```sql
-- Relación cuentas con usuario
-- Obtener el saldo promedio de todas las cuentas de un usuario específico.
--
SELECT AVG(CAST(saldo AS numeric)) AS SaldoPromedio
FROM cuentas
WHERE cedula_propietario = '12345';

```

Data Output | Messages | Notifications

	saldopromedio numeric
1	250.0000000000000000

```sql
-- Obtener el número total de cuentas de cada tipo de cuenta.
-- Escribir la sentencia SQL utilizando la función de agregación COUNT para contar el número de cuentas
-- Utilizar la cláusula GROUP BY para agrupar las cuentas por tipo de cuenta.
SELECT tipo_cuenta, COUNT(*) AS NumeroDeCuentas
FROM usuario
GROUP BY tipo_cuenta;

```

Data Output | Messages | Notifications

	tipo_cuenta character varying (20)	numerodecuentas bigint
1	Ahorros	2
2	Corriente	3

```sql
-- Obtener el monto total de compras realizadas por cada cliente.
-- Escribir la sentencia SQL utilizando la función de agregación SUM para sumar los montos de las compras.
-- Utilizar la cláusula GROUP BY para agrupar las compras por cliente.
SELECT c.cedula, c.nombre, c.apellido, SUM(co.monto) AS MontoTotal
FROM clientes c
JOIN compras co ON c.cedula = co.cedula
GROUP BY c.cedula, c.nombre, c.apellido;

```

Data Output | Messages | Notifications

	cedula [PK] character	nombre character varying (50)	apellido character varying (50)	montototal numeric
1	1755014238	Juan	Lopez	3450.00

```sql
select * from compras
-- Obtener la cantidad total de compras realizadas en una fecha específica.
-- Escribir la sentencia SQL utilizando la función de agregación COUNT para contar el número de compras.
-- Utilizar la cláusula WHERE para filtrar las compras por la fecha específica.
SELECT COUNT(*) AS CantidadCompras
FROM compras
WHERE fecha_compra = '2024-07-20';
```

Data Output | Messages | Notifications

	cantidadcompras bigint
1	1

```sql
-- Relación entre estudiantes y profesores
-- Obtener la cantidad total de estudiantes asignados a cada profesor.
-- Escribir la sentencia SQL utilizando la función de agregación COUNT para contar el número de estudiantes.
-- Utilizar la cláusula GROUP BY para agrupar los estudiantes por profesor.
SELECT p.codigo, COUNT(e.cedula) AS NumeroDeEstudiantes
FROM profesores p
JOIN estudiantes e ON p.codigo = e.codigo_profesor
GROUP BY p.codigo;
```

Data Output | Messages | Notifications

	codigo [PK] integer	numerodeestudiantes bigint
1	3	1
2	5	1
3	4	1
4	2	1
5	1	1

```sql
-- Obtener la edad promedio de los estudiantes.
-- Escribir la sentencia SQL utilizando la función ROUND para mostrar la edad_promedio como entero,
-- dentro de ROUND la función de agregación AVG (EXTRACT(YEAR FROM CURRENT_DATE) Extract para obtener
-- la diferencia en años entre la fecha actual y la fecha de nacimiento de cada estudiante seguido de
-- un guion hacer otro EXTRACT que lleva dentro EL AÑO (YEAR) de la fecha de nacimiento de los estudiantes,
-- con ello obtendremos el promedio de las edades de los estudiantes.
SELECT ROUND(AVG(EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM fecha_nacimiento))) AS EdadPromedio
FROM estudiantes;
```

Data Output | Messages | Notifications

	edadpromedio numeric
1	22

```sql
-- Relación entre persona y préstamo
-- Obtener la suma total de los montos de préstamos para cada persona.
-- Escribir la sentencia SQL utilizando la función de agregación SUM para sumar los montos de los préstamos.
-- Utilizar la cláusula GROUP BY para agrupar los préstamos por persona.
SELECT p.cedula, p.nombre, p.apellido, SUM(pr.monto) AS MontoTotalPrestamos
FROM personas p
JOIN prestamo pr ON p.cedula = pr.cedula
GROUP BY p.cedula, p.nombre, p.apellido;
```

Data Output | Messages | Notifications

	cedula [PK] character	nombre character varying (50)	apellido character varying (50)	montototalprestamos money
1	1755014238	Juan	Rodriguez	$1.000,00

```sql
-- Obtener la cantidad total de personas que tienen más de un hijo.
-- Escribir la sentencia SQL utilizando la función de agregación COUNT para contar el número de personas.
-- Utilizar la cláusula WHERE para filtrar las personas que tienen más de un hijo.
SELECT COUNT(*) AS NumeroDePersonas
FROM personas
WHERE numero_hijos > 1;
```

Data Output | Messages | Notifications

	numerodepersonas bigint
1	3

```sql
-- Relación entre Productos y Ventas
-- Obtener el máximo precio de todos los productos.
-- Escribir la sentencia SQL utilizando la función de agregación MAX para obtener el máximo precio de los productos.
SELECT MAX(precio) AS PrecioMaximo
FROM productos;
```

Data Output | Messages | Notifications

	preciomaximo integer
1	3000

```sql
-- Obtener la suma total de la cantidad de productos vendidos.
-- Escribir la sentencia SQL utilizando la función de agregación SUM para sumar la cantidad de productos vendidos.
SELECT SUM(cantidad) AS TotalProductosVendidos
FROM ventas;
```

Data Output | Messages | Notifications

	totalproductosvendidos bigint
1	15

```sql
-- Relación entre Transacciones y Banco
-- Obtener la cantidad total de transacciones de tipo 'C' (crédito).
-- Escribir la sentencia SQL utilizando la función de agregación COUNT para contar el número de transacciones.
-- Utilizar la cláusula WHERE para filtrar las transacciones de tipo 'C' (crédito).
SELECT COUNT(*) AS total_transacciones_credito
FROM transacciones
WHERE tipo = 'C';
```

Data Output | Messages | Notifications

	total_transacciones_credito bigint
1	3

```sql
-- Obtener el promedio de montos de transacciones para cada número de cuenta.
-- Escribir la sentencia SQL selecciona el número de cuenta seguido de una coma utilizar la función
-- ROUND dentro utiliza la función de agregación AVG para calcular el promedio de los montos de las
-- transacciones dentro de la función AVG usa la función CAST para el monto castéalo como decimal y
-- fuera del paréntesis especificar el numero 2 para que nos resulten dos decimales y con ello
-- obtendremos el monto_promedio a través de las transacciones.
-- SELECT ****** ROUND(AVG(CAST(******AS decimal)),2) AS ****** FROM ****** GROUP BY *******;
-- Utilizar la cláusula GROUP BY para agrupar las transacciones por número de cuenta.
SELECT numero_cuenta, ROUND(AVG(CAST(monto AS DECIMAL)), 2) AS MontoPromedio
FROM transacciones
GROUP BY numero_cuenta;
```

Data Output | Messages | Notifications

	numero_cuenta character varying (5)	montopromedio numeric
1	22111	35.00
2	22678	85.00

```sql
-- Relación entre Videojuegos y Plataformas
-- Obtener la cantidad total de plataformas disponibles para cada videojuego.
-- Escribir la sentencia SQL utilizando la función de agregación COUNT para contar el número de plataformas.
-- Utilizar la cláusula GROUP BY para agrupar las plataformas por videojuego usando el código del videojuego.
SELECT v.nombre, COUNT(p.id_plataforma) AS NumeroDePlataformas
FROM videojuegos v
JOIN plataformas p ON v.codigo = p.codigo_videojuego
GROUP BY v.nombre;
```

Data Output | Messages | Notifications

	nombre character varying (100)	numerodeplataformas bigint
1	Juego E	1
2	Juego C	1
3	Juego A	1
4	Juego B	1
5	Juego D	1

```sql
-- Obtener la valoración promedio de todos los videojuegos.
-- Escribir la sentencia SQL utilizando la función de agregación AVG para calcular el
-- promedio de las valoraciones de los videojuegos, Utiliza la función ROUND antes de
-- AVG para redondear a 2 decimales el resultado .
-- Ejemplo: SELECT ROUND(AVG(*******),2) AS ******** FROM ******;
SELECT ROUND(AVG(valoracion), 2) AS ValoracionPromedio
FROM videojuegos;
```

Data Output | Messages | Notifications

	valoracionpromedio numeric
1	8.20

```sql
-- Relación entre registros_entrada y empleado
-- Obtener la cantidad total de registros de entrada realizados por cada empleado.
-- Escribir la sentencia SQL utilizando la función de agregación COUNT para contar el número de registros de entrada.
-- Utilizar la cláusula GROUP BY para agrupar los registros de entrada por empleado.
SELECT e.nombre, COUNT(r.codigo_registro) AS NumeroRegistrosEntrada
FROM empleado e
JOIN registros_entrada r ON e.codigo_empleado = r.codigo_empleado
GROUP BY e.nombre;
```

Data Output | Messages | Notifications

	nombre character varying (25)	numeroregistrosentrada bigint
1	Empleado C	1
2	Empleado B	1
3	Empleado D	1
4	Empleado A	1
5	Empleado E	1

```sql
-- Obtener la fecha mínima y máxima de los registros de entrada.
-- Escribir la sentencia SQL utilizando la función de agregación MIN
-- para obtener la fecha mínima y la función de agregación MAX para obtener
-- la fecha máxima de los registros de entrada.
SELECT MIN(fecha) AS FechaMinima, MAX(fecha) AS FechaMaxima
FROM registros_entrada;
```

Data Output | Messages | Notifications

	fechaminima date	fechamaxima date
1	2023-08-01	2023-09-10