Caso práctico "Bases de datos"

Sergio Soler Rocha

1 Crear la tabla Hoteles

```sql
CREATE TABLE Hoteles (
    CodHotel VARCHAR(10) NOT NULL,
    Nombre VARCHAR(30) NOT NULL,
    Direccion VARCHAR(50),
    Ciudad VARCHAR(30) NOT NULL,
    FecInaug DATE,
    Categoria NUMERIC(1) NOT NULL CHECK (Categoria BETWEEN 1 AND 5),
    NumHabitac NUMERIC(3) NOT NULL CHECK (NumHabitac > 0),
    PRIMARY KEY (CodHotel)
);
```

Figure 1: Captura de pantalla del código para crear la tabla Hoteles.

Explicación del código

La tabla **Hoteles** se define con las siguientes columnas y restricciones:

- **CodHotel**: Identificador único de cada hotel. Es la clave primaria, no admite valores nulos y debe ser único.
- **Nombre**: Nombre del hotel. No admite valores nulos.
- **Direccion**: Dirección del hotel. Puede ser nula.
- **Ciudad**: Ciudad donde se ubica el hotel. No admite valores nulos.
- **FecInaug**: Fecha de inauguración del hotel. Puede ser nula.
- **Categoria**: Categoría del hotel. Debe estar entre 1 y 5 y no admite valores nulos.
- **NumHabitac**: Número de habitaciones del hotel. No puede ser cero o negativo y no admite valores nulos.

2 Crear la tabla Clientes

```sql
CREATE TABLE Clientes (
    NIF VARCHAR(9) NOT NULL,
    Nombre VARCHAR(30) NOT NULL,
    Apellido1 VARCHAR(30) NOT NULL,
    Apellido2 VARCHAR(30),
    FecNacim DATE NOT NULL,
    Direccion VARCHAR(50),
    Ciudad VARCHAR(30) NOT NULL,
    PRIMARY KEY (NIF)
);
```

Figure 2: Captura de pantalla del código para crear la tabla Clientes.

Explicación del código

La tabla **Clientes** se define con las siguientes columnas y restricciones:

- **NIF**: Número de Identificación Fiscal del cliente. Es la clave primaria, no admite valores nulos y debe ser único.
- **Nombre**: Nombre del cliente. No admite valores nulos.
- **Apellido1**: Primer apellido del cliente. No admite valores nulos.
- **Apellido2**: Segundo apellido del cliente. Puede ser nulo.
- **FecNacim**: Fecha de nacimiento del cliente. No admite valores nulos.
- **Direccion**: Dirección del domicilio del cliente. Puede ser nula.
- **Ciudad**: Ciudad donde reside el cliente. No admite valores nulos.

3 Crear la tabla Reservas

Explicación del código

La tabla **Reservas** se define con las siguientes columnas y restricciones:

- **CodReserva**: Código único identificativo para cada reserva, utilizado como clave primaria.

```
CREATE TABLE Reservas (
    CodReserva VARCHAR(10) PRIMARY KEY,
    CodHotel VARCHAR(10) NOT NULL,
    NIF VARCHAR(9) NOT NULL,
    FechaEntrada DATE,
    FechaReserva DATE NOT NULL,
    NumNoches NUMERIC(3) NOT NULL CHECK (NumNoches > 0),
    NumHabitac NUMERIC(3) NOT NULL,
    NumAdultos NUMERIC(4) NOT NULL CHECK (NumAdultos > 0) DEFAULT 2,
    NumNinyos NUMERIC(4) NOT NULL CHECK (NumNinyos >= 0) DEFAULT 0,
    PrecioNochHab NUMERIC(6,2) NOT NULL CHECK (PrecioNochHab > 0),
    FOREIGN KEY (CodHotel) REFERENCES Hoteles (CodHotel) ON DELETE CASCADE,
    FOREIGN KEY (NIF) REFERENCES Clientes (NIF) ON DELETE CASCADE
\.
```

Figure 3: Captura de pantalla del código para crear la tabla Reservas.

- **CodHotel**: Código del hotel relacionado con la reserva, que debe existir en la tabla "Hoteles". La relación se establece con una llave foránea y política de borrado en cascada.
- **NIF**: Número de Identificación Fiscal del cliente que realiza la reserva, debe existir en la tabla "Clientes". También es una llave foránea con borrado en cascada.
- **FechaEntrada**: La fecha en que el cliente planea comenzar su estancia. Puede ser nula.
- **FechaReserva**: La fecha en que se realizó la reserva. No admite valores nulos.
- **NumNoches**: Número de noches que el cliente se quedará. Debe ser mayor que cero.
- **NumHabitac**: Número de habitaciones reservadas. No admite valores nulos.
- **NumAdultos**: Número de adultos en la reserva. No puede ser menor que uno y tiene un valor predeterminado de dos.
- **NumNinyos**: Número de niños en la reserva. No puede ser negativo y tiene un valor predeterminado de cero.
- **PrecioNochHab**: Precio por noche por habitación. Debe ser mayor que cero.

4 CONSULTA Listar Hoteles de alta categoría

Esta consulta selecciona todos los hoteles que tienen una categoría de 4 o 5, mostrando el `CodHotel`, nombre, ciudad y categoría.

```
SELECT CodHotel, Nombre, Ciudad, Categoria
FROM Hoteles
WHERE Categoria IN (4,5);
```

Figure 4: Captura de pantalla del código SQL para la consulta de hoteles de alta categoría.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT CodHotel, Nombre, Ciudad, Categoria
FROM Hoteles
WHERE Categoria IN (4, 5)
```

Descripción de la consulta:

- **SELECT CodHotel, Nombre, Ciudad, Categoria**: Esta parte de la consulta especifica las columnas que se quieren recuperar de la tabla Hoteles.
- **FROM Hoteles**: Indica que la consulta se realizará en la tabla Hoteles.
- **WHERE Categoria IN (4, 5)**: Esta condición filtra los hoteles para incluir solo aquellos cuya categoría es 4 o 5.

CodHotel	Nombre	Ciudad	Categoria
H001	Hotel Sol	Madrid	4
H002	Hotel Luna	Barcelona	5
H005	Hotel Galaxia	Madrid	4

Figure 5: Resultado de la consulta de hoteles de alta categoría.

5 CONSULTA Contar Clientes por Ciudad

Esta consulta cuenta cuántos clientes hay de cada ciudad, ordenados de forma descendente según ese número, lo cual es útil para entender la distribución geográfica de la clientela.

```sql
SELECT Ciudad, count(*) as numero_de_clientes
FROM Clientes
group by Ciudad
order by numero_de_clientes
desc;
```

Figure 6: Captura de pantalla del código SQL para la consulta de contar clientes por ciudad.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT Ciudad, COUNT(*) AS NumClientes
FROM Clientes
GROUP BY Ciudad
ORDER BY NumClientes DESC
```

Descripción de la consulta:

- **SELECT Ciudad, COUNT(*) AS NumClientes**: Esta parte selecciona la ciudad de los clientes y cuenta el número total de clientes en cada ciudad, asignando el resultado al alias `NumClientes`.
- **FROM Clientes**: Indica que los datos se están extrayendo de la tabla Clientes.
- **GROUP BY Ciudad**: Agrupa los resultados por ciudad, lo cual es necesario para la función de agregación `COUNT()`.
- **ORDER BY NumClientes DESC**: Ordena los resultados por el número de clientes en cada ciudad de forma descendente.

Ciudad	numero_de_clientes
Madrid	4
Valencia	2
Barcelona	2
Sevilla	2

Figure 7: Resultado de la consulta de contar clientes por ciudad.

6 CONSULTA Detalle de las ocupaciones de los próximos 30 días

Esta consulta recupera detalles de las reservas realizadas para entrar en los próximos 30 días, incluyendo el código de la reserva, el código del hotel, la fecha de reserva, la fecha de entrada, y el número de noches. El resultado es ordenado por fecha de entrada ascendente.

```
SELECT CodReserva, CodHotel, FechaReserva, FechaEntrada, NumNoches
FROM Reservas
WHERE FechaEntrada BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)
ORDER BY FechaEntrada ASC;
```

Figure 8: Captura de pantalla del código SQL para la consulta de detalles de las ocupaciones de los próximos 30 días.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT CodReserva, CodHotel, FechaReserva, FechaEntrada, NumNoches
FROM Reservas
WHERE FechaEntrada BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)
ORDER BY FechaEntrada ASC
```

Descripción de la consulta:

- **SELECT CodReserva, CodHotel, FechaReserva, FechaEntrada, NumNoches**: Selecciona las columnas relevantes para la consulta desde la tabla Reservas.
- **FROM Reservas**: Indica que la consulta se realizará en la tabla Reservas.
- **WHERE FechaEntrada BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)**: Filtra las reservas para incluir solo aquellas cuya fecha de entrada está dentro de los próximos 30 días. `CURDATE()` devuelve la fecha actual en MySQL.
- **ORDER BY FechaEntrada ASC**: Ordena los resultados por la fecha de entrada de forma ascendente, lo que ayuda a ver las reservas más próximas primero.

CodReserva	CodHotel	FechaReserva	FechaEntrada	NumNoches
R024	H004	2024-03-25	2024-04-20	5
R025	H005	2024-04-15	2024-05-01	6
R041	H004	2024-04-06	2024-05-05	10
R033	H001	2024-04-14	2024-05-10	5

Figure 9: Resultado de la consulta de detalles de las ocupaciones de los próximos 30 días.

7 CONSULTA Ingresos futuros por hotel

Esta consulta calcula los ingresos futuros esperados de cada hotel basados en las reservas que aún no han comenzado, utilizando el número de habitaciones reservadas, número de noches y el precio por noche.

```
SELECT CodHotel, SUM(NumHabitac * NumNoches * PrecioNochHab) AS IngresosFuturos
FROM Reservas
WHERE FechaEntrada > CURDATE()
GROUP BY CodHotel
ORDER BY IngresosFuturos DESC;
```

Figure 10: Captura de pantalla del código SQL para la consulta de ingresos futuros por hotel.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT CodHotel, SUM(NumHabitac * NumNoches * PrecioNochHab) AS IngresosFuturos
FROM Reservas
WHERE FechaEntrada > CURDATE()
GROUP BY CodHotel
ORDER BY IngresosFuturos DESC
```

Descripción de la consulta:

- **SELECT CodHotel, SUM(NumHabitac * NumNoches * PrecioNochHab) AS IngresosFuturos**: Esta parte de la consulta calcula los ingresos futuros esperados para cada hotel. Utiliza la función `SUM()` para sumar los ingresos de todas las reservas que corresponden a cada hotel, multiplicando el número de habitaciones reservadas, el número de noches y el precio por noche.
- **FROM Reservas**: Especifica que los datos se extraen de la tabla Reservas.

- **WHERE FechaEntrada ¿ CURDATE()**: Filtra las reservas para incluir solo aquellas cuya fecha de entrada es posterior a la fecha actual, es decir, reservas que aún no han comenzado.
- **GROUP BY CodHotel**: Agrupa los resultados por el código del hotel, lo cual es necesario para aplicar la función de agregación `SUM()`.
- **ORDER BY IngresosFuturos DESC**: Ordena los hoteles de acuerdo con los ingresos futuros esperados en orden descendente, mostrando primero los hoteles con mayores ingresos.

CodHotel	IngresosFuturos
H002	6160.00
H005	5220.00
H004	3900.00
H001	3090.00
H003	1675.00

Figure 11: Resultado de la consulta de ingresos futuros por hotel.

8 CONSULTA Clientes frecuentes por ciudad (3 o más reservas)

Esta consulta identifica a los clientes que han realizado tres o más reservas en hoteles de una misma ciudad. Lista el nombre y primer apellido del cliente, la ciudad de los hoteles y el número total de reservas en hoteles de esa ciudad.

```
SELECT c.Nombre, c.Apellido1, h.Ciudad, COUNT(r.CodReserva) AS TotalReservas
FROM Clientes c
JOIN Reservas r ON c.NIF = r.NIF
JOIN Hoteles h ON r.CodHotel = h.CodHotel
GROUP BY c.Nombre, c.Apellido1, h.Ciudad
HAVING COUNT(r.CodReserva) >= 3
ORDER BY TotalReservas DESC;
```

Figure 12: Captura de pantalla del código SQL para la consulta de clientes frecuentes por ciudad.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT c.Nombre, c.Apellido1, h.Ciudad, COUNT(r.CodReserva) AS TotalReservas
FROM Clientes c
JOIN Reservas r ON c.NIF = r.NIF
JOIN Hoteles h ON r.CodHotel = h.CodHotel
GROUP BY c.Nombre, c.Apellido1, h.Ciudad
HAVING COUNT(r.CodReserva) >= 3
ORDER BY TotalReservas DESC
```

Descripción de la consulta:

- **SELECT c.Nombre, c.Apellido1, h.Ciudad, COUNT(r.CodReserva) AS TotalReservas**: Esta parte de la consulta selecciona el nombre y el primer apellido del cliente, la ciudad del hotel, y cuenta el total de reservas que cada cliente tiene en hoteles de esa ciudad.
- **FROM Clientes c**: Indica que los datos principales provienen de la tabla Clientes, que se abrevia como 'c' para referencias más breves en la consulta.
- **JOIN Reservas r ON c.NIF = r.NIF**: Realiza un JOIN con la tabla Reservas (abreviada como 'r') basándose en el NIF del cliente, lo que une los registros de clientes con sus reservas.
- **JOIN Hoteles h ON r.CodHotel = h.CodHotel**: Realiza un JOIN con la tabla Hoteles (abreviada como 'h') basándose en el código del hotel, lo que vincula cada reserva con el hotel específico.
- **GROUP BY c.Nombre, c.Apellido1, h.Ciudad**: Agrupa los resultados por el nombre y apellido del cliente y por la ciudad del hotel.
- **HAVING COUNT(r.CodReserva) ¿= 3**: Filtra los grupos para incluir solo aquellos donde el número total de reservas es tres o más.
- **ORDER BY TotalReservas DESC**: Ordena los resultados por el número total de reservas en orden descendente, mostrando primero los clientes más frecuentes.

Nombre	Apellido1	Ciudad	TotalReservas
Sofía	Pérez	Madrid	6
Carlos	Jiménez	Barcelona	6
Marta	López	Valencia	4
Antonio	García	Sevilla	4
Beatriz	Vega	Madrid	4
Carmen	Ruiz	Madrid	3
Ana	García	Valencia	3

Figure 13: Resultado de la consulta de clientes frecuentes por ciudad.

9 CONSULTA Reservas para entrar en un hotel en el mismo mes de su inauguración

Listar todas las reservas cuya entrada coincide con el mes de aniversario de la inauguración del hotel, mostrando el nombre del hotel, fecha de inauguración, nombre del cliente y la fecha de entrada. El mes de la fecha de entrada debe coincidir con el mes de la fecha de inauguración.

```
SELECT
    h.Nombre AS NombreHotel,
    h.FecInaug AS FechaInauguracion,
    CONCAT(c.Nombre, ' ', c.Apellido1) AS NombreCliente,
    r.FechaEntrada
FROM Reservas r
JOIN Hoteles h ON r.CodHotel = h.CodHotel
JOIN Clientes c ON r.NIF = c.NIF
WHERE MONTH(r.FechaEntrada) = MONTH(h.FecInaug)
  AND YEAR(r.FechaEntrada) >= YEAR(h.FecInaug)
ORDER BY h.Nombre, r.FechaEntrada;
```

Figure 14: Captura de pantalla del código SQL para la consulta de reservas durante el mes de inauguración del hotel.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT h.Nombre AS NombreHotel, h.FecInaug AS FechaInauguracion,
CONCAT(c.Nombre, ' ', c.Apellido1) AS NombreCliente, r.FechaEntrada
FROM Reservas r
JOIN Hoteles h ON r.CodHotel = h.CodHotel
JOIN Clientes c ON r.NIF = c.NIF
WHERE MONTH(r.FechaEntrada) = MONTH(h.FecInaug)
AND YEAR(r.FechaEntrada) >= YEAR(h.FecInaug)
ORDER BY h.Nombre, r.FechaEntrada
```

Descripción de la consulta:

- **SELECT h.Nombre AS NombreHotel, h.FecInaug AS FechaInauguracion, CONCAT(c.Nombre, ' ', c.Apellido1) AS NombreCliente, r.FechaEntrada**: Selecciona el nombre del hotel, la fecha de inauguración del hotel, el nombre y apellido del cliente concatenados, y la fecha de entrada de la reserva.

- **FROM Reservas r**: Selecciona desde la tabla Reservas, abreviada como 'r'.
- **JOIN Hoteles h ON r.CodHotel = h.CodHotel**: Realiza un JOIN con la tabla Hoteles (abreviada como 'h') basándose en el código del hotel.
- **JOIN Clientes c ON r.NIF = c.NIF**: Realiza un JOIN con la tabla Clientes (abreviada como 'c') basándose en el NIF del cliente.
- **WHERE MONTH(r.FechaEntrada) = MONTH(h.FecInaug) AND YEAR(r.FechaEntrada) ¿= YEAR(h.FecInaug)**: Filtra las reservas para incluir solo aquellas donde el mes de la fecha de entrada coincide con el mes de la fecha de inauguración, y donde el año de la entrada es igual o posterior al año de la inauguración.
- **ORDER BY h.Nombre, r.FechaEntrada**: Ordena los resultados primero por el nombre del hotel y luego por la fecha de entrada.

NombreHotel	FechaInauguracion	NombreCliente	FechaEntrada
Hotel Estrella	2015-03-30	Marta López	2024-03-10
Hotel Luna	2010-06-15	Jorge Fernández	2023-06-05
Hotel Luna	2010-06-15	Carlos Jiménez	2024-06-15
Hotel Sol	2000-05-20	Sofía Pérez	2024-05-10

Figure 15: Resultado de la consulta de reservas durante el mes de inauguración del hotel.

10 CONSULTA Listado de hoteles con meses de alta ocupación en 2023

Listar los hoteles que tuvieron una ocupación de más del 2% en cualquier mes de 2023, mostrando el nombre del hotel, año, mes y el porcentaje de ocupación. El porcentaje de ocupación en un mes se calcula como el número de habitaciones reservadas dividido por el número de habitaciones del hotel multiplicado por 100.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT h.Nombre AS NombreHotel, YEAR(r.FechaEntrada) AS Ano, MONTH(r.FechaEntrada) AS Mes,
(SUM(r.NumHabitac) / h.NumHabitac * 100) AS PorcentajeOcupacion
FROM Reservas r
JOIN Hoteles h ON r.CodHotel = h.CodHotel
WHERE YEAR(r.FechaEntrada) = 2023
```

```sql
SELECT
    h.Nombre AS NombreHotel,
    YEAR(r.FechaEntrada) AS Ano,
    MONTH(r.FechaEntrada) AS Mes,
    (SUM(r.NumHabitac) / h.NumHabitac * 100) AS PorcentajeOcupacion
FROM Reservas r
JOIN Hoteles h ON r.CodHotel = h.CodHotel
WHERE YEAR(r.FechaEntrada) = 2023
GROUP BY h.Nombre, YEAR(r.FechaEntrada), MONTH(r.FechaEntrada), h.NumHabitac
HAVING (SUM(r.NumHabitac) / h.NumHabitac * 100) > 2
ORDER BY h.Nombre, YEAR(r.FechaEntrada), MONTH(r.FechaEntrada);
```

Figure 16: Captura de pantalla del código SQL para la consulta de ocupación de hoteles en 2023.

```
GROUP BY h.Nombre, YEAR(r.FechaEntrada), MONTH(r.FechaEntrada), h.NumHabitac
HAVING (SUM(r.NumHabitac) / h.NumHabitac * 100) > 2
ORDER BY h.Nombre, YEAR(r.FechaEntrada), MONTH(r.FechaEntrada)
```

Descripción de la consulta:

- **SELECT h.Nombre AS NombreHotel, YEAR(r.FechaEntrada) AS Ano, MONTH(r.FechaEntrada) AS Mes, (SUM(r.NumHabitac) / h.NumHabitac * 100) AS PorcentajeOcupacion**: Esta selección muestra el nombre del hotel, el año y el mes de la reserva, y calcula el porcentaje de ocupación como el total de habitaciones reservadas en ese mes dividido por el número total de habitaciones en el hotel, multiplicado por 100 para obtener un porcentaje.
- **FROM Reservas r**: Extrae los datos de la tabla Reservas.
- **JOIN Hoteles h ON r.CodHotel = h.CodHotel**: Une la tabla Reservas con la tabla Hoteles usando el código del hotel.
- **WHERE YEAR(r.FechaEntrada) = 2023**: Filtra las reservas para incluir solo aquellas del año 2023.
- **GROUP BY h.Nombre, YEAR(r.FechaEntrada), MONTH(r.FechaEntrada), h.NumHabitac**: Agrupa los datos por el nombre del hotel, año y mes de la fecha de entrada, así como el total de habitaciones del hotel.
- **HAVING (SUM(r.NumHabitac) / h.NumHabitac * 100) ¿ 2**: Filtra los grupos para incluir solo aquellos donde el porcentaje de ocupación es mayor al 2

- **ORDER BY h.Nombre, YEAR(r.FechaEntrada), MONTH(r.FechaEntrada)**: Ordena los resultados por nombre del hotel, año y mes.

NombreHotel	Ano	Mes	PorcentajeOcupacion
Hotel Cometa	2023	5	6.6667
Hotel Cometa	2023	12	3.3333
Hotel Estrella	2023	5	6.0000
Hotel Galaxia	2023	7	2.5000
Hotel Sol	2023	4	3.0000

Figure 17: Resultado de la consulta de ocupación de hoteles en 2023.

11 CONSULTA Clientes sin reservas recientes

Esta consulta encuentra clientes que no hayan hecho ninguna reserva en el último año, seleccionando su nombre y primer apellido.

```
SELECT c.Nombre, c.Apellido1
FROM Clientes c
LEFT JOIN Reservas r ON c.NIF = r.NIF AND r.FechaReserva >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
WHERE r.CodReserva IS NULL
```

Figure 18: Captura de pantalla del código SQL para la consulta de clientes sin reservas recientes.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT c.Nombre, c.Apellido1
FROM Clientes c
LEFT JOIN Reservas r ON c.NIF = r.NIF AND r.FechaReserva >= DATE_SUB(CURDATE(), INTERVAL 1
WHERE r.CodReserva IS NULL
GROUP BY c.Nombre, c.Apellido1
```

Descripción de la consulta:

- **SELECT c.Nombre, c.Apellido1**: Esta parte de la consulta selecciona el nombre y el primer apellido del cliente.
- **FROM Clientes c**: Indica que los datos se están extrayendo de la tabla Clientes, abreviada como 'c'.

- **LEFT JOIN Reservas r ON c.NIF = r.NIF AND r.FechaReserva ¿= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)**: Realiza un LEFT JOIN con la tabla Reservas (abreviada como 'r'). El LEFT JOIN incluye todas las entradas de la tabla Clientes, independientemente de si tienen correspondencia en la tabla Reservas. La condición adicional asegura que solo se consideren las reservas hechas en el último año.
- **WHERE r.CodReserva IS NULL**: Esta condición filtra para incluir solo aquellos clientes que no tienen reservas en el último año (es decir, donde el código de reserva resultante del JOIN es nulo).
- **GROUP BY c.Nombre, c.Apellido1**: Agrupa los resultados por nombre y apellido del cliente para evitar duplicados en el resultado final.

Nombre	Apellido1
Ana	García
Luis	Martín

Figure 19: Resultado de la consulta de clientes sin reservas recientes.

12 CONSULTA Listado de hoteles y coste medio de habitación

Listado de todos los hoteles con el nombre, categoría y precio medio de la habitación. Ordenar descendente por el precio medio.

```
SELECT
    h.Nombre,
    h.Categoria,
    AVG(r.PrecioNochHab) AS PrecioMedioHabitacion
FROM Hoteles h
LEFT JOIN Reservas r ON h.CodHotel = r.CodHotel
GROUP BY h.Nombre, h.Categoria
ORDER BY PrecioMedioHabitacion DESC;
```

Figure 20: Captura de pantalla del código SQL para la consulta del precio medio de habitación en hoteles.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT h.Nombre, h.Categoria, AVG(r.PrecioNochHab) AS PrecioMedioHabitacion
FROM Hoteles h
LEFT JOIN Reservas r ON h.CodHotel = r.CodHotel
GROUP BY h.Nombre, h.Categoria
ORDER BY PrecioMedioHabitacion DESC
```

Descripción de la consulta:

- **SELECT h.Nombre, h.Categoria, AVG(r.PrecioNochHab) AS PrecioMedioHabitacion**: Esta selección muestra el nombre del hotel, la categoría del hotel, y calcula el precio medio de la habitación usando la función `AVG()` para promediar el precio por noche por habitación registrado en las reservas.
- **FROM Hoteles h**: Indica que los datos se están extrayendo de la tabla Hoteles.
- **LEFT JOIN Reservas r ON h.CodHotel = r.CodHotel**: Realiza un LEFT JOIN con la tabla Reservas (abreviada como 'r'). Este tipo de unión garantiza que se incluyan todos los hoteles, incluso aquellos que no han tenido reservas.
- **GROUP BY h.Nombre, h.Categoria**: Agrupa los resultados por el nombre y la categoría del hotel para permitir el cálculo correcto del promedio.
- **ORDER BY PrecioMedioHabitacion DESC**: Ordena los hoteles por el precio medio de la habitación de manera descendente, mostrando primero aquellos con el precio medio más alto.

Nombre	Categoria	PrecioMedioHabitacion
Hotel Luna	5	153.000000
Hotel Sol	4	115.454545
Hotel Galaxia	4	101.428571
Hotel Estrella	3	86.250000
Hotel Cometa	2	71.875000

Figure 21: Resultado de la consulta del precio medio de habitación en hoteles.

13 VISTA VistaDetalleReservas con toda la información clave de las reservas

Este apartado detalla la creación de una vista llamada `VistaDetalleReservas` que compila información detallada de cada reserva.

```sql
CREATE VIEW VistaDetalleReservas AS
SELECT
    r.CodReserva,
    h.Nombre AS HotelNombre,
    h.Ciudad AS HotelCiudad,
    CONCAT(c.Nombre, ' ', c.Apellido1) AS ClienteNombre,
    c.Apellido1 AS ClienteApellido1,
    r.FechaEntrada,
    r.NumNoches,
    r.NumHabitac,
    r.NumAdultos,
    r.NumNinyos,
    r.PrecioNochHab
FROM Reservas r
JOIN Hoteles h ON r.CodHotel = h.CodHotel
JOIN Clientes c ON r.NIF = c.NIF;
```

Figure 22: Captura de pantalla del SQL para la creación de la vista `VistaDetalleReservas`.

Explicación de la instrucción SQL

La declaración SQL se ejecuta de la siguiente manera:

```
CREATE VIEW VistaDetalleReservas AS
SELECT r.CodReserva,
h.Nombre AS HotelNombre,
h.Ciudad AS HotelCiudad,
CONCAT(c.Nombre, ' ', c.Apellido1) AS ClienteNombre,
r.FechaEntrada, r.NumNoches, r.NumHabitac, r.NumAdultos, r.NumNinyos, r.PrecioNochHab
FROM Reservas r
JOIN Hoteles h ON r.CodHotel = h.CodHotel
JOIN Clientes c ON r.NIF = c.NIF
```

Descripción de la instrucción:

- **CREATE VIEW VistaDetalleReservas AS**: Esto comienza la declaración para crear una nueva vista llamada `VistaDetalleReservas`.
- **SELECT ...**: Lista de campos que se incluirán en la vista, seleccionados y renombrados según sea necesario para proporcionar una descripción clara de cada elemento de la reserva, como el código de la reserva, el nombre y la ciudad del hotel, el nombre completo del cliente, y detalles específicos de la reserva como fechas, número de noches, habitaciones, adultos, niños y precio por noche.

- **FROM Reservas r**: Indica que la fuente principal de datos es la tabla Reservas.
- **JOIN Hoteles h ON r.CodHotel = h.CodHotel**: Une la tabla Reservas con la tabla Hoteles para obtener información del hotel.
- **JOIN Clientes c ON r.NIF = c.NIF**: Une la tabla Reservas con la tabla Clientes para obtener información del cliente.

14 CONSULTA Reservas realizadas en los últimos 30 días

Esta consulta utiliza la vista `VistaDetalleReservas` para encontrar todas las reservas con entrada en los próximos 30 días, mostrando todos los campos disponibles de la vista.

```
SELECT *
FROM VistaDetalleReservas
WHERE FechaEntrada BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)
ORDER BY FechaEntrada ASC;
```

Figure 23: Captura de pantalla del código SQL para la consulta de reservas en los próximos 30 días usando la vista `VistaDetalleReservas`.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT *
FROM VistaDetalleReservas
WHERE FechaEntrada BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)
ORDER BY FechaEntrada ASC
```

Descripción de la consulta:

- **SELECT ***: Selecciona todos los campos disponibles de la vista `VistaDetalleReservas`. Dado que la vista ya contiene todos los campos relevantes de las tablas Reservas, Hoteles y Clientes, esto incluye información detallada sobre cada reserva.
- **FROM VistaDetalleReservas**: Indica que la consulta se ejecuta sobre la vista `VistaDetalleReservas`.

- **WHERE FechaEntrada BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)**: Filtra las reservas para incluir solo aquellas cuya fecha de entrada está entre la fecha actual (CURDATE()) y la fecha actual más 30 días. Esto asegura que solo se incluyan las reservas con entrada en los próximos 30 días.
- **ORDER BY FechaEntrada ASC**: Ordena las reservas por la fecha de entrada de forma ascendente, lo que ayuda a visualizar las reservas en el orden en que ocurrirán.

CodReserva	HotelNombre	HotelCiudad	ClienteNombre	ClienteApellido1	FechaEntrada	NumNoches	NumHabitac	NumAdultos	NumNinyos	PrecioNochHab
R024	Hotel Cometa	Sevilla	Antonio García	García	2024-04-20	5	1	2	3	70.00
R025	Hotel Galaxia	Madrid	Beatriz Vega	Vega	2024-05-01	6	2	2	2	95.00
R041	Hotel Cometa	Sevilla	Antonio García	García	2024-05-05	10	2	2	2	80.00
R033	Hotel Sol	Madrid	Sofía Pérez	Pérez	2024-05-10	5	1	2	1	110.00

Figure 24: Resultado de la consulta de reservas en los próximos 30 días.

15 FUNCION CostoReserva() que calcule el costo de una reserva

Esta sección describe la creación de la función `CostoReserva()` que toma como entrada el código de una reserva y devuelve el costo de esa reserva específica, calculando el costo total de la reserva.

```
DELIMITER //

CREATE FUNCTION CostoReserva(CodReserva VARCHAR(10))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE costo DECIMAL(10,2);
    SELECT SUM(NumNoches * NumHabitac * PrecioNochHab) INTO costo
    FROM Reservas
    WHERE CodReserva = CodReserva;
    RETURN costo;
END //

DELIMITER ;
```

Figure 25: Captura de pantalla del SQL para la función `CostoReserva`.

Explicación de la función

La declaración SQL se ejecuta de la siguiente manera:

```
CREATE FUNCTION CostoReserva(CodReserva VARCHAR(10)) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
DECLARE costo DECIMAL(10,2);
SELECT SUM(NumNoches * NumHabitac * PrecioNochHab) INTO costo
FROM Reservas WHERE CodReserva = CodReserva;
RETURN costo;
END;
```

Descripción de la función:

- **CREATE FUNCTION CostoReserva(CodReserva VARCHAR(10))**: Define la función `CostoReserva` que toma un parámetro `CodReserva`, el cual es el código identificador de la reserva.
- **RETURNS DECIMAL(10,2)**: Especifica que la función retornará un valor decimal con dos dígitos decimales, adecuado para representar cantidades monetarias.
- **DETERMINISTIC**: Indica que la función retorna el mismo resultado cada vez que se le pasan los mismos parámetros.
- **DECLARE costo DECIMAL(10,2)**: Declara una variable local `costo` para almacenar el resultado del cálculo.
- **SELECT SUM(NumNoches * NumHabitac * PrecioNochHab) INTO costo FROM Reservas WHERE CodReserva = CodReserva**: Calcula el costo total de la reserva seleccionando y sumando el producto del número de noches, el número de habitaciones y el precio por noche, para la reserva especificada.
- **RETURN costo**: Retorna el valor calculado de `costo`.

16 CONSULTA Costo medio de las reservas de cada cliente

Esta consulta lista el nombre de todos los clientes, el número de reservas que ha hecho, el costo medio de las reservas que ha hecho y el costo total de las reservas hechas, empleando la función `CostoReserva()`.

Explicación de la consulta

La consulta se ejecuta de la siguiente manera:

```
SELECT c.Nombre, COUNT(r.CodReserva) AS NumeroDeReservas,
AVG(CostoReserva(r.CodReserva)) AS CostoMedio,
SUM(CostoReserva(r.CodReserva)) AS CostoTotal
```

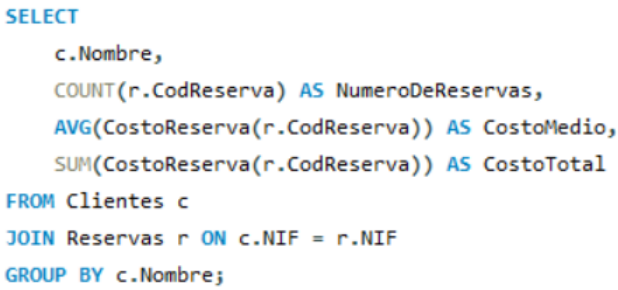

Figure 26: Captura de pantalla del código SQL para la consulta del costo medio de las reservas de cada cliente.

```
FROM Clientes c
JOIN Reservas r ON c.NIF = r.NIF
GROUP BY c.Nombre
```

Descripción de la consulta:

- **SELECT c.Nombre, COUNT(r.CodReserva) AS NumeroDeReservas, AVG(CostoReserva(r.CodReserva)) AS CostoMedio, SUM(CostoReserva(r.CodReserva AS CostoTotal**: Esta línea selecciona el nombre del cliente, cuenta el número de reservas que ha hecho, calcula el costo medio de esas reservas utilizando la función `CostoReserva()`, y suma el costo total de todas las reservas del cliente.
- **FROM Clientes c**: Indica que los datos se están extrayendo de la tabla Clientes.
- **JOIN Reservas r ON c.NIF = r.NIF**: Realiza un JOIN con la tabla Reservas para acceder a las reservas asociadas a cada cliente.
- **GROUP BY c.Nombre**: Agrupa los resultados por el nombre del cliente para asegurar que el conteo, el promedio y la suma de costos se calculen por cliente.

Nombre	NumeroDeReservas	CostoMedio	CostoTotal
Carmen	4	40385.000000	161540.00
Ana	5	40385.000000	201925.00
Sofía	6	40385.000000	242310.00
Carlos	6	40385.000000	242310.00
Jorge	5	40385.000000	201925.00
Marta	4	40385.000000	161540.00
Laura	3	40385.000000	121155.00
Antonio	4	40385.000000	161540.00
Beatriz	4	40385.000000	161540.00
Luis	3	40385.000000	121155.00

Figure 27: Resultado de la consulta del costo medio de las reservas de cada cliente.