

Una vez se han creado las tablas y se han insertado datos (usando los archivos que se adjuntan en la tarea) podrás ejecutar las consultas en el gestor de Oracle y comprobar el resultado para que le ayudes a Juan.



Creamos las tablas

```
CREATE TABLE CLIENTES
(
  CODIGO VARCHAR2(4) PRIMARY KEY,
  NOMBRE VARCHAR2(30) NOT NULL,
  APELLIDOS VARCHAR2(30) NOT NULL,
  EDAD NUMBER(2,0) NOT NULL
);
```

Results Explain Describe Saved SQL I

Table created.

0.06 seconds



☒ Autocommit Rows 10   Save Run

```
CREATE TABLE PEDIDOS
(
  NUM      NUMBER(5,0) PRIMARY KEY,
  FECHA    DATE NOT NULL,
  GASTOS_ENVIO    NUMBER(5,2),
  FECHA_PREVISTA DATE NOT NULL,
  TOTAL    NUMBER(10,2) ,
  CLIENTE  VARCHAR2(4),
  CONSTRAINT CLIENTES_FK FOREIGN KEY (CLIENTE) REFERENCES CLIENTES (CODIGO)
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds



☒ Autocommit Rows 10   Save Run

```
CREATE TABLE PRODUCTOS
(
  CODIGO    NUMBER(5,0) PRIMARY KEY,
  NOMBRE    VARCHAR2(30) NOT NULL,
  PRECIO    NUMBER(7,2) NOT NULL
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

☒ Autocommit Rows   Save Run



```
CREATE TABLE LINEAS
(
NUM NUMBER(2,0),
NUMPEDIDO NUMBER(5,0),
PRODUCTO NUMBER(5,0) NOT NULL ,
CANTIDAD NUMBER(8,0) NOT NULL ,
IMPORTE NUMBER(6,2),
CONSTRAINT DETALLE_PK PRIMARY KEY (NUM, NUMPEDIDO),
CONSTRAINT PEDIDO_FK FOREIGN KEY (NUMPEDIDO) REFERENCES PEDIDOS (NUM) ON DELETE CASCADE,
CONSTRAINT PRODUCTO_FK FOREIGN KEY (PRODUCTO) REFERENCES PRODUCTOS (CODIGO)
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

E introducimos los datos:



☒ Autocommit Rows   Save Run

```
BEGIN
INSERT INTO CLIENTES VALUES ('C01', 'Luis', 'Garcia Perez', 20);
INSERT INTO CLIENTES VALUES ('C02', 'Maria', 'Lopez García', 50);
INSERT INTO CLIENTES VALUES ('C03', 'Javier', 'Gamez Valiente', 20);
INSERT INTO CLIENTES VALUES ('C04', 'Luis Mª', 'Rico Martin', 17);
INSERT INTO CLIENTES VALUES ('C05', 'Ana Belen', 'Dimas Marco', 25);
INSERT INTO CLIENTES VALUES ('C06', 'Jose Luis', 'García Sanchez', 50);
INSERT INTO CLIENTES VALUES ('C07', 'Mª Pilar', 'Perez Bermejo', 45);
END;
```

Results Explain Describe Saved SQL History

Statement processed.

0.00 seconds



☒ Autocommit Rows   Save Run

```
BEGIN
INSERT INTO PEDIDOS VALUES (1, '11/01/2024', NULL, '11/21/2024', 240, 'C01');
INSERT INTO PEDIDOS VALUES (2, '11/02/2024', NULL, '11/12/2024', 200, 'C02');
INSERT INTO PEDIDOS VALUES (3, '12/03/2024', 10, '12/23/2024', 90, 'C03');
INSERT INTO PEDIDOS VALUES (4, '01/03/2025', 10, '01/23/2025', 100, 'C03');
INSERT INTO PEDIDOS VALUES (5, '01/04/2025', 10, '01/24/2025', 115, 'C04');
INSERT INTO PEDIDOS VALUES (6, '01/14/2025', 10, '02/05/2025', 45, 'C04');
INSERT INTO PEDIDOS VALUES (7, '01/15/2025', 10, '02/05/2025', 45, 'C05');
INSERT INTO PEDIDOS VALUES (8, '02/15/2025', 10, '03/06/2025', 40, 'C05');
INSERT INTO PEDIDOS VALUES (9, '02/25/2025', 5, '03/06/2025', 155, 'C05');
INSERT INTO PEDIDOS VALUES (10, '02/06/2025', 10, '03/08/2025', 70, 'C06');
INSERT INTO PEDIDOS VALUES (11, '02/16/2025', 10, '03/10/2025', 100, 'C06');
INSERT INTO PEDIDOS VALUES (12, '02/26/2025', NULL, '03/11/2025', 320, 'C06');
INSERT INTO PEDIDOS VALUES (13, '03/07/2025', 10, '03/27/2025', 100, 'C07');
INSERT INTO PEDIDOS VALUES (14, '03/17/2025', 10, '03/30/2025', 50, 'C07');
END;
```

Results Explain Describe Saved SQL History

Statement processed.

0.03 seconds



☒ Autocommit Rows   Save Run

```
BEGIN
INSERT INTO PRODUCTOS VALUES (1001, 'RATÓN N', 50);
INSERT INTO PRODUCTOS VALUES (1002, 'RATÓN C', 65);
INSERT INTO PRODUCTOS VALUES (1003, 'RATÓN S/C', 70);
INSERT INTO PRODUCTOS VALUES (2001, 'PENDRIVE 32', 30);
INSERT INTO PRODUCTOS VALUES (2002, 'PENDRIVE 64', 35);
INSERT INTO PRODUCTOS VALUES (3001, 'SDD 512G', 80);
INSERT INTO PRODUCTOS VALUES (3002, 'SDD 1T', 85);
INSERT INTO PRODUCTOS VALUES (4001, 'HDD X', 40);
INSERT INTO PRODUCTOS VALUES (4002, 'HDD Y', 45);
INSERT INTO PRODUCTOS VALUES (4003, 'HDD Z', 50);
END;
```

Results Explain Describe Saved SQL History

Statement processed.

0.00 seconds

☒ Autocommit Rows 10   Save Run

```
INSERT INTO LINEAS VALUES (2,2, 4005, 5, 150);
INSERT INTO LINEAS VALUES (1,3, 4002, 2, 90);
INSERT INTO LINEAS VALUES (1,4, 1001, 2, 100);
INSERT INTO LINEAS VALUES (1,5, 2002, 2, 70);
INSERT INTO LINEAS VALUES (2,5, 4002, 1, 45);
INSERT INTO LINEAS VALUES (1,6, 4002, 1, 45);
INSERT INTO LINEAS VALUES (1,7, 4002, 1, 45);
INSERT INTO LINEAS VALUES (1,8, 4001, 1, 40);
INSERT INTO LINEAS VALUES (1,9, 4003, 1, 50);
INSERT INTO LINEAS VALUES (2,9, 2002, 3, 105);
INSERT INTO LINEAS VALUES (1,10,2002, 2, 70);
INSERT INTO LINEAS VALUES (1,11,4003, 2, 100);
INSERT INTO LINEAS VALUES (1,12,3001, 4, 320);
INSERT INTO LINEAS VALUES (1,13,1001, 2, 100);
INSERT INTO LINEAS VALUES (1,14,1001, 1, 50);
END;
```

Results Explain Describe Saved SQL History

✓

Statement processed.

0.01 seconds

Ahora sí, comenzamos con la Tarea.

Estas son las consultas que debes hacer, para cada una tienes que escribir la sentencia y copiar el resultado obtenido:

Voy a copiar la Query aquí y hacer captura de pantalla del resultado

1. Datos de los productos cuyo precio esté entre 30 y 60 € ordenados de mayor a menor precio

```
SELECT * FROM PRODUCTOS
WHERE PRECIO BETWEEN 30 AND 60
ORDER BY PRECIO DESC;
```

Results Explain Describe Saved SQL History

CODIGO	NOMBRE	PRECIO
1001	RATÓN N	50
4003	HDD Z	50
4002	HDD Y	45
4001	HDD X	40
2002	PENDRIVE 64	35
2001	PENDRIVE 32	30

6 rows returned in 0.01 seconds Download

2. Datos de los pedidos que se han realizado en los últimos 100 días

```
SELECT * FROM PEDIDOS
WHERE FECHA > SYSDATE - 100;
```

Results

Explain

Describe

Saved SQL

History

NUM	FECHA	GASTOS_ENVIO	FECHA_PREVISTA	TOTAL	CLIENTE
3	12/03/2024	10	12/23/2024	90	C03
4	01/03/2025	10	01/23/2025	100	C03
5	01/04/2025	10	01/24/2025	115	C04
6	01/14/2025	10	02/05/2025	45	C04
7	01/15/2025	10	02/05/2025	45	C05
8	02/15/2025	10	03/06/2025	40	C05
9	02/25/2025	5	03/06/2025	155	C05
10	02/06/2025	10	03/08/2025	70	C06
11	02/16/2025	10	03/10/2025	100	C06
12	02/26/2025	-	03/11/2025	320	C06

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds

[Download](#)

3. Lista de los pedidos que tengan gastos de envío con: NUM, FECHA, CLIENTE, TOTAL, GASTOS ENVIO e IMPORTE_TOTAL (que será para cada pedido la suma del importe y los gastos de envío).

```
SELECT NUM, FECHA, CLIENTE, TOTAL, GASTOS_ENVIO, (TOTAL + GASTOS_ENVIO)
AS IMPORTE_TOTAL FROM PEDIDOS WHERE GASTOS_ENVIO IS NOT NULL;
```

Results Explain Describe Saved SQL History

NUM	FECHA	CLIENTE	TOTAL	GASTOS_ENVIO	IMPORTE_TOTAL
3	12/03/2024	C03	90	10	100
4	01/03/2025	C03	100	10	110
5	01/04/2025	C04	115	10	125
6	01/14/2025	C04	45	10	55
7	01/15/2025	C05	45	10	55
8	02/15/2025	C05	40	10	50
9	02/25/2025	C05	155	5	160
10	02/06/2025	C06	70	10	80
11	02/16/2025	C06	100	10	110
13	03/07/2025	C07	100	10	110
14	03/17/2025	C07	50	10	60

11 rows returned in 0 01 seconds
Download

4. Lista de todos los pedidos con: NUM, FECHA, CLIENTE, TOTAL, GASTOS ENVIO e IMPORTE_TOTAL (que será la suma del importe + los gastos de envío).

SELECT NUM, FECHA, CLIENTE, TOTAL, GASTOS_ENVIO, (TOTAL + NVL(GASTOS_ENVIO,0)) AS IMPORTE_TOTAL FROM PEDIDOS;

Aquí hemos añadido NVL(Gastos_Envio,0)) porque antes me salía nulo el importe total cuando el envío era nulo. De esta manera queda:

NUM	FECHA	CLIENTE	TOTAL	GASTOS_ENVIO	IMPORTE_TOTAL
1	11/01/2024	C01	240	-	240
2	11/02/2024	C02	200	-	200
3	12/03/2024	C03	90	10	100
4	01/03/2025	C03	100	10	110
5	01/04/2025	C04	115	10	125
6	01/14/2025	C04	45	10	55
7	01/15/2025	C05	45	10	55
8	02/15/2025	C05	40	10	50
9	02/25/2025	C05	155	5	160
10	02/06/2025	C06	70	10	80
11	02/16/2025	C06	100	10	110
12	02/26/2025	C06	320	-	320
13	03/07/2025	C07	100	10	110
14	03/17/2025	C07	50	10	60

14 rows returned in 0.00 seconds [Download](#)

5. Datos de cada pedido y los datos del cliente de cada uno

Hacemos un Join de las tablas pedidos y clientes.

```
SELECT PEDIDOS.*, CLIENTES.* FROM PEDIDOS JOIN CLIENTES ON
PEDIDOS.CLIENTE = CLIENTES.CODIGO;
```

O también podemos usar los alias:

```
SELECT P.*, C.*
FROM PEDIDOS P
JOIN CLIENTES C ON P.CLIENTE = C.CODIGO;
```


Results

Explain

Describe

Saved SQL

History

NUM	FECHA	GASTOS_ENVIO	FECHA_PREVISTA	TOTAL	CLIENTE	CODIGO	NOMBRE	APELLIDOS	EDAD
1	11/01/2024	-	11/21/2024	240	C01	C01	Luis	Garcia Perez	20
2	11/02/2024	-	11/12/2024	200	C02	C02	Maria	Lopez García	50
3	12/03/2024	10	12/23/2024	90	C03	C03	Javier	Gamez Valiente	20
4	01/03/2025	10	01/23/2025	100	C03	C03	Javier	Gamez Valiente	20
5	01/04/2025	10	01/24/2025	115	C04	C04	Luis Mª	Rico Martin	17
6	01/14/2025	10	02/05/2025	45	C04	C04	Luis Mª	Rico Martin	17
7	01/15/2025	10	02/05/2025	45	C05	C05	Ana Belen	Dimas Marco	25
9	02/25/2025	5	03/06/2025	155	C05	C05	Ana Belen	Dimas Marco	25
8	02/15/2025	10	03/06/2025	40	C05	C05	Ana Belen	Dimas Marco	25
10	02/06/2025	10	03/08/2025	70	C06	C06	Jose Luis	García Sanchez	50
11	02/16/2025	10	03/10/2025	100	C06	C06	Jose Luis	García Sanchez	50
12	02/26/2025	-	03/11/2025	320	C06	C06	Jose Luis	García Sanchez	50
13	03/07/2025	10	03/27/2025	100	C07	C07	Mª Pilar	Perez Bermejo	45
14	03/17/2025	10	03/30/2025	50	C07	C07	Mª Pilar	Perez Bermejo	45

14 rows returned in 0.02 seconds

Download

6. Lista con todos los productos que no se hayan pedido ninguna vez.

```
SELECT *
FROM PRODUCTOS
WHERE CODIGO NOT IN (SELECT PRODUCTO FROM LINEAS);
```

Miramos en la tabla LINEAS, los códigos que no están presentes de entre todos los productos.

Results Explain Describe Saved SQL History

CODIGO	NOMBRE	PRECIO
1002	RATÓN C	65
1003	RATÓN S/C	70
3002	SDD 1T	85

3 rows returned in 0.00 seconds [Download](#)

7. Cantidad total de productos diferentes que vende la empresa (en la columna debe aparecer "Nº de productos")

```
SELECT COUNT(*) AS "Nº de productos"
FROM PRODUCTOS;
```

Results	Explain	Describe	Saved SQL	History
N° de productos				
10				
1 rows returned in 0.00 seconds Download				

8. De cada cliente que haya realizado pedidos en 2024 mostrar su código, nombre y número total de pedidos que ha realizado en 2024

```
SELECT C.CODIGO, C.NOMBRE, COUNT(P.NUM) AS TOTAL_PEDIDOS_2024
FROM CLIENTES C
JOIN PEDIDOS P ON C.CODIGO = P.CLIENTE
WHERE EXTRACT(YEAR FROM P.FECHA) = 2024
GROUP BY C.CODIGO, C.NOMBRE;
```

Hacemos un Join de CLIENTES y PEDIDOS, con la condición de que la parte YEAR de la columna fecha sea 2024. De esa información, hacemos count de la tabla NUM de PEDIDOS.

GROUP BY sirve para agrupar las filas al hacer el COUNT.

Results Explain Describe Saved SQL History

CODIGO	NOMBRE	TOTAL_PEDIDOS_2024
C02	Maria	1
C01	Luis	1
C03	Javier	1

3 rows returned in 0.01 seconds [Download](#)

9. Para **todos los clientes**, mostrar su código, nombre, y número total de pedidos que han realizado durante 2024 incluidos los que hayan hecho 0 pedidos (usando combinación externa)

Entiendo que combinación externa es LEFT JOIN u OUTER JOIN ¿no?

```
SELECT C.CODIGO, C.NOMBRE, COUNT(P.NUM) AS TOTAL_PEDIDOS_2024
FROM CLIENTES C
LEFT JOIN PEDIDOS P ON C.CODIGO = P.CLIENTE AND EXTRACT(YEAR FROM
P.FECHA) = 2024
GROUP BY C.CODIGO, C.NOMBRE;
```

Con el left join conseguimos mostrar todos los clientes(tabla izquierda), incluyendo los que no han hecho ningún pedido

Results	Explain	Describe	Saved SQL	History
CODIGO	NOMBRE	TOTAL_PEDIDOS_2024		
C02	Maria	1		
C04	Luis Mª	0		
C05	Ana Belen	0		
C01	Luis	1		
C03	Javier	1		
C06	Jose Luis	0		
C07	Mª Pilar	0		

7 rows returned in 0.00 seconds [Download](#)

10. Código, nombre y número total de pedidos de los clientes que han realizado más de 2 pedidos

```
SELECT C.CODIGO, C.NOMBRE, COUNT(P.NUM) AS TOTAL_PEDIDOS
FROM CLIENTES C
JOIN PEDIDOS P ON C.CODIGO = P.CLIENTE
GROUP BY C.CODIGO, C.NOMBRE
HAVING COUNT(P.NUM) > 2;
```

Añadimos la cláusula having para filtrar los clientes con más de dos pedidos.

Results	Explain	Describe	Saved SQL	History
CODIGO	NOMBRE	TOTAL_PEDIDOS		
C05	Ana Belen	3		
C06	Jose Luis	3		

2 rows returned in 0.01 seconds [Download](#)

11. Datos del cliente (o clientes) que más pedidos han realizado.

```
SELECT C.*, P.TOTAL_PEDIDOS
FROM CLIENTES C
JOIN (
    SELECT CLIENTE, COUNT(*) AS TOTAL_PEDIDOS
    FROM PEDIDOS
    GROUP BY CLIENTE
) P ON C.CODIGO = P.CLIENTE
WHERE P.TOTAL_PEDIDOS = (
    SELECT MAX(COUNT(*))
    FROM PEDIDOS
    GROUP BY CLIENTE
);
```

En el WHERE encontramos las filas(Clientes) con el mayor número de pedidos.

En el JOIN contamos el número de pedidos por cliente. Los comparamos y tenemos:

Results Explain Describe Saved SQL History				
CODIGO	NOMBRE	APELLIDOS	EDAD	TOTAL_PEDIDOS
C06	Jose Luis	García Sanchez	50	3
C05	Ana Belen	Dimas Marco	25	3

2 rows returned in 0.00 seconds [Download](#)

12. Mostrar código de cliente, nombre de producto y cantidad total de unidades que ha pedido cada cliente de cada producto

```
SELECT C.CODIGO, C.NOMBRE AS NOMBRE_CLIENTE, PR.NOMBRE AS
NOMBRE_PRODUCTO, SUM(L.CANTIDAD) AS CANTIDAD_TOTAL FROM CLIENTES C
JOIN PEDIDOS P ON C.CODIGO = P.CLIENTE JOIN LINEAS L ON P.NUM =
L.NUMPEDIDO JOIN PRODUCTOS PR ON L.PRODUCTO = PR.CODIGO GROUP BY
C.CODIGO,C.NOMBRE, PR.NOMBRE;
```

Tenemos que hacer JOIN de todas las tablas ya que todas intervienen en la operación. Con SUM(L.CANTIDAD) sacamos la cantidad de unidades pedidas.

Results	Explain	Describe	Saved SQL	History
CODIGO	NOMBRE_CLIENTE	NOMBRE_PRODUCTO	CANTIDAD_TOTAL	
C02	Maria	PENDRIVE 32	1	
C05	Ana Belen	PENDRIVE 64	3	
C01	Luis	SDD 512G	1	
C05	Ana Belen	HDD X	1	
C05	Ana Belen	HDD Y	1	
C04	Luis Mª	HDD Y	2	
C06	Jose Luis	PENDRIVE 64	2	
C06	Jose Luis	HDD Z	2	
C06	Jose Luis	SDD 512G	4	
C03	Javier	HDD Y	2	
C05	Ana Belen	HDD Z	1	
C02	Maria	HDD Z	3	
C07	Mª Pilar	RATÓN N	3	
C03	Javier	RATÓN N	2	
C01	Luis	RATÓN N	2	
C01	Luis	PENDRIVE 32	2	
C04	Luis Mª	PENDRIVE 64	2	

17 rows returned in 0.01 seconds [Download](#)

13. Datos de todos los productos que ha pedido el cliente Luis Garcia Perez (sin saber cómo están almacenados estos datos) de dos maneras: una sin combinar tablas, solo con subconsultas y otra sin subconsultas, solo combinando tablas

SUBCONSULTAS

```
SELECT *
FROM PRODUCTOS
WHERE CODIGO IN (
SELECT PRODUCTO
FROM LINEAS
WHERE NUMPEDIDO IN (
SELECT NUM
FROM PEDIDOS
WHERE CLIENTE = (
SELECT CODIGO
```

```

FROM CLIENTES
WHERE NOMBRE = 'Luis' AND APELLIDOS = 'Garcia Perez'
)
)
);

```

1) Mirando esta Matrioska a la inversa, primero sacamos el código cliente de Luis Garcia Perez.

2) Después encontramos los números de pedido realizados por ese código.

3) Encontramos los códigos de producto para esos números de pedido

4) Seleccionamos todos los productos cuyos códigos coincidan.

Results Explain Describe Saved SQL History

CODIGO	NOMBRE	PRECIO
1001	RATÓN N	50
2001	PENDRIVE 32	30
3001	SDD 512G	80

3 rows returned in 0.00 seconds [Download](#)

COMBINANDO TABLAS

```

SELECT PR.*
FROM PRODUCTOS PR
JOIN LINEAS L ON PR.CODIGO = L.PRODUCTO
JOIN PEDIDOS P ON L.NUMPEDIDO = P.NUM
JOIN CLIENTES C ON P.CLIENTE = C.CODIGO
WHERE C.NOMBRE = 'Luis' AND C.APELLIDOS = 'Garcia Perez';

```

Este no tiene mucha explicación, hemos hecho JOIN hasta que hemos sacado los mismos valores que arriba.

Results Explain Describe Saved SQL History

CODIGO	NOMBRE	PRECIO
3001	SDD 512G	80
2001	PENDRIVE 32	30
1001	RATÓN N	50

3 rows returned in 0.01 seconds [Download](#)

14. Producto del que más unidades se han pedido (en total)

```
WITH TOTAL_UNIDADES AS (  
    SELECT PRODUCTO, SUM(CANTIDAD) AS TOTAL  
    FROM LINEAS  
    GROUP BY PRODUCTO  
)  
  
SELECT P.*, TU.TOTAL AS UNIDADES_VENDIDAS  
FROM PRODUCTOS P  
JOIN TOTAL_UNIDADES TU ON P.CODIGO = TU.PRODUCTO  
WHERE TU.TOTAL = (SELECT MAX(TOTAL) FROM TOTAL_UNIDADES);
```

Definimos la expresión TOTAL_UNIDADES con WITH. (Total de unidades pedidas para cada producto). Hacemos un join de nuestra expresión y PRODUCTOS.

Comparamos con WHERE el máximo de unidades vendidas con el total de unidades vendidas. De ahí sacamos el código de producto.

Seleccionamos la información del producto cuyo código coincida con esta comparación.

Results Explain Describe Saved SQL History

CODIGO	NOMBRE	PRECIO	UNIDADES_VENDIDAS
1001	RATÓN N	50	7
2002	PENDRIVE 64	35	7

2 rows returned in 0.01 seconds

Download

15. Datos del producto de precio más caro del pedido 1

```
SELECT L.NUMPEDIDO, P.*  
FROM PRODUCTOS P
```

```

JOIN LINEAS L ON P.CODIGO = L.PRODUCTO
WHERE L.NUMPEDIDO = 1
AND P.PRECIO = (
SELECT MAX(PR.PRECIO)
FROM PRODUCTOS PR
JOIN LINEAS LI ON PR.CODIGO = LI.PRODUCTO
WHERE LI.NUMPEDIDO = 1
);

```

Mostramos el número de pedido de líneas y los datos de productos. Hacemos un join entre estas dos tablas con la condición de que NUMPEDIDO sea 1 y con la expresión AND encontramos el más caro en la columna PRECIO.

NUMPEDIDO	CODIGO	NOMBRE	PRECIO
1	3001	SDD 512G	80

1 rows returned in 0.01 seconds [Download](#)

16. Datos del producto de precio más alto de cada pedido (con una consulta correlacionada)

```

SELECT P.NUMPEDIDO, PR.*
FROM LINEAS P
JOIN PRODUCTOS PR ON P.PRODUCTO = PR.CODIGO
WHERE PR.PRECIO = (
SELECT MAX(PRD.PRECIO)
FROM LINEAS LI
JOIN PRODUCTOS PRD ON LI.PRODUCTO = PRD.CODIGO
WHERE LI.NUMPEDIDO = P.NUMPEDIDO
)
ORDER BY P.NUMPEDIDO;

```

-Mostramos las columnas de PRODUCTOS y la columna NUMPEDIDO de LINEAS. Ordenamos ascendentemente por la columna NUMPEDIDO.

-Dentro del WHERE tenemos una subconsulta relacionada, esto quiere decir que la consulta se ejecuta para cada fila de la consulta externa.

-La subconsulta en sí filtra el MAX(PRECIO) de los PRODUCTOS en el mismo pedido que la fila actual de la consulta externa.

NUMPEDIDO	CODIGO	NOMBRE	PRECIO
1	3001	SDD 512G	80
2	4003	HDD Z	50
3	4002	HDD Y	45
4	1001	RATÓN N	50
5	4002	HDD Y	45
6	4002	HDD Y	45
7	4002	HDD Y	45
8	4001	HDD X	40
9	4003	HDD Z	50
10	2002	PENDRIVE 64	35
11	4003	HDD Z	50
12	3001	SDD 512G	80
13	1001	RATÓN N	50
14	1001	RATÓN N	50

14 rows returned in 0.01 seconds

[Download](#)

17. Para cada cliente, mostrar los datos de su pedido cuyo importe total sea superior al importe total medio de todos sus pedidos (consulta correlacionada).

```
SELECT P.*
FROM PEDIDOS P
WHERE (P.TOTAL + NVL(P.GASTOS_ENVIO, 0)) > (
SELECT AVG(PE.TOTAL + NVL(PE.GASTOS_ENVIO, 0))
FROM PEDIDOS PE
WHERE PE.CLIENTE = P.CLIENTE
);
```

Usamos la subconsulta correlacionada con un AVG, que calcula el importe total medio (Average) de todos los pedidos del mismo cliente.

NUM	FECHA	GASTOS_ENVIO	FECHA_PREVISTA	TOTAL	CLIENTE
4	01/03/2025	10	01/23/2025	100	C03
5	01/04/2025	10	01/24/2025	115	C04
9	02/25/2025	5	03/06/2025	155	C05
12	02/26/2025	-	03/11/2025	320	C06
13	03/07/2025	10	03/27/2025	100	C07

5 rows returned in 0.01 seconds

[Download](#)

18. Código de cada cliente menor de 30 años y cantidad total que gastó en 2024 (incluidos gastos de envío)

```
SELECT C.CODIGO, SUM(P.TOTAL + NVL(P.GASTOS_ENVIO, 0)) AS
GASTO_TOTAL_2024
FROM CLIENTES C
JOIN PEDIDOS P ON C.CODIGO = P.CLIENTE
WHERE C.EDAD < 30
AND EXTRACT(YEAR FROM P.FECHA) = 2024
GROUP BY C.CODIGO;
```

CODIGO	GASTO_TOTAL_2024
C01	240
C03	100

No hay nada nuevo en esta query que no hayamos hecho en las anteriores.

19. Datos del producto más caro y del más barato.

```
SELECT *
FROM PRODUCTOS
WHERE PRECIO = (SELECT MAX(PRECIO) FROM PRODUCTOS)
OR PRECIO = (SELECT MIN(PRECIO) FROM PRODUCTOS);
```

Introducimos las cláusulas con OR para sumar una fila con el precio MIN (Mínimo)

CODIGO	NOMBRE	PRECIO
2001	PENDRIVE 32	30
3002	SDD 1T	85

20. Muestra los pedidos de los clientes con este formato:

```
SELECT
P.NUM AS "Nº PEDIDO",
TO_CHAR(P.FECHA, 'Day:DD/Month/YYYY', 'NLS_DATE_LANGUAGE = SPANISH') AS
"FECHA PEDIDO",
C.APELLIDOS || ' ' || C.NOMBRE AS "NOMBRE COMPLETO"
FROM PEDIDOS P
JOIN CLIENTES C ON P.CLIENTE = C.CODIGO
ORDER BY P.NUM;
```

Con la línea

```
TO_CHAR(P.FECHA, 'Day:DD/Month/YYYY', 'NLS_DATE_LANGUAGE = SPANISH') AS
"FECHA PEDIDO",
```

Conseguimos el formato deseado, con los días de la semana en español.

Nº PEDIDO	FECHA PEDIDO	NOMBRE COMPLETO
1	Viernes :01/Noviembre /2024	Garcia Perez Luis
2	Sábado :02/Noviembre /2024	Lopez García Maria
3	Martes :03/Diciembre /2024	Gamez Valiente Javier
4	Viernes :03/Enero /2025	Gamez Valiente Javier
5	Sábado :04/Enero /2025	Rico Martin Luis Mª
6	Martes :14/Enero /2025	Rico Martin Luis Mª
7	Miércoles:15/Enero /2025	Dimas Marco Ana Belen
8	Sábado :15/Febrero /2025	Dimas Marco Ana Belen
9	Martes :25/Febrero /2025	Dimas Marco Ana Belen
10	Jueves :06/Febrero /2025	García Sanchez Jose Luis
11	Domingo :16/Febrero /2025	García Sanchez Jose Luis
12	Miércoles:26/Febrero /2025	García Sanchez Jose Luis
13	Viernes :07/Marzo /2025	Perez Bermejo Mª Pilar
14	Lunes :17/Marzo /2025	Perez Bermejo Mª Pilar

14 rows returned in 0.01 seconds [Download](#)