de clientes con un promedio de pago mayor a 80.

	client_name	average_paid
▶	Verónica	100.00
	Laura	90.00

42) Subconsulta 2: Clientes con 2 o más reservaciones

Descripción: Esta subconsulta obtiene los nombres de los clientes que tienen 2 o más reservas.

Consulta SQL:

```sql
# 2 subconsulta where con clientes con 2 o más reservaciones
SELECT
    people.name AS client_name
FROM
    people
WHERE
    (SELECT COUNT(*)
     FROM reserves
     WHERE reserves.idUser = people.id) >= 2;
```

Explicación:

1. **Tablas involucradas**:
   - `people`: almacena información sobre los clientes.
   - `reserves`: contiene información sobre las reservas.
2. **Subconsulta**:
   - Cuenta las reservas para cada cliente y filtra aquellos con 2 o más.

Resultado: Devuelve una lista de clientes con al menos 2 reservas.

	client_name
▶	Luis

43) Subconsulta 3: Productos usados más de una vez

Descripción: Esta subconsulta obtiene los nombres de los productos que han sido reservados más de una vez.

Consulta SQL:

```sql
# 3 subconsulta where dónde un producto sea usado por más 1 veces
SELECT
    products.name AS product_name
FROM
    products
WHERE
    (SELECT SUM(reserveProducts.quantity)
     FROM reserveProducts
     WHERE reserveProducts.idProduct = products.id) > 1;
```

Explicación:

1. **Tablas involucradas**:
   - `products`: almacena información sobre los productos.
   - `reserveProducts`: vincula productos con reservas y sus cantidades.
2. **Subconsulta**:
   - Suma las cantidades de cada producto reservado y filtra aquellos que superan 1.

Resultado: Devuelve una lista de productos que han sido reservados más de una vez.

product_name
Cancha de Fútbol 11 - Estadio
Cancha de Fútbol 7 - Polideportivo
Cancha de Fútbol 6 - Complejo Deportivo
Cancha de Fútbol 5 - Mini Estadio
Cancha de Voleibol - Arena

44) Subconsulta 4: Cantidad de reservas agrupadas por su estado

Descripción: Esta subconsulta obtiene la cantidad de reservas agrupadas por su estado.

Consulta SQL:

```sql
#4 subconsulta From Obtener la cantidad de reservas agrupadas por su estado.
SELECT
    state_data.state_name,
    COUNT(state_data.reserve_id) AS total_reserves
FROM
    (SELECT
        reserves.id AS reserve_id,
        reserveStates.name AS state_name
     FROM
        reserves
     INNER JOIN reserveStates ON reserves.idState = reserveStates.id) AS state_data
GROUP BY
    state_data.state_name;
```

Explicación:

1. **Tablas involucradas**:
    - `reserves`: contiene información sobre las reservas.
    - `reserveStates`: almacena información sobre los estados de las reservas.
2. **Subconsulta**:
    - Extrae el ID de la reserva y el nombre del estado y luego cuenta las reservas por estado.

Resultado: Devuelve la cantidad de reservas agrupadas por su estado.

state_name	total_reserves
Confirmado	7
Pendiente	3

45) Subconsulta 5: Obtener la cantidad de reservas realizadas por cada cliente, agrupadas por el estado de la reserva

Descripción: Esta subconsulta obtiene la cantidad de reservas realizadas por cada cliente, organizadas por el estado de cada reserva.

Consulta SQL:

```sql
#5 subconsulta From,Obtener la cantidad de reservas realizadas por cada cliente, agrupadas por el estado de la reserva.
SELECT
    customer_data.client_name,
    reserveStates.name AS state_name,
    COUNT(customer_data.reserve_id) AS total_reserves
FROM
    (SELECT
        people.name AS client_name,
        reserves.id AS reserve_id,
        reserves.idState
     FROM
        people
     INNER JOIN reserves ON people.id = reserves.idUser) AS customer_data
INNER JOIN reserveStates ON customer_data.idState = reserveStates.id
GROUP BY
    customer_data.client_name, reserveStates.name;
```

Explicación:

1. **Tablas involucradas**:
   - `people`: contiene información sobre los clientes.
   - `reserves`: almacena información sobre las reservas.
   - `reserveStates`: define los estados de las reservas.
2. **Subconsulta**:
   - Selecciona el nombre del cliente, el ID de la reserva y el estado de la reserva.
3. **Cláusula `GROUP BY`**:
   - Agrupa los resultados por cliente y estado de reserva, contando el total de reservas.

Resultado: Devuelve el número de reservas por cliente, segmentadas por el estado de cada reserva.

client_name	state_name	total_reserves
Luis	Confirmado	2
Sofía	Confirmado	1
Verónica	Confirmado	1
Diego	Confirmado	1
Paola	Confirmado	1
Andrés	Confirmado	1
Pablo	Pendiente	1
Laura	Pendiente	1
Carolina	Pendiente	1

46) Subconsulta 6: Obtener información sobre las reservas y el descuento aplicado

Descripción: Esta subconsulta extrae información sobre las reservas y los descuentos aplicados a cada una de ellas.

Consulta SQL:

```sql
#6 subconsulta From Obtener información sobre las reservas y el descuento aplicado.
SELECT
    reserve_data.reserve_id,
    reserve_data.dateReserve,
    reserve_data.time,
    reserve_data.state_name,
    reserve_data.discount_amount
FROM
    (SELECT
        r.id AS reserve_id,
        r.dateReserve,
        r.time,
        rs.name AS state_name,
        d.total AS discount_amount
     FROM
        reserves r
     INNER JOIN reserveStates rs ON r.idState = rs.id
     LEFT JOIN paymentTransactionInformation pti ON r.id = pti.idCustomers
     LEFT JOIN payment p ON pti.id = p.idTransaction
     LEFT JOIN discount d ON p.idDiscount = d.id
    ) AS reserve_data;
```

Explicación:

1. **Tablas involucradas**:
    - `reserves`: información sobre las reservas.
    - `reserveStates`: para obtener el estado de las reservas.
    - `paymentTransactionInformation`: vincula reservas con pagos.
    - `payment`: contiene información de los pagos.
    - `discount`: información sobre los descuentos aplicados.
2. **Subconsulta**:
    - Combina todas las tablas para obtener el ID de la reserva, la fecha, la hora, el estado y el descuento.

Resultado: Devuelve una lista de reservas con su información y el descuento aplicado.

	reserve_id	dateReserve	time	state_name	discount_amount
▶	1	2024-11-01	10:00:00	Confirmado	NULL
	2	2024-11-01	11:00:00	Confirmado	NULL
	4	2024-11-01	13:00:00	Confirmado	NULL
	5	2024-11-01	14:00:00	Confirmado	20
	7	2024-11-01	16:00:00	Confirmado	NULL
	8	2024-11-01	17:00:00	Confirmado	NULL
	10	2024-11-02	19:00:00	Confirmado	NULL
	3	2024-11-01	12:00:00	Pendiente	10
	6	2024-11-01	15:00:00	Pendiente	30
	9	2024-11-01	18:00:00	Pendiente	NULL

47) Subconsulta 7: Obtener información sobre las reservas confirmadas y el descuento aplicado

Descripción: Esta subconsulta obtiene información sobre reservas confirmadas y los descuentos aplicados a cada una de ellas.

Consulta SQL:

```sql
#7 subconsulta Select Obtener información sobre las reservas confirmadas y el descuento aplicado.
SELECT
    r.id AS reserve_id,
    r.dateReserve,
    r.time,
    rs.name AS state_name,
    (SELECT d.total
     FROM payment p
     LEFT JOIN discount d ON p.idDiscount = d.id
     WHERE p.idTransaction IN
         (SELECT pti.id
          FROM paymentTransactionInformation pti
          WHERE pti.idCustomers = r.idUser)
       AND p.idTransaction IN (SELECT pti.id FROM paymentTransactionInformation pti
            WHERE pti.idCustomers = r.idUser)
     ORDER BY p.createdAt DESC
     LIMIT 1) AS discount_amount
FROM
    reserves r
INNER JOIN reserveStates rs ON r.idState = rs.id
WHERE r.idState = 2;
```

Explicación:

1. **Tablas involucradas**:
    - `reserves`: información sobre las reservas.
    - `reserveStates`: para obtener el estado de la reserva.
    - `payment`: información sobre los pagos.
    - `discount`: información sobre los descuentos aplicados.
2. **Subconsulta**:
    - Calcula el descuento más reciente aplicado a las reservas confirmadas (estado 2).

Resultado: Devuelve información de reservas confirmadas y el descuento aplicado más reciente.

reserve_id	dateReserve	time	state_name	discount_amount
1	2024-11-01	10:00:00	Confirmado	NULL
2	2024-11-01	11:00:00	Confirmado	NULL
4	2024-11-01	13:00:00	Confirmado	NULL
5	2024-11-01	14:00:00	Confirmado	20
7	2024-11-01	16:00:00	Confirmado	NULL
8	2024-11-01	17:00:00	Confirmado	NULL
10	2024-11-02	19:00:00	Confirmado	NULL

48) Subconsulta 8: Obtener el total de reservas realizadas por cada usuario

Descripción: Esta subconsulta obtiene el total de reservas realizadas por cada usuario.

Consulta SQL:

```sql
#8 subconsulta Selec, Obtiene el total de reservas realizadas por cada usuario.
SELECT
    users.id AS user_id,
    people.name,
    (SELECT COUNT(*)
     FROM reserves
     WHERE idUser = users.id) AS total_reservations
FROM
    users
INNER JOIN people ON users.id = people.idUser;
```

Explicación:

1. **Tablas involucradas**:
    - `users`: información sobre los usuarios.
    - `people`: información sobre las personas, incluyendo los clientes.
2. **Subconsulta**:
    - Cuenta el total de reservas realizadas por cada usuario.

Resultado: Devuelve el total de reservas por cada usuario junto con su nombre.

user_id	name	total_reservations
9	Andrés	1
11	Camilo	0
10	Carolina	1
5	Diego	1
7	Felipe	0
6	Laura	1
1	Luis	2
3	Pablo	1
8	Paola	1
2	Sofía	1
4	Verónica	1

49) Subconsulta 9: Obtener los productos reservados con sus precios originales y el precio final, con descuentos aplicables

Descripción: Esta subconsulta obtiene información sobre los productos reservados, sus precios originales y el precio final después de aplicar descuentos, si corresponde.

Consulta SQL:

```sql
#9 subconsulta select. los productos reservados con sus precios originales y el precio final,
# siempre que haya un descuento aplicable
SELECT
    reserves.id AS reserve_id,
    products.name AS product_name,
    products.price AS original_price,
    (products.price - IFNULL(SUM(discount.total), 0)) AS final_price
FROM
    reserves
INNER JOIN reserveProducts ON reserves.id = reserveProducts.idReserve
INNER JOIN products ON reserveProducts.idProduct = products.id
LEFT JOIN payment ON payment.idTransaction = reserves.id
LEFT JOIN discount ON payment.idDiscount = discount.id
WHERE
    reserves.idState IN (1, 2)
GROUP BY
    reserves.id, products.name, products.price
ORDER BY
    reserves.id;
```

Explicación:

1. **Tablas involucradas**:
    - `reserves`: información sobre las reservas.
    - `reserveProducts`: vincula reservas con productos.

- `products`: información sobre los productos.
- `payment`: información sobre los pagos.
- `discount`: información sobre los descuentos.

2. **Cláusula `LEFT JOIN`**:
   - Permite incluir información de descuentos incluso si no se aplican.
3. **Cálculo del precio final**:
   - Resta el descuento total del precio original del producto.

Resultado: Devuelve una lista de productos reservados, su precio original y el precio final después de aplicar descuentos, si los hay.

reserve_id	product_name	original_price	final_price
1	Cancha de Fútbol 11 - Estadio	1000	1000
2	Cancha de Fútbol 7 - Polideportivo	800	800
3	Cancha de Fútbol 6 - Complejo Deportivo	600	590
4	Cancha de Fútbol 5 - Mini Estadio	500	500
5	Cancha de Voleibol - Arena	300	280
6	Cancha de Fútbol 11 - Estadio	1000	970
7	Cancha de Fútbol 7 - Polideportivo	800	800
8	Cancha de Fútbol 6 - Complejo Deportivo	600	600
9	Cancha de Fútbol 5 - Mini Estadio	500	500
10	Cancha de Voleibol - Arena	300	300

50) Subconsulta 10: Obtener personas que tienen reservas en un estado específico

Descripción: Esta subconsulta obtiene los nombres y apellidos de las personas que tienen reservas en un estado específico (en este caso, el estado con ID 2: Confirmado).

Consulta SQL:

```sql
#10 subconsulta con IN Obtener personas que tienen reservas en un estado específico
SELECT p.name, p.lastName
FROM people p
WHERE p.id IN (
    SELECT r.idUser
    FROM reserves r
    WHERE r.idState IN (2)
);
```

Explicación:

1. **Tablas involucradas**:
   - `people`: contiene la información de las personas.
   - `reserves`: información sobre las reservas y su estado.
2. **Subconsulta**:

- Selecciona los IDs de los usuarios que tienen reservas en el estado especificado.

Resultado: Devuelve los nombres y apellidos de las personas con reservas en el estado 2.

	name	lastName
▶	Luis	Moreno
	Sofía	Herrera
	Verónica	Castro
	Diego	Ramírez
	Paola	Mendoza
	Andrés	Torres

51) Subconsulta 11: Listar reservas que tienen pagos asociados

Descripción: Esta subconsulta lista las reservas que tienen pagos asociados, mostrando la información del usuario y el estado de la reserva.

Consulta SQL:

```sql
#11 subconsulta con exists. Listar reservas que tienen pagos asociados
SELECT
    r.id AS reserve_id,
    r.dateReserve,
    p.name AS user_name,
    rs.name AS reserve_state,
    SUM(pmt.totalPay) AS total_paid
FROM
    reserves r
INNER JOIN
    people p ON r.idUser = p.idUser
INNER JOIN
    reserveStates rs ON r.idState = rs.id
LEFT JOIN
    paymentTransactionInformation pti ON pti.idCustomers = r.idUser
LEFT JOIN
    payment pmt ON pmt.idTransaction = pti.id
WHERE
    EXISTS (
        SELECT 1
        FROM payment pmt
        WHERE pmt.idTransaction = pti.id AND pmt.totalPay > 0
    )
GROUP BY r.id, r.dateReserve, p.name, rs.name
ORDER BY r.dateReserve DESC;
```

Explicación:

1. **Tablas involucradas**:
   - `reserves`: información sobre las reservas.
   - `people`: información sobre los usuarios.
   - `reserveStates`: para obtener el estado de cada reserva.
   - `paymentTransactionInformation`: vincula reservas con transacciones de pago.
   - `payment`: información sobre los pagos realizados.
2. **Condición `EXISTS`**:
   - Verifica si existen pagos asociados a la transacción.

Resultado: Devuelve una lista de reservas con información sobre el usuario, el estado de la reserva y el total pagado.

reserve_id	dateReserve	user_name	reserve_state	total_paid
10	2024-11-02	Luis	Confirmado	50.00
1	2024-11-01	Luis	Confirmado	50.00
2	2024-11-01	Sofía	Confirmado	30.00
3	2024-11-01	Pablo	Pendiente	70.00
4	2024-11-01	Verónica	Confirmado	100.00
5	2024-11-01	Diego	Confirmado	60.00
6	2024-11-01	Laura	Pendiente	90.00
7	2024-11-01	Paola	Confirmado	45.00
8	2024-11-01	Andrés	Confirmado	25.00
9	2024-11-01	Carolina	Pendiente	55.00

52) Subconsulta 12: Obtener Personas con un Documento Mayor que Cualquier Documento de un Usuario Específico

Descripción: Esta subconsulta obtiene los nombres y documentos de las personas cuyo documento es mayor que el de cualquier documento asociado a un usuario específico (en este caso, el usuario con `idUser` igual a 2).

Consulta SQL:

```sql
#12 Obtener Personas con un Documento Mayor que Cualquier Documento de un Usuario Específico
SELECT name, document
FROM people
WHERE document > ANY (
    SELECT document
    FROM people
    WHERE idUser = 2
);
```

Explicación:

1. **Tablas involucradas**:
   - `people`: contiene la información de las personas, incluyendo sus nombres y documentos.
2. **Subconsulta**:
   - La subconsulta selecciona los documentos de todas las personas que tienen `idUser` igual a 2. Esto puede ser útil para identificar todos los documentos asociados a un usuario específico.
3. **Condición `ANY`**:
   - El operador `ANY` compara el documento de cada persona en la tabla `people` con todos los documentos devueltos por la subconsulta. Se seleccionarán aquellas personas cuyo documento sea mayor que **cualquiera** de los documentos obtenidos.

Resultado: La consulta devuelve los nombres y documentos de las personas cuyo documento es mayor que cualquier documento del usuario con `idUser` igual a 2.

	name	document
▶	Diego	98765432

53) Subconsulta 13: Productos con menor precio que la cancha de fútbol 7

Descripción: Esta subconsulta lista los nombres de los productos cuyo precio es menor que el precio de la cancha de fútbol 7 (ID 2).

Consulta SQL:

```sql
#13 subconsulta Any. Productos con menor precio que cancha de futbol 7
SELECT name
FROM products
WHERE price < ALL (
    SELECT price
    FROM products
    WHERE id = 2
);
```

Explicación:

1. **Tablas involucradas**:
   - `products`: información sobre los productos.
2. **Subconsulta**:
   - Selecciona el precio de la cancha de fútbol 7.

Resultado: Devuelve los nombres de los productos cuyo precio es inferior al precio de la cancha de fútbol 7.

name
Cancha de Fútbol 6 - Complejo Deportivo
Cancha de Fútbol 5 - Mini Estadio
Cancha de Voleibol - Arena

54) Subconsulta 14: Listar reservas de usuarios que han realizado pagos mayores a 70

Descripción: Esta subconsulta obtiene las reservas realizadas por usuarios que han realizado pagos mayores a 70.

Consulta SQL:

```sql
#14 subconsulta Any. Listar reservas de usuarios que han realizado pagos mayores a un monto  70
SELECT
    r.id AS reserve_id,
    r.dateReserve,
    p.name AS user_name,
    rs.name AS reserve_state
FROM
    reserves r
INNER JOIN
    people p ON r.idUser = p.idUser
INNER JOIN
    reserveStates rs ON r.idState = rs.id
WHERE
    r.idUser = ANY (
        SELECT
            pti.idCustomers
        FROM
            paymentTransactionInformation pti
        INNER JOIN
            payment pmt ON pti.id = pmt.idTransaction
        WHERE
            pmt.totalPay > 70
    )
ORDER BY
    r.dateReserve DESC;
```

Explicación:

1. **Tablas involucradas**:

- `reserves`: información sobre las reservas.
- `people`: información sobre los usuarios.
- `reserveStates`: para obtener el estado de las reservas.
- `paymentTransactionInformation`: vincula reservas con transacciones de pago.
- `payment`: información sobre los pagos realizados.

2. **Condición `ANY`**:
    - Selecciona los usuarios que han realizado pagos superiores a 70.

Resultado: Devuelve las reservas de usuarios que han realizado pagos mayores a 70, incluyendo información sobre la reserva y el estado

	reserve_id	dateReserve	user_name	reserve_state
▶	4	2024-11-01	Verónica	Confirmado
	6	2024-11-01	Laura	Pendiente