

Tarea Unidad 4

Realización de consultas

Base de datos

Curso DAM 2024/25

Calificación obtenida: 10/10

ÍNDICE

Descripción de la tarea	3
Modelo E-R	4
Consultas	5
Consulta 1	5
Consulta 2	6
Consulta 3	7
Consulta 4	9
Consulta 5	11
Consulta 6	12
Consulta 7	13
Consulta 8	14
Consulta 9	16
Consulta 10	17
Consulta 11	19
Consulta 12	20
Anexo 1: Pasos para crear el modelo E-R	23
Anexo 2: create_restaurant_db.sql	27
Anexo 3: script insert_data_restaurant.sql	29

Descripción de la tarea

Se presenta la base de datos de gestión de un restaurante, con pedidos, clientes, reservas, etc.

En primer lugar, deberás descargar el script SQL con la creación de la estructura de esta base de datos: [create restaurant db.sql](#)

Ahora es el momento de insertar todos los datos facilitados en el siguiente script: [script insert_data_restaurant.sql](#)

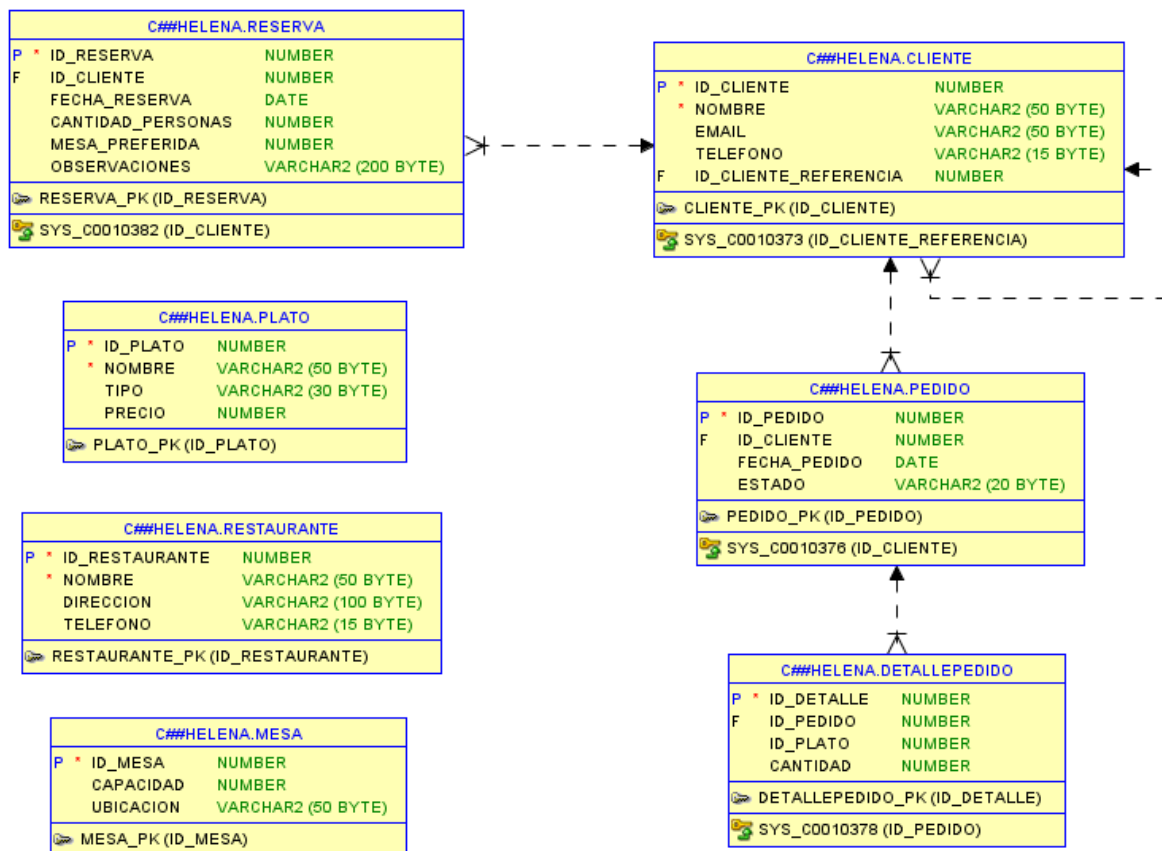
A continuación, explora el modelo relacional de la base de datos. Puedes crearlo manualmente o extraerlo de forma automática mediante SQL Developer siguiendo estas pautas: [Obtención del modelo entidad-relación a partir de las tablas con SQLDeveloper](#)

Resuelve diferentes consultas, elabora una memoria donde aparezca el modelo ER que has obtenido, y la resolución de cada consulta. Para cada consulta deberá aparecer:

- Enunciado de la consulta (texto del enunciado copiado).
- Resolución en SQL (no captura, sino texto escrito).
- Ejecución de la consulta en el entorno (captura)
- Resultados de la ejecución de la consulta en el entorno (captura).
- Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

Modelo E-R

En el [Anexo 1](#) se muestran los pasos seguidos para crear el Modelo E-R.



Consultas

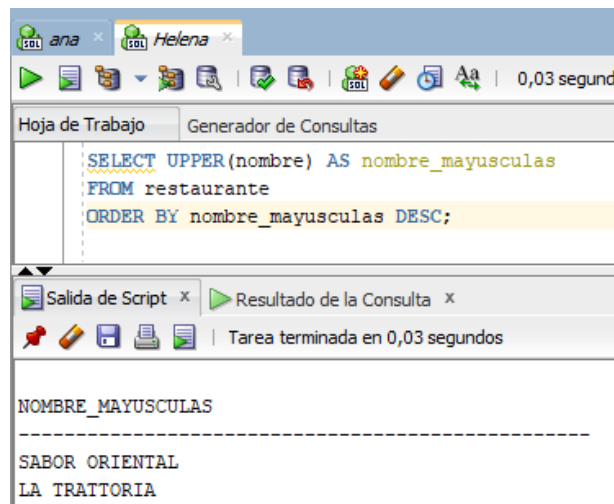
Consulta 1

Obtener todos los datos de los restaurantes ordenados descendientemente por nombre y su nombre deberá aparecer en mayúsculas.

Resolución en SQL

```
SELECT UPPER(nombre) AS nombre_mayusculas  
FROM restaurante  
ORDER BY nombre_mayusculas DESC;
```

Ejecución de la consulta en el entorno



Resultados de la ejecución de la consulta en el entorno

	NOMBRE_MAYUSCULAS
1	SABOR ORIENTAL
2	LA TRATTORIA

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- He utilizado la función UPPER para transformar los valores de la columna nombre a mayúsculas.
- La cláusula ORDER BY ordena los resultados en forma descendente.
- He usado el alias con AS (que es opcional) para darle un nombre más descriptivo a la columna, de esta manera, se puede interpretar más fácilmente lo que representa la columna en los resultados (en este caso, nombres en mayúsculas).

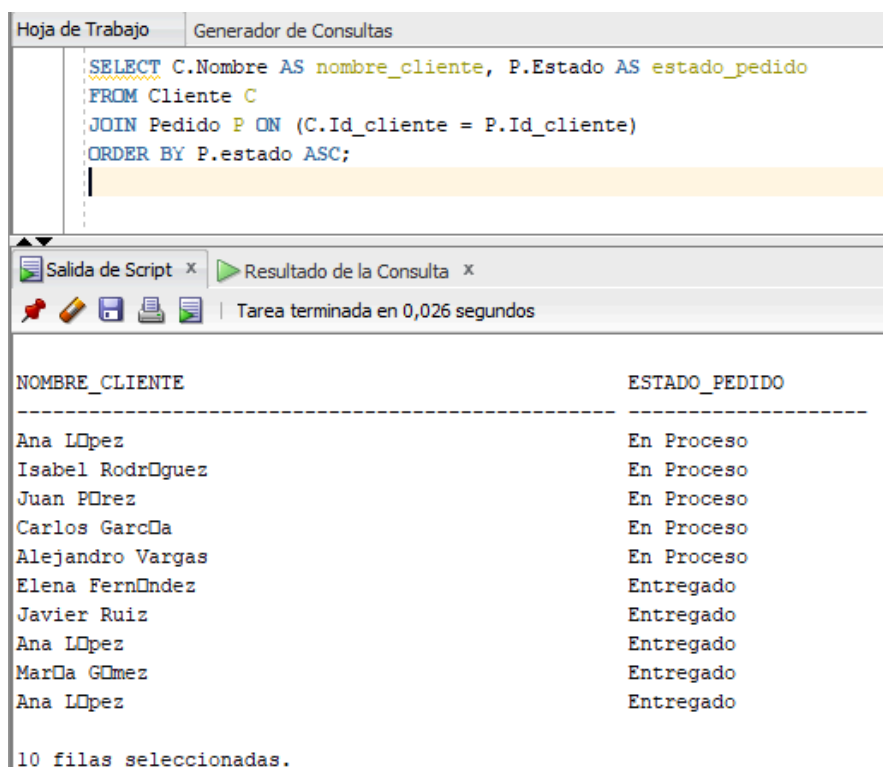
Consulta 2

Obtener el nombre del cliente y el estado de sus pedidos

Resolución en SQL

```
SELECT C.Nombre AS nombre_cliente, P.Estado AS estado_pedido
FROM Cliente C
JOIN Pedido P ON (C.Id_cliente = P.Id_cliente)
ORDER BY P.estado ASC;
```

Ejecución de la consulta en el entorno



Hoja de Trabajo | Generador de Consultas

```
SELECT C.Nombre AS nombre_cliente, P.Estado AS estado_pedido
FROM Cliente C
JOIN Pedido P ON (C.Id_cliente = P.Id_cliente)
ORDER BY P.estado ASC;
```

Salida de Script x | Resultado de la Consulta x

Tarea terminada en 0,026 segundos

NOMBRE_CLIENTE	ESTADO_PEDIDO
Ana López	En Proceso
Isabel Rodríguez	En Proceso
Juan Pérez	En Proceso
Carlos García	En Proceso
Alejandro Vargas	En Proceso
Elena Fernández	Entregado
Javier Ruiz	Entregado
Ana López	Entregado
María Gómez	Entregado
Ana López	Entregado

10 filas seleccionadas.

Resultados de la ejecución de la consulta en el entorno

	NOMBRE_CLIENTE	ESTADO_PEDIDO
1	Ana López	En Proceso
2	Isabel Rodríguez	En Proceso
3	Juan Pérez	En Proceso
4	Carlos García	En Proceso
5	Alejandro Vargas	En Proceso
6	Elena Fernández	Entregado
7	Javier Ruiz	Entregado
8	Ana López	Entregado
9	María Gómez	Entregado
10	Ana López	Entregado

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- Es una consulta de composición interna
- He usado los alias C y P para simplificar las referencias a las tablas en la consulta.
- También he usado alias para las columnas (nombre_cliente y estado_pedido) para darle un nombre más descriptivo a cada columna.
- He utilizado JOIN ON para vincular las tablas Cliente y Pedido a través de la clave foránea id_cliente. También podría haber usado NATURAL JOIN, ya que el nombre de la columna que coincide en ambas tablas es el mismo.
- La cláusula ORDER BY p.estado ASC organiza los resultados en orden ascendente según el estado de los pedidos, facilitando así la interpretación de los datos.

También se podría hacer de la siguiente manera y el resultado sería el mismo:

```
SELECT Nombre, PEDIDO.Estado
FROM CLIENTE, PEDIDO
WHERE PEDIDO.Id_cliente= CLIENTE.Id_cliente
ORDER BY PEDIDO.Estado ASC;
```

Consulta 3

Obtener el ID y el estado de los pedidos y clasificarlos como 'En Curso' o 'Completado'

Resolución en SQL

```
SELECT Id_pedido, Estado, 'En Curso' AS Clasificacion_estado
FROM PEDIDO
WHERE Estado = 'En Proceso'

UNION

SELECT Id_pedido, Estado, 'Completado' AS Clasificacion_estado
FROM PEDIDO
WHERE Estado = 'Entregado'
ORDER BY Clasificacion_estado DESC, Id_pedido ASC;
```

Ejecución de la consulta en el entorno

Hoja de Trabajo Generador de Consultas

```

SELECT Id_pedido, Estado, 'En Curso' AS Clasificacion_estado
FROM PEDIDO
WHERE Estado = 'En Proceso'

UNION

SELECT Id_pedido, Estado, 'Completado' AS Clasificacion_estado
FROM PEDIDO
WHERE Estado = 'Entregado'
ORDER BY Clasificacion_estado DESC, Id_pedido ASC;

```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0,057 segundos

ID_PEDIDO	ESTADO	CLASIFICAC
1	En Proceso	En Curso
4	En Proceso	En Curso
6	En Proceso	En Curso
8	En Proceso	En Curso
10	En Proceso	En Curso
2	Entregado	Completado
3	Entregado	Completado
5	Entregado	Completado
7	Entregado	Completado
9	Entregado	Completado

10 filas seleccionadas.

Resultados de la ejecución de la consulta en el entorno

	ID_PEDIDO	ESTADO	CLASIFICACION_ESTADO
1	1	En Proceso	En Curso
2	4	En Proceso	En Curso
3	6	En Proceso	En Curso
4	8	En Proceso	En Curso
5	10	En Proceso	En Curso
6	2	Entregado	Completado
7	3	Entregado	Completado
8	5	Entregado	Completado
9	7	Entregado	Completado
10	9	Entregado	Completado

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- He utilizado dos partes para clasificar los pedidos según su estado y he combinado los resultados en un único conjunto de datos utilizando el operador UNION:
 - En la primera consulta selecciono los pedidos cuyo estado es 'En proceso' y los clasifico como 'En Curso'. Además creo una columna adicional calculada llamada 'clasificacion_estado' con el valor 'En Curso' para todos los resultados.
 - En la segunda consulta selecciono los pedidos cuyo estado es 'Entregado' y los clasifico como 'Completado'. También uso la columna adicional 'clasificacion_estado' pero ahora con el valor 'Completado'.
 - Uso UNION para combinar los resultados de ambas consultas en una sola tabla eliminando posibles duplicados.
- Al final de la consulta he usado ORDER BY ordenando primero los resultados de forma descendente (DESC) por la columna 'clasificacion_estado' y después de forma ascendente (ASC) por la columna id_pedido.

Consulta 4

Obtener los platos y la cantidad pedida en un rango de fechas: del 1 de marzo al 1 de abril de 2024

Resolución en SQL

```
SELECT
  PEDIDO.Fecha_Pedido AS fecha_pedido,
  PLATO.Nombre AS nombre_plato,
  SUM(DETALLEPEDIDO.Cantidad) AS cantidad_total
FROM DETALLEPEDIDO
JOIN PEDIDO
ON DETALLEPEDIDO.Id_Pedido = PEDIDO.Id_Pedido
JOIN PLATO
ON DETALLEPEDIDO.Id_Plato = PLATO.Id_Plato
WHERE PEDIDO.Fecha_Pedido
  BETWEEN TO_DATE ('01-03-2024', 'DD-MM-YYYY')
  AND TO_DATE ('01-04-2024', 'DD-MM-YYYY')
GROUP BY
  PEDIDO.Fecha_Pedido,
  PLATO.Nombre;
```

Ejecución de la consulta en el entorno

Hoja de Trabajo Generador de Consultas

```

SELECT
    PEDIDO.Fecha_Pedido AS fecha_pedido,
    PLATO.Nombre AS nombre_plato,
    SUM(DETALLEPEDIDO.Cantidad) AS cantidad_total
FROM DETALLEPEDIDO
JOIN PEDIDO
ON DETALLEPEDIDO.Id_Pedido = PEDIDO.Id_Pedido
JOIN PLATO
ON DETALLEPEDIDO.Id_Plato = PLATO.Id_Plato
WHERE PEDIDO.Fecha_Pedido
    BETWEEN TO_DATE ('01-03-2024', 'DD-MM-YYYY')
    AND TO_DATE ('01-04-2024', 'DD-MM-YYYY')
GROUP BY
    PEDIDO.Fecha_Pedido,
    PLATO.Nombre;
    
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0,037 segundos

FECHA_PE	NOMBRE_PLATO	CANTIDAD_TOTAL
02/03/24	Pasta Carbonara	1
02/03/24	Sashimi de Salmón	2

Resultados de la ejecución de la consulta en el entorno

	FECHA_PEDIDO	NOMBRE_PLATO	CANTIDAD_TOTAL
1	02/03/24	Pasta Carbonara	1
2	02/03/24	Sashimi de Salmón	2

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- Selección de columnas: En el SELECT, incluyo la fecha del pedido (PEDIDO.Fecha_Pedido), el nombre del plato (PLATO.Nombre), y la suma total de las cantidades pedidas por cada combinación de fecha de pedido y nombre del plato, utilizando la función de agregación SUM(). Esto me permite obtener el total de las cantidades pedidas para cada plato en un rango de fechas determinado.
- Uso de FROM: En el FROM solo aparece DETALLEPEDIDO porque esta es la tabla principal de la que parte la consulta. Desde ahí, se establecen las relaciones con las otras tablas mediante JOIN para obtener los datos relacionados de otras tablas.
- Uso de JOIN: Utilizo JOIN para relacionar las tablas DETALLEPEDIDO, PEDIDO y PLATO. Esto es necesario para poder acceder a los detalles de los pedidos (como la fecha) y los platos (como el nombre), ya que

DETALLEPEDIDO solo contiene la cantidad de platos pedidos y las claves que permiten identificar los pedidos y los platos.

- Uso de ON: El ON se utiliza para especificar cómo deben relacionarse las tablas. Relaciono DETALLEPEDIDO con PEDIDO a través de la columna Id_Pedido, y DETALLEPEDIDO con PLATO a través de la columna Id_Plato. Estas son las claves que permiten asociar correctamente las filas entre las tablas.
- Uso de WHERE: En el WHERE, filtro los registros para incluir solo aquellos pedidos cuya fecha esté entre el 1 de marzo de 2024 y el 1 de abril de 2024. Esto asegura que solo se muestren los pedidos realizados dentro de ese rango de fechas.
- Uso de GROUP BY: Coloco PEDIDO.Fecha_Pedido y PLATO.Nombre en el GROUP BY porque son columnas que quiero mostrar. Al utilizar la función de agregación SUM() para calcular la cantidad total pedida, es necesario agrupar los resultados por estas columnas para que se muestren las sumas correctas para cada combinación de fecha de pedido y nombre de plato.

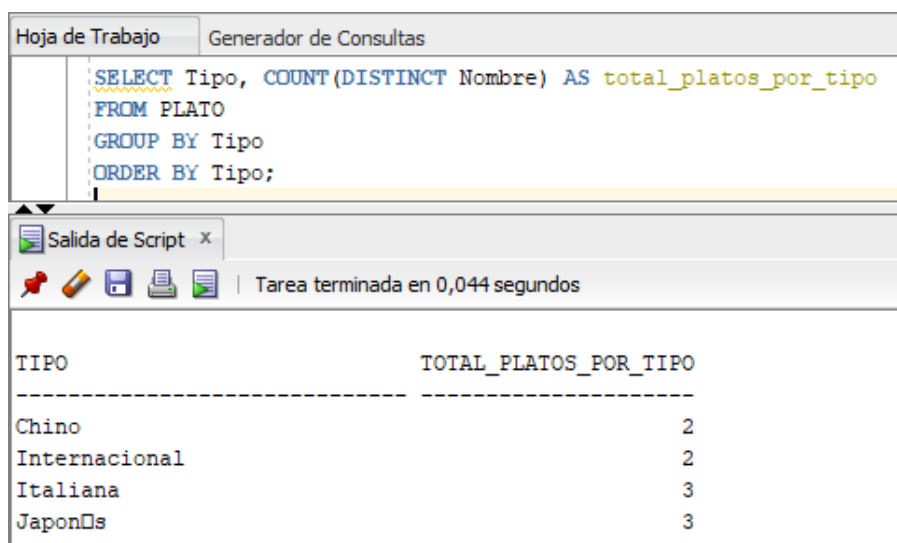
Consulta 5

Obtener la cantidad total de platos por tipo (debe aparecer el nombre de cada grupo de tipo de plato y el total de platos de ese tipo)

Resolución en SQL

```
SELECT Tipo, COUNT(DISTINCT Nombre) AS total_platos_por_tipo
FROM PLATO
GROUP BY Tipo
ORDER BY Tipo;
```

Ejecución de la consulta en el entorno



Hoja de Trabajo Generador de Consultas

```
SELECT Tipo, COUNT(DISTINCT Nombre) AS total_platos_por_tipo
FROM PLATO
GROUP BY Tipo
ORDER BY Tipo;
```

Salida de Script x

Tarea terminada en 0,044 segundos

TIPO	TOTAL_PLATOS_POR_TIPO
Chino	2
Internacional	2
Italiana	3
Japones	3

Resultados de la ejecución de la consulta en el entorno

	TIPO	TOTAL_PLATOS_POR_TIPO
1	Chino	2
2	Internacional	2
3	Italiana	3
4	Japones	3

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- En el SELECT selecciono el campo Tipo de la tabla PLATO y calculo el total de platos únicos por cada tipo usando la función COUNT(DISTINCT Nombre), que cuenta la cantidad de platos diferentes dentro de cada tipo. La palabra clave DISTINCT asegura que solo se cuenten los nombres de platos únicos, evitando duplicados.
- En el FROM especifico la tabla desde la cual se extraen los datos (PLATO)
- En el GROUP BY agrupo los resultados por el campo Tipo.
- Uso ORDER BY para organizar los resultados en orden ascendente según el campo Tipo.

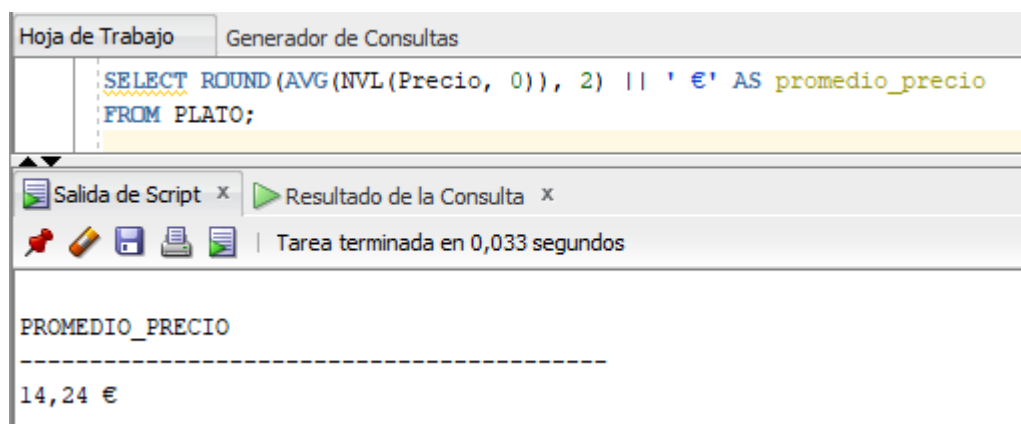
Consulta 6

Obtener el promedio de precios de platos, considerando nulos como 0 €

Resolución en SQL

```
SELECT ROUND(AVG(NVL(Precio, 0)), 2) || ' €' AS promedio_precio
FROM PLATO;
```

Ejecución de la consulta en el entorno



Resultados de la ejecución de la consulta en el entorno

PROMEDIO_PRECIO
1 14,24 €

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- Uso la función de agregación AVG() para calcular el promedio de los precios de los platos en la columna precio.
- Uso la función NVL(precio, 0) para reemplazar los valores nulos (NULL) en la columna precio por un valor predeterminado, en este caso, 0. Esto asegura que los precios nulos no afecten al cálculo del promedio.
- Uso la función ROUND(... , 2) para redondear el resultado del promedio a dos decimales.
- Uso el operador de concatenación || para agregar el símbolo '€' al valor numérico del promedio. Esto permite que el resultado sea más legible y esté en el formato esperado.

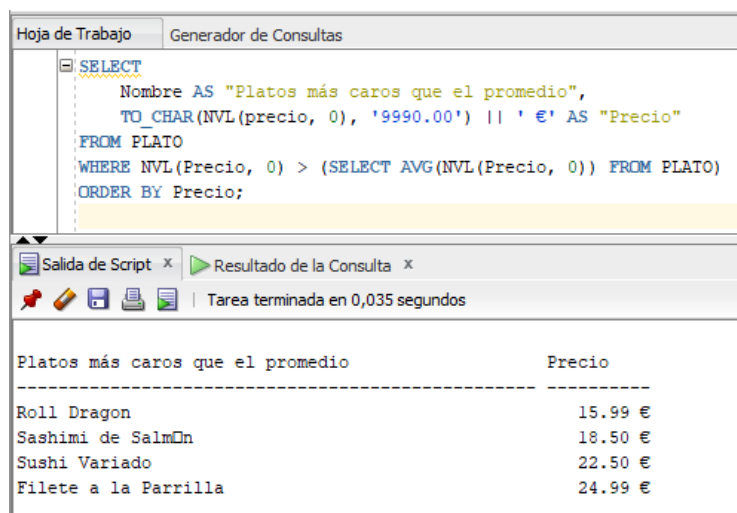
Consulta 7

Obtener el nombre de los platos más caros que el promedio

Resolución en SQL

```
SELECT
  Nombre AS "Platos más caros que el promedio",
  TO_CHAR(NVL(precio, 0), '9990.00') || ' €' AS "Precio"
FROM PLATO
WHERE NVL(Precio, 0) > (SELECT AVG(NVL(Precio, 0)) FROM PLATO)
ORDER BY Precio;
```

Ejecución de la consulta en el entorno



Platos más caros que el promedio	Precio
Roll Dragon	15.99 €
Sashimi de Salmón	18.50 €
Sushi Variado	22.50 €
Filete a la Parrilla	24.99 €

Resultados de la ejecución de la consulta en el entorno

	Platos más caros que el promedio	Precio
1	Roll Dragon	15.99 €
2	Sashimi de Salmón	18.50 €
3	Sushi Variado	22.50 €
4	Filete a la Parrilla	24.99 €

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- En el SELECT selecciono el Nombre de los platos y uso la función TO_CHAR para convertir los valores numéricos en texto, aplicando un formato específico (9990.00). Esto garantiza que los precios se muestren siempre con dos decimales, incluso cuando el valor es exacto (por ejemplo, 15.00 en lugar de 15).
- Uso la función NVL(precio, 0) para asegurar que los valores nulos en la columna Precio se traten como 0, evitando errores al calcular el promedio y permitiendo comparar correctamente los precios.
- En el WHERE hago la subconsulta (SELECT AVG(NVL(Precio, 0)) FROM PLATO) para calcular el promedio de los precios, considerando los valores nulos como 0. Esto garantiza que la comparación se realice correctamente, incluso si hay platos sin precio asignado.
- Uso ORDER BY para ordenar los resultados por precio en orden ascendente

Problemas encontrados:

- Inicialmente usé ROUND pero al mostrar los precios, los decimales no aparecían correctamente debido a la omisión de ceros finales. Solucioné esto aplicando TO_CHAR con un formato que asegura la precisión decimal.

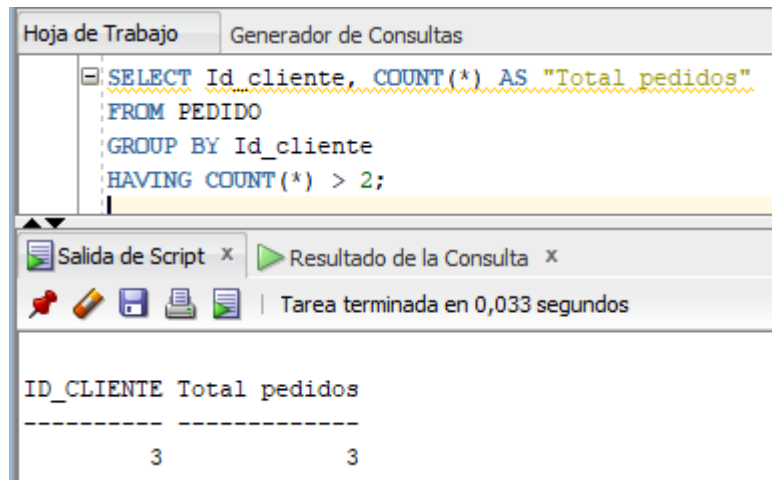
Consulta 8

Obtener el dato id_cliente y el total de pedidos de los clientes con más de 2 pedidos

Resolución en SQL

```
SELECT Id_cliente, COUNT(*) AS "Total pedidos"
FROM PEDIDO
GROUP BY Id_cliente
HAVING COUNT(*) > 2;
```

Ejecución de la consulta en el entorno



Resultados de la ejecución de la consulta en el entorno

ID_CLIENTE	Total pedidos
3	3

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- En el SELECT selecciono la columna `Id_cliente` para poder asociar los pedidos con los clientes correspondientes. Uso la función `COUNT(*)` para contar el número total de pedidos para cada cliente.
- En el GROUP BY agrupo los resultados por cliente (`Id_cliente`). Cada grupo representará a un cliente único, y la función `COUNT(*)` calculará cuántos pedidos tiene cada cliente.
- Uso la cláusula HAVING para filtrar los grupos después de agruparlos. El filtro permite que solo se muestren aquellos clientes que tienen más de 2 pedidos.

Observación: Usar WHERE no funcionaría en este caso porque no puede operar con el resultado de la agregación antes de que se haga el agrupamiento (WHERE se ejecuta siempre antes de que se realice el GROUP BY).

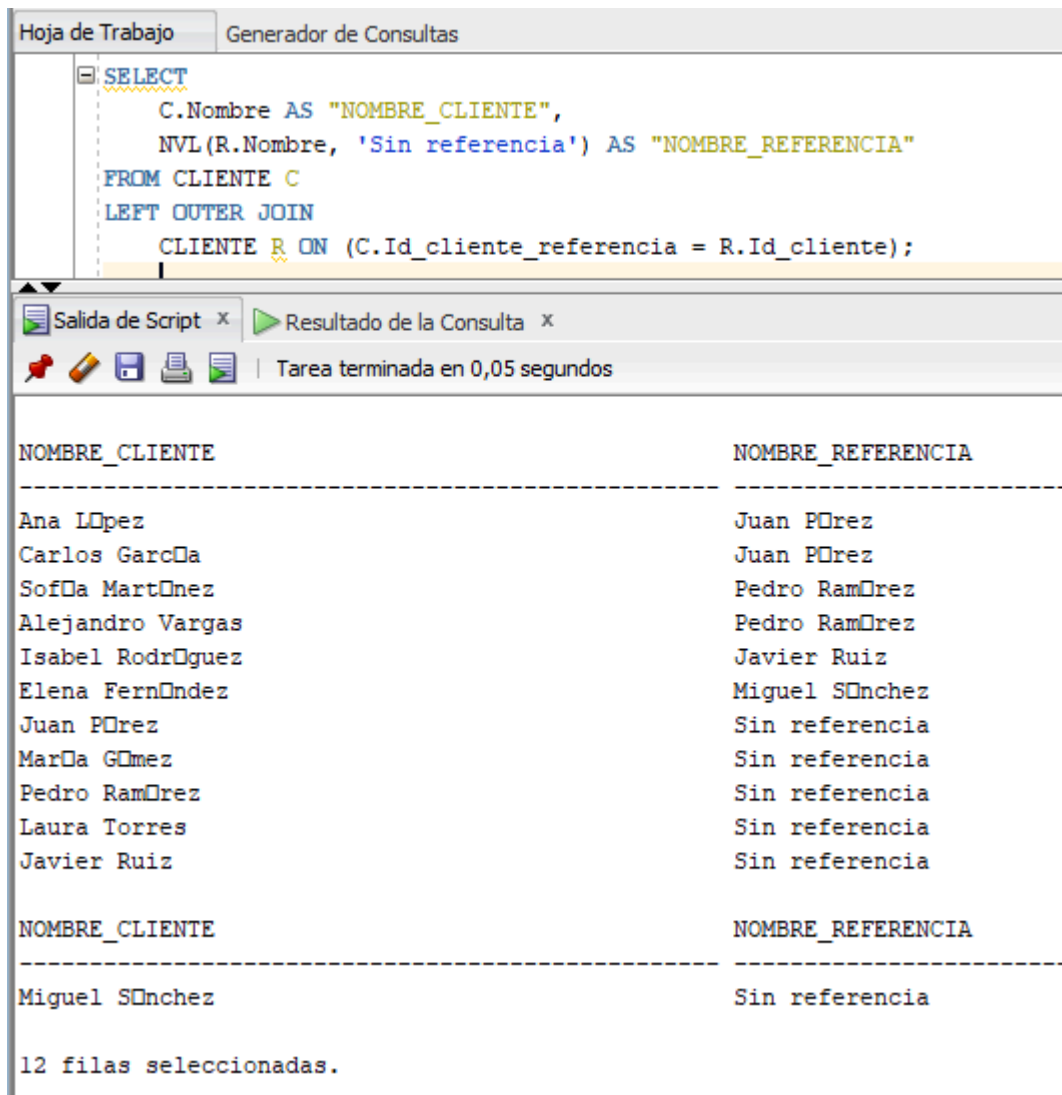
Consulta 9

Obtener los nombres de clientes y los nombres de sus referencias (referencia es el nombre del cliente por el que viene recomendado (id_cliente_referencia))

Resolución en SQL

```
SELECT
  C.Nombre AS "NOMBRE_CLIENTE",
  NVL(R.Nombre, 'Sin referencia') AS "NOMBRE_REFERENCIA"
FROM CLIENTE C
LEFT OUTER JOIN
  CLIENTE R ON (C.Id_cliente_referencia = R.Id_cliente);
```

Ejecución de la consulta en el entorno



The screenshot shows a SQL query execution environment. The top panel displays the query: `SELECT C.Nombre AS "NOMBRE_CLIENTE", NVL(R.Nombre, 'Sin referencia') AS "NOMBRE_REFERENCIA" FROM CLIENTE C LEFT OUTER JOIN CLIENTE R ON (C.Id_cliente_referencia = R.Id_cliente);`. Below the query, a status bar indicates "Tarea terminada en 0,05 segundos". The results are displayed in a table with two columns: "NOMBRE_CLIENTE" and "NOMBRE_REFERENCIA". The table contains 12 rows of data, with the last row being "Miguel Sánchez" and "Sin referencia".

NOMBRE_CLIENTE	NOMBRE_REFERENCIA
Ana López	Juan Pérez
Carlos García	Juan Pérez
Sofía Martínez	Pedro Ramírez
Alejandro Vargas	Pedro Ramírez
Isabel Rodríguez	Javier Ruiz
Elena Fernández	Miguel Sánchez
Juan Pérez	Sin referencia
María Gómez	Sin referencia
Pedro Ramírez	Sin referencia
Laura Torres	Sin referencia
Javier Ruiz	Sin referencia
Miguel Sánchez	Sin referencia

12 filas seleccionadas.

Resultados de la ejecución de la consulta en el entorno

	NOMBRE_CLIENTE	NOMBRE_REFERENCIA
1	Ana López	Juan Pérez
2	Carlos García	Juan Pérez
3	Sofía Martínez	Pedro Ramírez
4	Alejandro Vargas	Pedro Ramírez
5	Isabel Rodríguez	Javier Ruiz
6	Elena Fernández	Miguel Sánchez
7	Juan Pérez	Sin referencia
8	María Gómez	Sin referencia
9	Pedro Ramírez	Sin referencia
10	Laura Torres	Sin referencia
11	Javier Ruiz	Sin referencia
12	Miguel Sánchez	Sin referencia

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- Todos los datos de la consulta provienen de la tabla CLIENTE. En el SELECT, utilizo los alias C y R para distinguir entre el nombre del cliente (C.nombre) y el nombre del cliente de referencia (R.nombre). También uso NVL para incluir los valores NULL los cuales serán reemplazados por "Sin referencia"
- Uso LEFT OUTER JOIN porque quiero que aparezcan todos los clientes, tanto los que tienen un cliente de referencia como los que no. La condición de combinación es C.id_cliente_referencia = R.id_cliente, lo que significa que se busca el cliente en Cliente R cuyo id_cliente coincida con el id_cliente_referencia del cliente en Cliente C.

Consulta 10

Calcula el coste total del pedido con identificador 1 (en euros)

Resolución en SQL

```
SELECT
  P.nombre AS "Platos pedido",
  TO_CHAR(DP.cantidad) AS "Cantidad",
  TO_CHAR(P.precio, '9990.00') || ' €' AS "Precio unidad",
  TO_CHAR(DP.cantidad * P.precio, '9990.00') || ' €' AS "Coste total"
FROM DetallePedido DP
JOIN Plato P ON DP.id_plato = P.id_plato
WHERE DP.id_pedido = 1
```

UNION ALL

```
SELECT
  'COSTE TOTAL DEL PEDIDO' AS "Platos pedido",
  '-' AS "Cantidad",
  '-' AS "Precio unidad",
  TO_CHAR(SUM(DP.cantidad * P.precio), '9990.00') || ' €' AS "Coste total"
FROM DetallePedido DP
JOIN Plato P ON DP.id_plato = P.id_plato
WHERE DP.id_pedido = 1
GROUP BY 'COSTE TOTAL DEL PEDIDO';
```

Ejecución de la consulta en el entorno

Hoja de Trabajo Generador de Consultas

```
SELECT
  P.nombre AS "Platos pedido",
  TO_CHAR(DP.cantidad) AS "Cantidad",
  TO_CHAR(P.precio, '9990.00') || ' €' AS "Precio unidad",
  TO_CHAR(DP.cantidad * P.precio, '9990.00') || ' €' AS "Coste total"
FROM DetallePedido DP
JOIN Plato P ON DP.id_plato = P.id_plato
WHERE DP.id_pedido = 1

UNION ALL

SELECT
  'COSTE TOTAL DEL PEDIDO' AS "Platos pedido",
  '-' AS "Cantidad",
  '-' AS "Precio unidad",
  TO_CHAR(SUM(DP.cantidad * P.precio), '9990.00') || ' €' AS "Coste total"
FROM DetallePedido DP
JOIN Plato P ON DP.id_plato = P.id_plato
WHERE DP.id_pedido = 1
GROUP BY 'COSTE TOTAL DEL PEDIDO';
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0,036 segundos

Platos pedido	Cantidad	Precio uni	Coste tota
Pizza Margherita	2	10.99 €	21.98 €
Sushi Variado	1	22.50 €	22.50 €
COSTE TOTAL DEL PEDIDO	-	-	44.48 €

Resultados de la ejecución de la consulta en el entorno

	Platos pedido	Cantidad	Precio unidad	Coste total
1	Pizza Margherita	2	10.99 €	21.98 €
2	Sushi Variado	1	22.50 €	22.50 €
3	COSTE TOTAL DEL PEDIDO	-	-	44.48 €

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- La consulta consta de dos partes combinadas con un UNION ALL

- Uso TO_CHAR para convertir las cantidades a texto y poder formatear los valores numéricos, como los precios y los costes totales, asegurando que se muestren con dos decimales y con el símbolo de euro (€).
- Uso UNION ALL para combinar los dos conjuntos de resultados. La primera parte devuelve los detalles de los platos del pedido, mientras que la segunda parte calcula y muestra el coste total del pedido.
- Uso SUM() en la segunda parte de la consulta para calcular el coste total del pedido sumando los costes de todos los platos.
- Uso GROUP BY para poder aplicar la función SUM() en el cálculo del coste total.

Observaciones: Primero probé la consulta usando UNION pero el COSTE TOTAL DEL PEDIDO aparecía antes que los platos. Al buscar información sobre esto, encontré que *“el motivo por el cual el coste total aparece en un orden diferente con UNION en lugar de UNION ALL se debe a que UNION ordena los resultados y podría estar reorganizando el conjunto de resultados. UNION ALL, por otro lado, no realiza ninguna operación de ordenación, lo que permite que se mantenga el orden original de las consultas”*

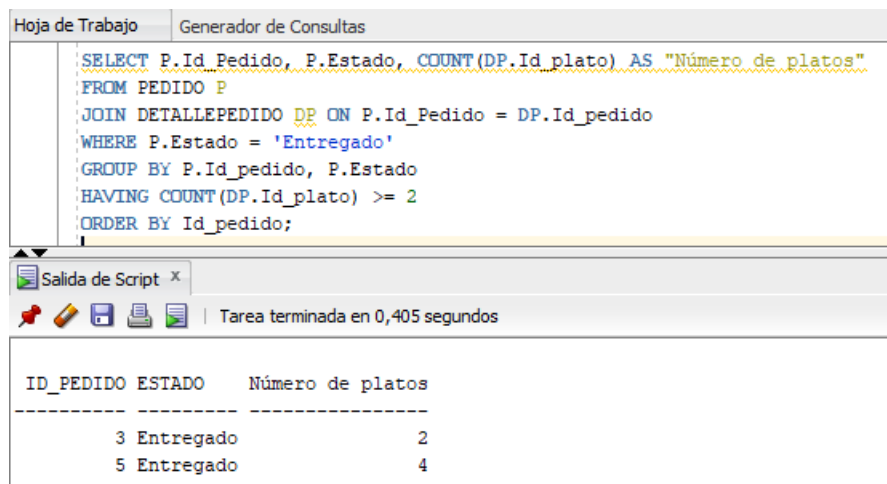
Consulta 11

Obtener los pedidos (ID, estado y número de platos) que han sido entregados con 2 o más platos

Resolución en SQL

```
SELECT P.Id_Pedido, P.Estado, COUNT(DP.Id_plato) AS "Número de platos"
FROM PEDIDO P
JOIN DETALLEPEDIDO DP ON P.Id_Pedido = DP.Id_pedido
WHERE P.Estado = 'Entregado'
GROUP BY P.Id_pedido, P.Estado
HAVING COUNT(DP.Id_plato) >= 2
ORDER BY Id_pedido;
```

Ejecución de la consulta en el entorno



Hoja de Trabajo | Generador de Consultas

```
SELECT P.Id_Pedido, P.Estado, COUNT(DP.Id_plato) AS "Número de platos"
FROM PEDIDO P
JOIN DETALLEPEDIDO DP ON P.Id_Pedido = DP.Id_pedido
WHERE P.Estado = 'Entregado'
GROUP BY P.Id_pedido, P.Estado
HAVING COUNT(DP.Id_plato) >= 2
ORDER BY Id_pedido;
```

Salida de Script x | Tarea terminada en 0,405 segundos

ID_PEDIDO	ESTADO	Número de platos
3	Entregado	2
5	Entregado	4

Resultados de la ejecución de la consulta en el entorno

ID_PEDIDO	ESTADO	Número de platos
1	3 Entregado	2
2	5 Entregado	4

Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- Utilizo un JOIN entre las tablas PEDIDO y DETALLEPEDIDO para asociar cada pedido con los detalles de los platos solicitados. La condición de unión es que el id_pedido de ambas tablas sea igual
- Utilizo el WHERE para filtrar los pedidos y que solo se incluyan aquellos pedidos que cuyo estado sea 'Entregado'.
- Uso el GROUP BY para agrupar los resultados por id de pedido y estado. Esto es necesario porque he usado la función COUNT para poder contar el número de platos de cada pedido .
- Al haber hecho una agrupación tengo que usar el HAVING (no WHERE), mediante el cual filtro los grupos resultantes seleccionando solo aquellos pedidos que tengan 2 o más platos.
- Uso el ORDER BY para ordenar los resultados por el ID de pedido de manera ascendente.

Consulta 12

Obtener la lista de clientes que han realizado pedidos y el total gastado por cada uno de ellos. Incluye también a aquellos clientes que no han realizado ningún pedido. Ordena descendientemente de más gasto a menos gasto (si algún cliente no ha tenido gasto aparecerá un 0 en lugar de NULL)

Resolución en SQL

```
SELECT
  C.nombre AS "Cliente",
  TO_CHAR(NVL((SELECT SUM(DP.cantidad * P.precio)
    FROM DETALLEPEDIDO DP
    JOIN PLATO P ON DP.id_plato = P.id_plato
    JOIN PEDIDO Ped ON DP.id_pedido = Ped.id_pedido
    WHERE Ped.id_cliente = C.id_cliente), 0), '9990.00') || ' €' AS "Total gastado"
FROM Cliente C
ORDER BY "Total gastado" DESC;
```

Ejecución de la consulta en el entorno

```
SELECT
  C.nombre AS "Cliente",
  TO_CHAR(NVL((SELECT SUM(DP.cantidad * P.precio)
                FROM DETALLEPEDIDO DP
                JOIN PLATO P ON DP.id_plato = P.id_plato
                JOIN PEDIDO Ped ON DP.id_pedido = Ped.id_pedido
                WHERE Ped.id_cliente = C.id_cliente), 0), '9990.00') || ' €' AS "Total gastado"
FROM Cliente C
ORDER BY "Total gastado" DESC;
```

Salida de Script x Resultado de la Consulta x
Tarea terminada en 0,037 segundos

Cliente	Total gast
-----	-----
Ana López	159.41 €
Carlos García	49.98 €
Juan Pérez	44.48 €
María Gómez	39.96 €
Isabel Rodríguez	34.00 €
Elena Fernández	31.98 €
Javier Ruiz	9.99 €
Miguel Sánchez	0.00 €
Alejandro Vargas	0.00 €
Laura Torres	0.00 €
Pedro Ramírez	0.00 €

Cliente	Total gast
-----	-----
Sofía Martínez	0.00 €

12 filas seleccionadas.

Resultados de la ejecución de la consulta en el entorno

	Cliente	Total gastado
1	Ana López	159.41 €
2	Carlos García	49.98 €
3	Juan Pérez	44.48 €
4	María Gómez	39.96 €
5	Isabel Rodríguez	34.00 €
6	Elena Fernández	31.98 €
7	Javier Ruiz	9.99 €
8	Miguel Sánchez	0.00 €
9	Alejandro Vargas	0.00 €
10	Laura Torres	0.00 €
11	Pedro Ramírez	0.00 €
12	Sofía Martínez	0.00 €

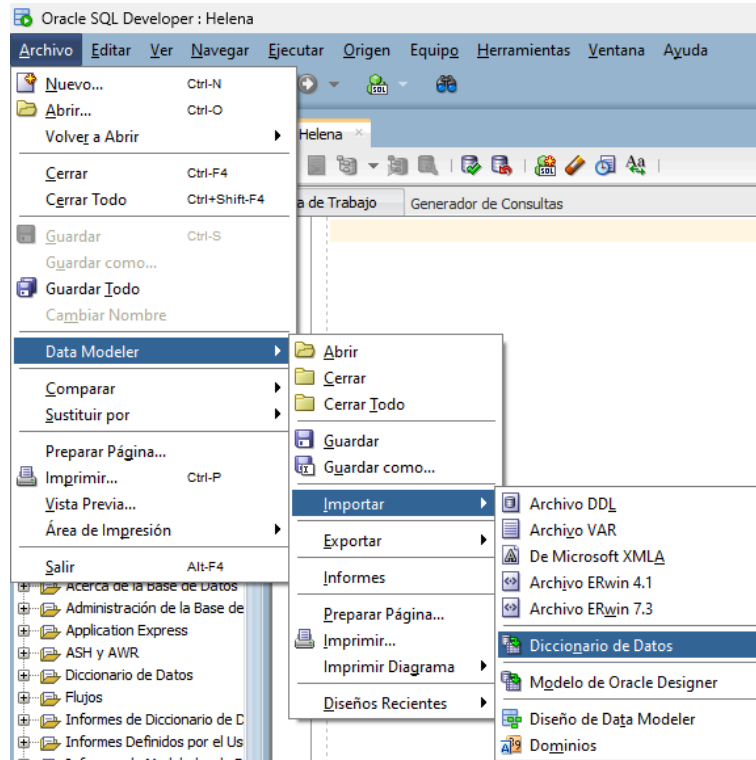
Breve reflexión y posibles comentarios sobre la consulta y los predicados y/o funciones utilizados, problemas encontrados, etc.

- En el SELECT hago una subconsulta para calcular el total gastado por cada cliente la cual multiplica la cantidad de cada plato por su precio.
- Uso la función NVL() para asegurarme de que si un cliente no tiene pedidos (es decir, la subconsulta devuelve NULL), el total gastado se muestre como 0 en lugar de NULL.
- Uso SUM(DP.cantidad * P.precio) para sumar el total gastado por cada cliente a lo largo de todos los pedidos que ha realizado.
- Uso JOIN para combinar 3 tablas y poder calcular el total gastado por cada cliente:
 - JOIN PLATO P ON DP.id_plato = P.id_plato: para obtener el precio de los platos
 - JOIN PEDIDO Ped ON DP.id_pedido = Ped.id_pedido: para obtener el cliente asociado con el pedido
- En el ORDER BY ordeno los resultados de mayor a menor gasto (DESC).

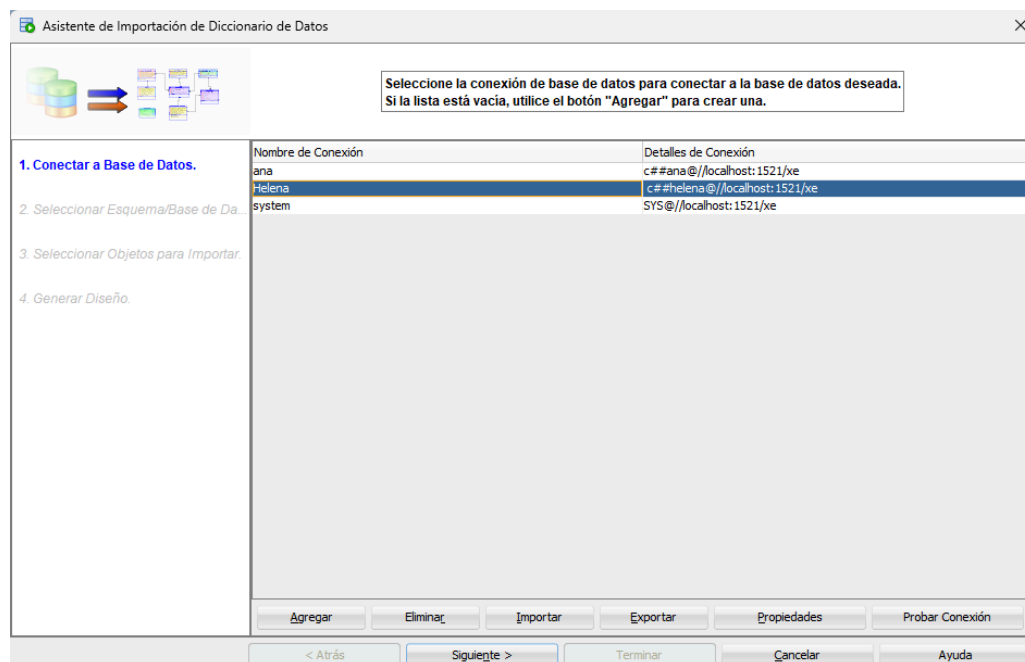
Anexo 1: Pasos para crear el modelo E-R

PASOS SEGUIDOS PARA ELABORAR EL MODELO E-R

Importar el Diccionario de datos



Conectar a Base de datos



Seleccionar Esquema/Base de datos

Asistente de Importación de Diccionario de Datos

Seleccione el esquema/base de datos que desea importar.

1. Conectar a Base de Datos.
2. Seleccionar Esquema/Base de D...
3. Seleccionar Objetos para Importar.
4. Generar Diseño.

Selección	Esquema
☐	ANONYMOUS
☐	APPQOSSYS
☐	AUDSYS
☐	C##ANA
☒	C##HELENA
☐	C##INVITADO
☐	CTXSYS
☐	DBSFUSER
☐	DBSNMP
☐	DIP
☐	DVF
☐	DVSYS
☐	GGSYS
☐	GSMADMIN_INTERNAL
☐	GSMCATUSER
☐	GSMUSER
☐	LBACSYS
☐	MDDATA
☐	MDSYS
☐	OJMSYS
☐	OLAPSYS
☐	ORACLE_OCM

Filtro: ☐ Todos Seleccionados

Importar a: ☐ Intercambiar Modelo de Destino

< Atrás

Seleccionar objetos a importar

Asistente de Importación de Diccionario de Datos

Seleccione los objetos que desea importar.

1. Conectar a Base de Datos.
2. Seleccionar Esquema/Base de D...
3. Seleccionar Objetos para Importar.
4. Generar Diseño.

Selección	Esquema	Nombre de Objeto
☒	C##HELENA	CLIENTE
☒	C##HELENA	DETALLEPEDIDO
☐	C##HELENA	FAMILIA
☒	C##HELENA	MESA
☒	C##HELENA	PEDIDO
☒	C##HELENA	PLATO
☐	C##HELENA	PRODUCTO
☐	C##HELENA	PRODXTIENDAS
☒	C##HELENA	RESERVA
☒	C##HELENA	RESTAURANTE
☒	C##HELENA	TIENDA

Tablas Vistas Usuarios Roles Directorios Tablas Externas Logs de Vistas Materializadas Contextos
Clusters Secuencias Sinónimos Tablespaces Tablespaces Temporales Dimensiones
Tipos Paquetes Procedimientos Almacenados Funciones Tablespaces de Deshacer

Filtro:

< Atrás

24

Terminar



Asistente de Importación de Diccionario de Datos

Ver el resumen y generar el diseño de Oracle SQL Developer Data Modeler.

1. Conectar a Base de Datos.

2. Seleccionar Esquema/Base de D...

3. Seleccionar Objetos para Importar.

4. Generar Diseño.

Nombre de la Base de Datos: Oracle

Versión de la Base de Datos: Oracle Database 18c Express Edition Release 18.0.0.0.0 - Production Version 18.4.0.0.0

Objetos de base de datos que se importarán:

TABLE	7
-------	---

< Atrás

Siguiente >

Terminar

Cancelar

Ayuda

Ver Log

 Ver Log

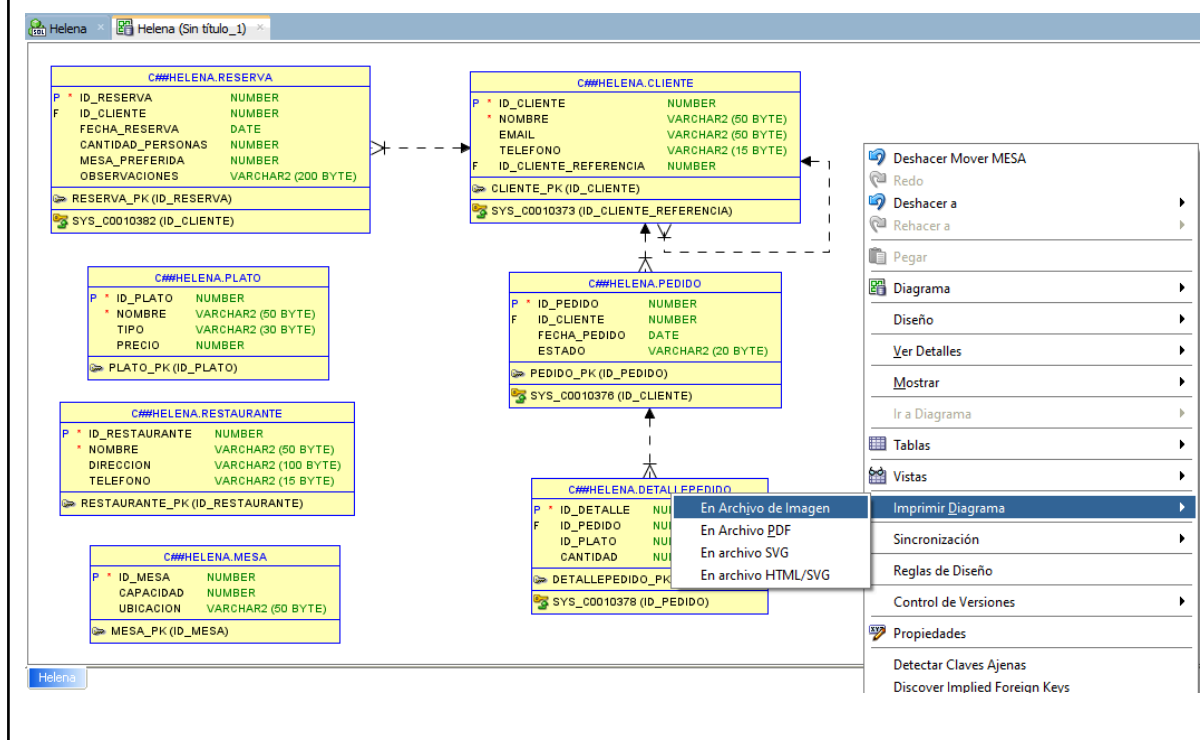
Oracle SQL Developer Data Modeler 24.3.0.275.2224
Oracle SQL Developer Data Modeler Log de Importación
Fecha y Hora: 2024-12-31 06:56:20 CET
Nombre de Diseño: Sin título_1
RDBMS : Oracle Database 12cR2

Todas las Sentencias:		7
Sentencias Importadas:	7	
Sentencias Fallidas:		0
Sentencias No Reconocidas:	0	

Guardar

Cerrar

Guardar el diagrama en un archivo de imagen



Anexo 2: create_restaurant_db.sql

```
-- Creación de tablas
CREATE TABLE Restaurante (
  id_restaurante NUMBER PRIMARY KEY,
  nombre VARCHAR2(50) NOT NULL,
  direccion VARCHAR2(100),
  telefono VARCHAR2(15)
);

CREATE TABLE Cliente (
  id_cliente NUMBER PRIMARY KEY,
  nombre VARCHAR2(50) NOT NULL,
  email VARCHAR2(50),
  telefono VARCHAR2(15)
);

-- Referencia reflexiva
ALTER TABLE Cliente ADD id_cliente_referencia NUMBER REFERENCES
Cliente(id_cliente);

CREATE TABLE Mesa (
  id_mesa NUMBER PRIMARY KEY,
  capacidad NUMBER,
  ubicacion VARCHAR2(50)
);

CREATE TABLE Pedido (
  id_pedido NUMBER PRIMARY KEY,
  id_cliente NUMBER REFERENCES Cliente(id_cliente),
  fecha_pedido DATE,
  estado VARCHAR2(20)
);

CREATE TABLE DetallePedido (
  id_detalle NUMBER PRIMARY KEY,
  id_pedido NUMBER REFERENCES Pedido(id_pedido),
  id_plato NUMBER,
  cantidad NUMBER
);

CREATE TABLE Plato (
  id_plato NUMBER PRIMARY KEY,
  nombre VARCHAR2(50) NOT NULL,
  tipo VARCHAR2(30),
  precio NUMBER
);
```

```
CREATE TABLE Reserva (  
    id_reserva NUMBER PRIMARY KEY,  
    id_cliente NUMBER REFERENCES Cliente(id_cliente),  
    fecha_reserva DATE,  
    cantidad_personas NUMBER,  
    mesa_preferida NUMBER,  
    observaciones VARCHAR2(200)  
);
```

Anexo 3: script insert_data_restaurant.sql

```
-- Inserción de datos
-- Restaurantes
INSERT INTO Restaurante VALUES (1, 'La Trattoria', 'Calle Principal 123', '555-1234');
INSERT INTO Restaurante VALUES (2, 'Sabor Oriental', 'Avenida Secundaria 456',
'555-5678');

-- Clientes
INSERT INTO Cliente VALUES (1, 'Juan Pérez', 'juan@example.com', '555-4321', NULL);
INSERT INTO Cliente VALUES (2, 'María Gómez', 'maria@example.com', '555-8765',
NULL);
INSERT INTO Cliente VALUES (3, 'Ana López', 'ana@example.com', '555-9876', 1); -- Juan
Pérez es la referencia de Ana
INSERT INTO Cliente VALUES (4, 'Pedro Ramírez', 'pedro@example.com', '555-5432',
NULL);
INSERT INTO Cliente VALUES (5, 'Laura Torres', 'laura@example.com', '555-6789', NULL);
INSERT INTO Cliente VALUES (6, 'Carlos García', 'carlos@example.com', '555-8765', 1);
INSERT INTO Cliente VALUES (7, 'Sofía Martínez', NULL, '555-1234', 4);
INSERT INTO Cliente VALUES (8, 'Javier Ruiz', 'javier@example.com', '555-4321', NULL);
INSERT INTO Cliente VALUES (9, 'Isabel Rodríguez', 'isabel@example.com', NULL, 8);
INSERT INTO Cliente VALUES (10, 'Miguel Sánchez', 'miguel@example.com', '555-1111',
NULL);
INSERT INTO Cliente VALUES (11, 'Elena Fernández', 'elena@example.com', '555-2222',
10);
INSERT INTO Cliente VALUES (12, 'Alejandro Vargas', 'alejandro@example.com',
'555-3333', 4);

-- Mesas
INSERT INTO Mesa VALUES (1, 4, 'Interior');
INSERT INTO Mesa VALUES (2, 6, 'Terraza');
INSERT INTO Mesa VALUES (3, 8, 'Jardón');
INSERT INTO Mesa VALUES (4, 4, 'Barra');
INSERT INTO Mesa VALUES (5, 6, 'Terraza VIP');
INSERT INTO Mesa VALUES (6, 2, 'Interior Bar');

-- Pedidos
INSERT INTO Pedido VALUES (1, 1, TO_DATE('2024-01-01', 'YYYY-MM-DD'), 'En
Proceso');
INSERT INTO Pedido VALUES (2, 2, TO_DATE('2024-02-01', 'YYYY-MM-DD'), 'Entregado');
INSERT INTO Pedido VALUES (3, 3, TO_DATE('2024-03-02', 'YYYY-MM-DD'), 'Entregado');
INSERT INTO Pedido VALUES (4, 3, TO_DATE('2024-04-02', 'YYYY-MM-DD'), 'En
Proceso');
INSERT INTO Pedido VALUES (5, 3, TO_DATE('2024-05-02', 'YYYY-MM-DD'), 'Entregado');
INSERT INTO Pedido VALUES (6, 6, TO_DATE('2024-09-15', 'YYYY-MM-DD'), 'En
Proceso');
```

```

INSERT INTO Pedido VALUES (7, 8, TO_DATE('2024-10-20', 'YYYY-MM-DD'), 'Entregado');
INSERT INTO Pedido VALUES (8, 9, TO_DATE('2024-11-10', 'YYYY-MM-DD'), 'En
Proceso');
INSERT INTO Pedido VALUES (9, 11, TO_DATE('2024-12-05', 'YYYY-MM-DD'),
'Entregado');
INSERT INTO Pedido VALUES (10, 12, TO_DATE('2025-01-03', 'YYYY-MM-DD'), 'En
Proceso');

```

-- Detalle de Pedidos

```

INSERT INTO DetallePedido VALUES (1, 1, 1, 2);
INSERT INTO DetallePedido VALUES (2, 1, 2, 1);
INSERT INTO DetallePedido VALUES (3, 2, 3, 4);
INSERT INTO DetallePedido VALUES (4, 3, 4, 1);
INSERT INTO DetallePedido VALUES (5, 3, 5, 2);
INSERT INTO DetallePedido VALUES (6, 4, 6, 3);
INSERT INTO DetallePedido VALUES (7, 5, 7, 1);
INSERT INTO DetallePedido VALUES (8, 5, 8, 2);
INSERT INTO DetallePedido VALUES (9, 5, 9, 1);
INSERT INTO DetallePedido VALUES (10, 5, 10, 2);
INSERT INTO DetallePedido VALUES (11, 6, 7, 2);
INSERT INTO DetallePedido VALUES (12, 7, 8, 1);
INSERT INTO DetallePedido VALUES (13, 8, 9, 4);
INSERT INTO DetallePedido VALUES (14, 9, 10, 2);
INSERT INTO DetallePedido VALUES (15, 10, 11, 3);

```

-- Platos

```

INSERT INTO Plato VALUES (1, 'Pizza Margherita', 'Italiana', 10.99);
INSERT INTO Plato VALUES (2, 'Sushi Variado', 'Japon◆s', 22.50);
INSERT INTO Plato VALUES (3, 'Pollo a la Naranja', 'Chino', 9.99);
INSERT INTO Plato VALUES (4, 'Pasta Carbonara', 'Italiana', 12.99);
INSERT INTO Plato VALUES (5, 'Sashimi de Salm◆n', 'Japon◆s', 18.50);
INSERT INTO Plato VALUES (6, 'Arroz Tres Delicias', 'Chino', 7.99);
INSERT INTO Plato VALUES (7, 'Filete a la Parrilla', 'Internacional', 24.99);
INSERT INTO Plato VALUES (8, 'Ensalada C◆sar', 'Internacional', 9.99);
INSERT INTO Plato VALUES (9, 'Tiramis◆', 'Italiana', 8.50);
INSERT INTO Plato VALUES (10, 'Roll Dragon', 'Japon◆s', 15.99);

```

-- Reservas

```

INSERT INTO Reserva VALUES (1, 1, TO_DATE('2024-03-01', 'YYYY-MM-DD'), 3, 1, 'Mesa
cerca de la ventana');
INSERT INTO Reserva VALUES (2, 2, TO_DATE('2024-04-01', 'YYYY-MM-DD'), 5, 2,
NULL);
INSERT INTO Reserva VALUES (3, 4, TO_DATE('2024-06-01', 'YYYY-MM-DD'), 2, 3, 'Mesa
con sombrilla');
INSERT INTO Reserva VALUES (4, 5, TO_DATE('2024-07-01', 'YYYY-MM-DD'), 4, NULL,
'Preferencia vegetariana');

```

```

INSERT INTO Reserva VALUES (5, 6, TO_DATE('2024-08-01', 'YYYY-MM-DD'), 6, 5,
'Celebraci n de cumplea os');
INSERT INTO Reserva VALUES (6, 7, TO_DATE('2024-09-01', 'YYYY-MM-DD'), 3, 1, 'Mesa
cerca de la ventana');
INSERT INTO Reserva VALUES (7, 8, TO_DATE('2024-10-01', 'YYYY-MM-DD'), 5, 2, 'Sin
preferencias');
INSERT INTO Reserva VALUES (8, 9, TO_DATE('2024-11-01', 'YYYY-MM-DD'), 2, NULL,
NULL);
INSERT INTO Reserva VALUES (9, 10, TO_DATE('2024-12-01', 'YYYY-MM-DD'), 4, 4,
'Mesa con vista al jard n');
INSERT INTO Reserva VALUES (10, 11, TO_DATE('2025-01-01', 'YYYY-MM-DD'), 6, 6,
'Celebraci n de aniversario');

```