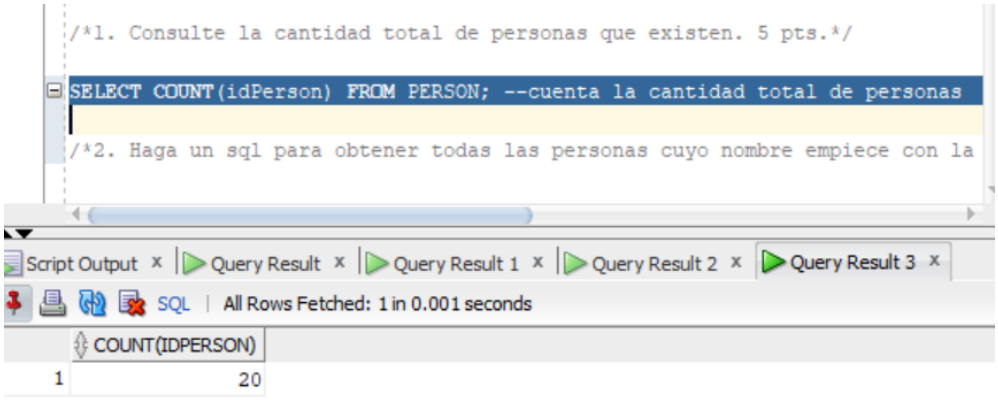
Laboratorio #3: Consultas

1. **Consulte la cantidad total de personas que existen. 5 pts.**



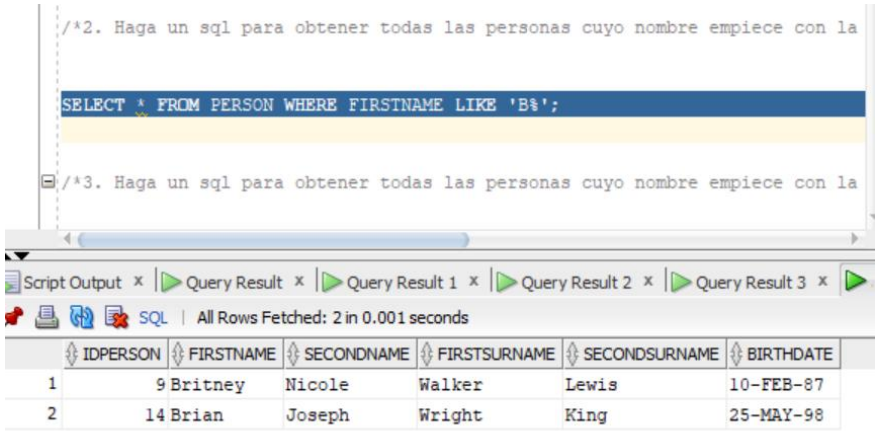
```
/*1. Consulte la cantidad total de personas que existen. 5 pts.*/

SELECT COUNT(idPerson) FROM PERSON; --cuenta la cantidad total de personas

/*2. Haga un sql para obtener todas las personas cuyo nombre empiece con la
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

SQL | All Rows Fetched: 1 in 0.001 seconds

	COUNT(IDPERSON)
1	20

2. Haga un sql para obtener todas las personas cuyo nombre empiece con la letra B. 4 pts.



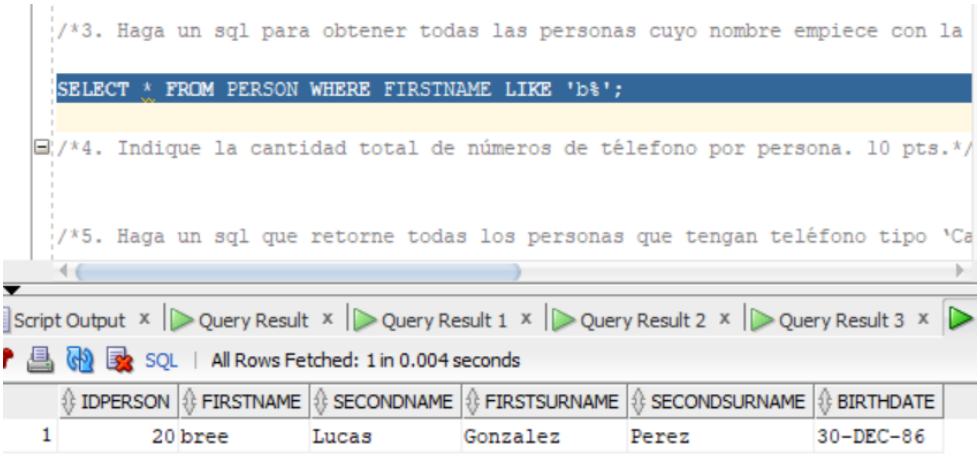
```
/*2. Haga un sql para obtener todas las personas cuyo nombre empiece con la

SELECT * FROM PERSON WHERE FIRSTNAME LIKE 'B%';

/*3. Haga un sql para obtener todas las personas cuyo nombre empiece con la
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

SQL | All Rows Fetched: 2 in 0.001 seconds

	IDPERSON	FIRSTNAME	SECONDNAME	FIRSTSURNAME	SECONDSURNAME	BIRTHDATE
1	9	Britney	Nicole	Walker	Lewis	10-FEB-87
2	14	Brian	Joseph	Wright	King	25-MAY-98

3. **Haga un sql para obtener todas las personas cuyo nombre empiece con la letra b (en minúscula).**



```
/*3. Haga un sql para obtener todas las personas cuyo nombre empiece con la

SELECT * FROM PERSON WHERE FIRSTNAME LIKE 'b%';

/*4. Indique la cantidad total de números de télefono por persona. 10 pts.*/

/*5. Haga un sql que retorne todas los personas que tengan teléfono tipo 'Ca
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

SQL | All Rows Fetched: 1 in 0.004 seconds

	IDPERSON	FIRSTNAME	SECONDNAME	FIRSTSURNAME	SECONDSURNAME	BIRTHDATE
1	20	bree	Lucas	Gonzalez	Perez	30-DEC-86

4. Indique la cantidad total de números de télefono por persona. 10 pts.

```
SELECT p.idPerson, COUNT(px.idPhone) AS totalPhoneNumbers
FROM Person p
LEFT JOIN Phone px ON p.idPerson = px.idPerson
GROUP BY p.idPerson
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Quer

SQL | All Rows Fetched: 20 in 0.015 seconds

	IDPERSON	TOTALPHONENUMBERS
2	13	0
3	11	0
4	21	0
5	14	0
6	20	0
7	2	0
8	4	2
9	5	0
10	17	1
11	8	1
12	7	1
13	3	0
14	18	0
15	10	2
16	19	1
17	12	1
18	15	0
19	16	0
20	9	0

5. Haga un sql que retorne todas los personas que tengan teléfono tipo 'Casa'. 10 pts.

```
/*5. Haga un sql que retorne todas los personas que tengan teléfono tipo 'Casa'. 10 pts.*/

SELECT p.idPerson, p.firstName, pt.phoneType
FROM person p
INNER JOIN phone ph ON p.idPerson = ph.idPerson
INNER JOIN phonetype pt ON ph.idPhoneType = pt.idPhoneType
WHERE pt.phoneType = 'Casa';
```

Query Result | Script Output | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 3 in 0.005 seconds

	IDPERSON	FIRSTNAME	PHONETYPE
1	6	David	Casa
2	8	Christopher	Casa
3	19	Sophia	Casa

6. **Cree una vista que contenga todas las personas que ganan menos de $3000. 10 pts.**

```
/*6. Cree una vista que contenga todas las personas que ganan menos de $3000. 10 pts.*/

CREATE VIEW employeeBadSalary AS
SELECT *
FROM employee
WHERE salary < 3000;

SELECT * FROM employeeBadSalary;

/*7. Cree dos vistas con el top 3 de personas con más salario con base en los 2 siguientes queries
¿Cuál es la diferencia entre cada uno? ver enunciado 20 pts.
ian*/
```

Query Result | Script Output | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 3 in 0.003 seconds

	IDPERSON	SALARY
1	5	2900
2	9	1500
3	14	2500

7. **Cree dos vistas con el top 5 de personas con más salario con base en los 2 siguientes queries. ¿Cuál es la diferencia entre cada uno? ver enunciado 20 pts.**

   El primer query muestra solamente 1 persona con el mayor salario. Si hay salarios repetidos, muestra la primera opción registrada.

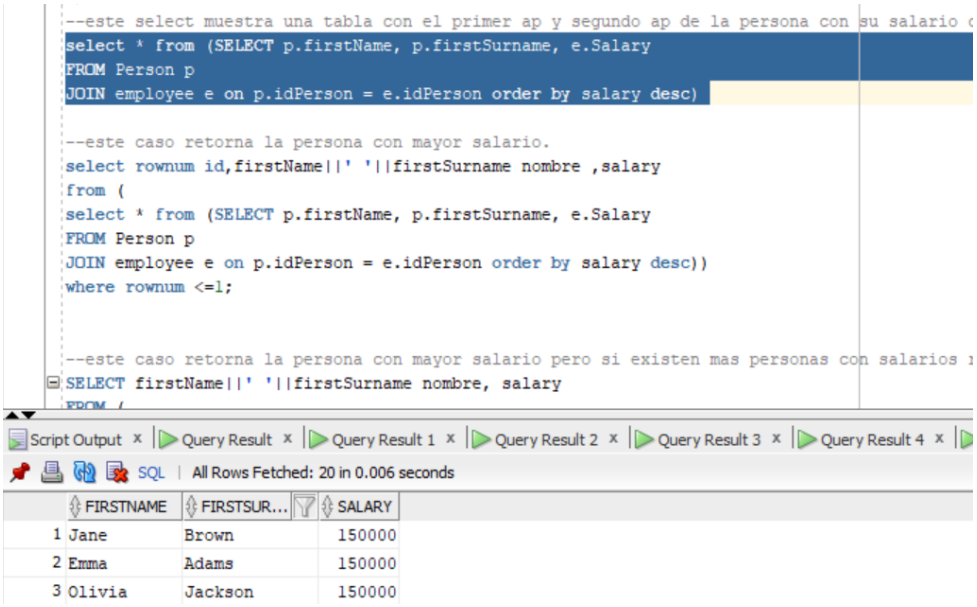
```
--este caso retorna la persona con mayor salario.
select rownum id,firstName||' '||firstSurname nombre ,salary
from (
select * from (SELECT p.firstName, p.firstSurname, e.Salary
FROM Person p
JOIN employee e on p.idPerson = e.idPerson order by salary desc))
where rownum <=1;


--este caso retorna la persona con mayor salario pero si existen mas personas
SELECT firstName||' '||firstSurname nombre, salary
FROM (
select * from (SELECT p.firstName, p.firstSurname, e.salary, rank() over (ord
FROM Person p
JOIN employee e on p.idPerson = e.idPerson order by salary desc)
)
WHERE salary_rank <= 1;

/*8. Consulte la cantidad total de clientes que tienen compras. 5 pts.
ian*/

/*9. Consulte la cantidad total de clientes que existen. 5 pts.
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

SQL | All Rows Fetched: 1 in 0.004 seconds

	ID	NOMBRE	SALARY
1	1	Sarah Harris	700000

Aquí, se hizo un cambio a la tabla y se le ingresaron 5 salarios iguales. El select muestra todas las personas con sus salarios, los tres más altos quedan arriba:



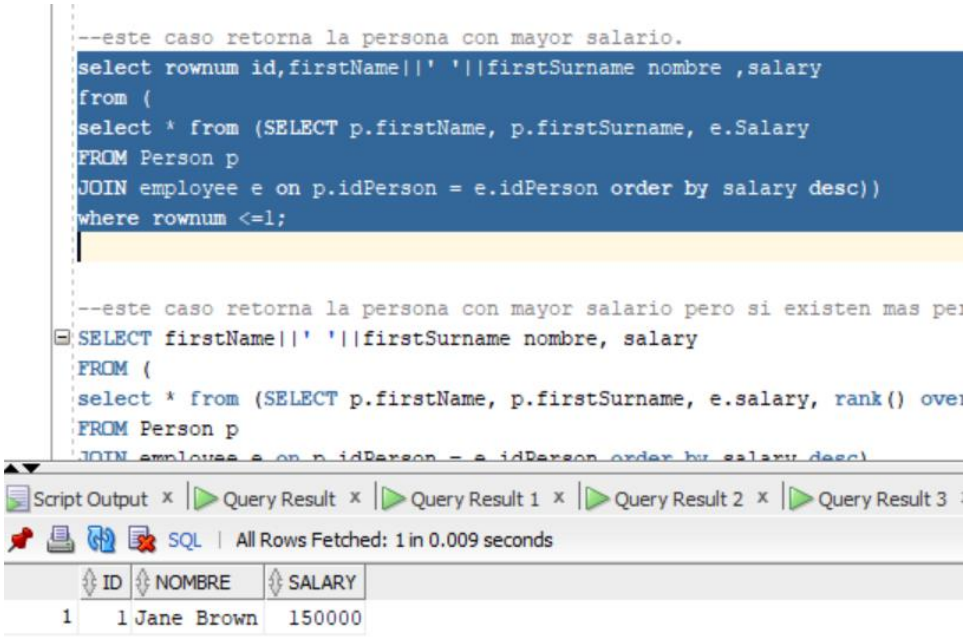
```
--este select muestra una tabla con el primer ap y segundo ap de la persona con su salario
select * from (SELECT p.firstName, p.firstSurname, e.Salary
FROM Person p
JOIN employee e on p.idPerson = e.idPerson order by salary desc)

--este caso retorna la persona con mayor salario.
select rownum id,firstName||' '||firstSurname nombre ,salary
from (
select * from (SELECT p.firstName, p.firstSurname, e.Salary
FROM Person p
JOIN employee e on p.idPerson = e.idPerson order by salary desc))
where rownum <=1;


--este caso retorna la persona con mayor salario pero si existen mas personas con salarios
SELECT firstName||' '||firstSurname nombre, salary
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | All Rows Fetched: 20 in 0.006 seconds

	FIRSTNAME	FIRSTSUR...	SALARY
1	Jane	Brown	150000
2	Emma	Adams	150000
3	Olivia	Jackson	150000

Así actúa el primer query:



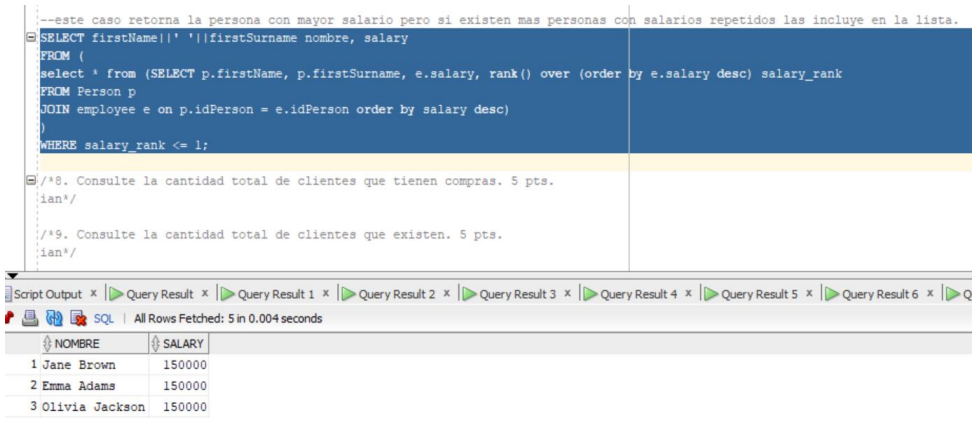
```
--este caso retorna la persona con mayor salario.
select rownum id,firstName||' '||firstSurname nombre ,salary
from (
select * from (SELECT p.firstName, p.firstSurname, e.Salary
FROM Person p
JOIN employee e on p.idPerson = e.idPerson order by salary desc))
where rownum <=1;


--este caso retorna la persona con mayor salario pero si existen mas per
SELECT firstName||' '||firstSurname nombre, salary
FROM (
select * from (SELECT p.firstName, p.firstSurname, e.salary, rank() over
FROM Person p
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

SQL | All Rows Fetched: 1 in 0.009 seconds

	ID	NOMBRE	SALARY
1	1	Jane Brown	150000

Sin embargo, el segundo query actúa así:



```
--este caso retorna la persona con mayor salario pero si existen mas personas con salarios repetidos las incluye en la lista.
SELECT firstName||' '||firstSurname nombre, salary
FROM (
select * from (SELECT p.firstName, p.firstSurname, e.salary, rank() over (order by e.salary desc) salary_rank
FROM Person p
JOIN employee e on p.idPerson = e.idPerson order by salary desc)
)
WHERE salary_rank <= 1;

/*8. Consulte la cantidad total de clientes que tienen compras. 5 pts.
ian*/

/*9. Consulte la cantidad total de clientes que existen. 5 pts.
ian*/
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4 | Query Result 5 | Query Result 6

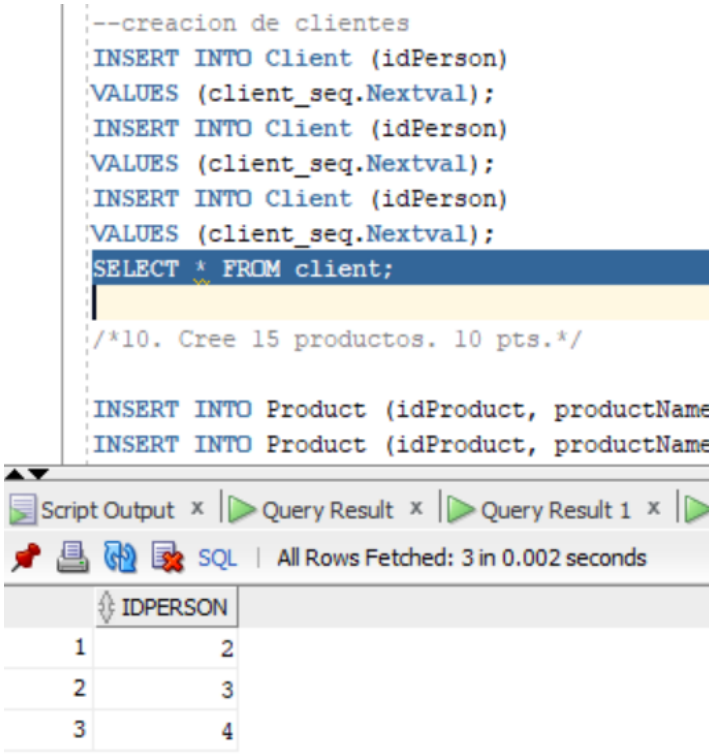
SQL | All Rows Fetched: 5 in 0.004 seconds

	NOMBRE	SALARY
1	Jane Brown	150000
2	Emma Adams	150000
3	Olivia Jackson	150000

Se observa que este incluye los mayores salarios, en caso de que haya más de 1 repetido. A diferencia del primer caso, que sólo incluye el primero que encuentra.

8. Consulte la cantidad total de clientes que tienen compras.

Para los ejemplos 8 y 9 se crearon 3 clientes:

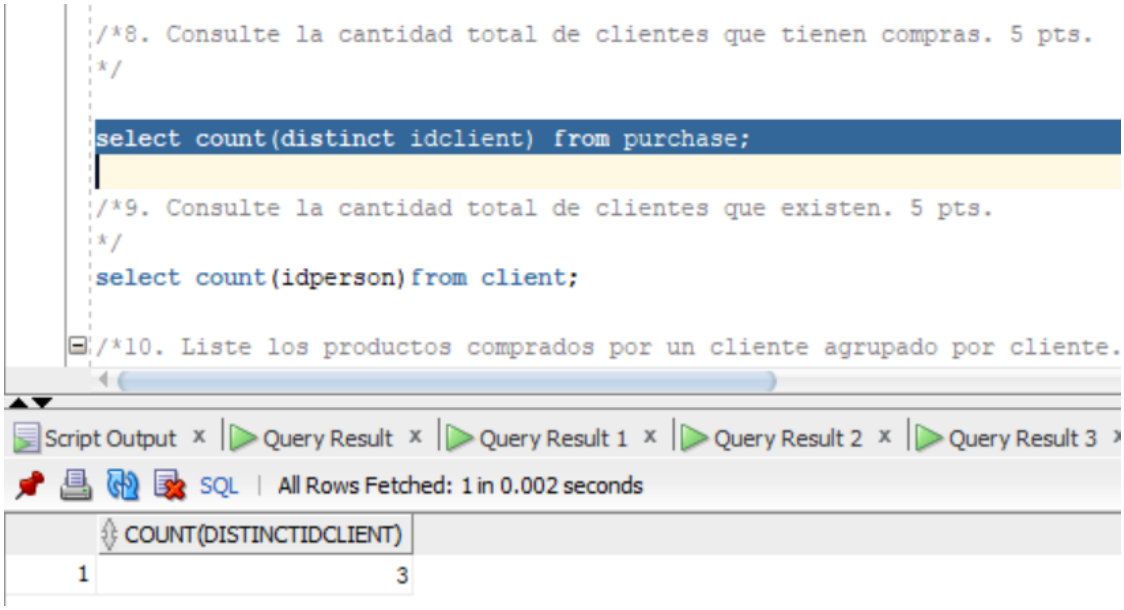


```
--creacion de clientes
INSERT INTO Client (idPerson)
VALUES (client_seq.Nextval);
INSERT INTO Client (idPerson)
VALUES (client_seq.Nextval);
INSERT INTO Client (idPerson)
VALUES (client_seq.Nextval);
SELECT * FROM client;

/*10. Cree 15 productos. 10 pts.*/

INSERT INTO Product (idProduct, productName
INSERT INTO Product (idProduct, productName
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 3 in 0.002 seconds

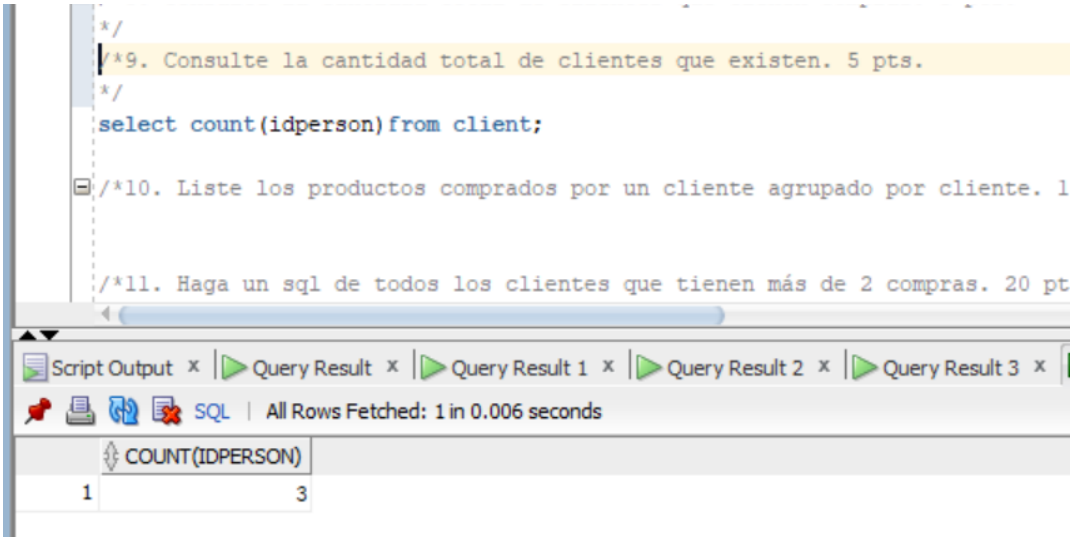
	IDPERSON
1	2
2	3
3	4



El distinct hace que salgan los 3 clientes, y no 4, como en realidad hay 4 ids de clientes dentro de esa tabla porque un id está repetido. Por lo tanto, hay que tomar ese cliente como un solo cliente y el count normalmente lo contaría como 2.

9. Consulte la cantidad total de clientes que existen.

Como se ve en la pregunta 8, hay 3 clientes registrados.



10. Liste los productos comprados por un cliente agrupado por cliente. 10 pts.

```
/*10. Liste los productos comprados por un cliente agrupado por cliente. 10 pts.*/

select e.idclient, d.idproduct producto
from purchase e
right join purchasexproduct d
on e.idpurchase=d.idpurchase
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Re.

SQL | All Rows Fetched: 6 in 0.003 seconds

	IDCLIENT	PRODUCTO
1	2	4
2	2	5
3	2	3
4	3	7
5	4	8
6	4	4

En la siguiente imagen, se ven algunas de las compras de productos asociados a los clientes registrados, que coinciden con la imagen anterior.

```
--compra 2
INSERT INTO Purchase(idPurchase, idClient)
VALUES (purchase_seq.Nextval, 3);

INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (purchase_seq.currval, 7, 2);

--compra 3
INSERT INTO Purchase(idPurchase, idClient)
VALUES (purchase_seq.Nextval, 4);

INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (purchase_seq.currval, 4, 7);

INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (purchase_seq.currval, 8, 2);

select * from purchasexproduct;
```

11. Haga un sql de todos los clientes que tienen más de 2 compras. 20 pts.

Para la pregunta, se hicieron 3 compras asociadas a la misma persona con id 2.

```
--compra 1
INSERT INTO Purchase(idPurchase, idClient)
VALUES (purchase_seq.Nextval, 2);

INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (purchase_seq.currval, 1, 3);

INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (purchase_seq.currval, 4, 5);

--compra 1.1 (mismo cliente)
INSERT INTO Purchase(idPurchase, idClient)
VALUES (purchase_seq.Nextval, 2);

INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (purchase_seq.currval, 3, 2);

INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (purchase_seq.currval, 5, 8);
--compra 1.2(mismo cliente)
INSERT INTO Purchase(idPurchase,idClient)
VALUES (6,2);
INSERT INTO PurchaseXProduct(idPurchase, idProduct, quantity)
VALUES (6,6,5);
```

```
/*11. Haga un sql de todos los clientes que tienen más de 2 compras. 20 pts.*/

SELECT e.idPerson, COUNT(d.idPurchase) AS total
FROM Client e
RIGHT JOIN purchase d
ON e.idPerson = d.idClient
GROUP BY e.idPerson
HAVING COUNT (d.idPurchase) > 2;
```

	IDPERSON	TOTAL
1	2	3

El resultado muestra en total la cantidad de compras de la persona. En este caso solo hay 1 cliente, entonces se muestra 1.