

## 1. El trabajo debe ser INDIVIDUAL y ORIGINAL (creado por el propio estudiante)

La base de datos **empresa\_camiones** está diseñada para gestionar y organizar la información clave de una empresa de transporte de camiones, he tomado esta referencia gracias a mi padre el cual la fundó hace más de 20 años y nunca ha ido mejor. Esta empresa ha crecido y evolucionado a lo largo de las décadas, y la base de datos refleja esta evolución

- **Empleados:** Almacena la información de los empleados de la empresa, incluyendo datos personales, puesto, salario y fecha de contratación.
- **Clientes:** Contiene los datos de los clientes, como nombre, apellidos, correo electrónico y teléfono.
- **Camiones:** Registra detalles sobre los camiones disponibles en la empresa, incluyendo marca, modelo, año de fabricación, precio y disponibilidad.
- **Ventas:** Guarda información sobre las ventas de camiones, enlazando clientes y camiones con detalles de fecha y precio de venta.
- **Tipos de Mercancía:** Define los diferentes tipos de mercancía que la empresa transporta.
- **Envíos:** Captura los detalles de los envíos realizados, incluyendo origen, destino, tipo de mercancía y fecha de envío.
- **Mantenimiento:** Almacena registros de mantenimiento realizados en los camiones, con descripciones, costos y fechas.
- **Historial de Pagos (pagos):** Registra los pagos asociados a las ventas de camiones, detallando el total pagado y la fecha de pago.
- **Detalles de Envíos:** Incluye información detallada de cada envío, como cantidad y peso de la mercancía.
- **Comprobación:** Vincula los registros de mantenimiento con los empleados responsables de realizar el mantenimiento

**2. Piensa en un escenario de base de datos. Escribe las especificaciones de diseño, tal como hicimos en el Tema 1. El modelo debe tener al menos 8 tablas relacionadas entre sí.**

Aquí hablaremos las tablas y sus especificaciones, empezando por la tabla de camión:

```
create table camiones (  
    id_camion int auto_increment primary key,  
    marca varchar(30) not null,  
    modelo varchar(30) not null,  
    ano_fabricacion date,  
    precio decimal(10, 2),  
    disponible char(1) not null default 'n'  
);
```

Esta tabla consiste por un **id\_camion** y su especificación de **auto\_increment**, este hará que su valor se incremente automáticamente cada vez que se inserta una nueva fila.

Además podemos ver los componentes de esta tabla camión empezando por marca que tiene **varchar(30)** esto significa que sus datos **no podrán sobrepasar 30 caracteres**, a continuación nos fijamos en la fila de disponible, esta es **Char(1)**, la cual significa que solo podrá tener un valor y **“not default ‘n’”** que significa que por defecto **aparecerá con la letra ‘N’**.

También deberemos de tener en cuenta la fila “ano\_fabricacion” ya que su clave es **date**, esto significa que sus valores serán **solo fechas** como por ejemplo: ‘2024-06-30’.

Por último deberemos fijarnos en la clave **Decimal (10,2)**, este hace normalmente referencia a la fila Precio, Salario, Pago etc...

*Continuamos con la tabla empleados:*

```
create table empleados (  
    id_empleado int auto_increment primary key,  
    nombre varchar(30) not null,  
    apellido varchar(30),  
    puesto varchar(30),  
    salario decimal(10, 2),  
    fecha_contratacion date  
);
```

Esta tabla consiste por un **id\_empleado** con la misma especificación que la anterior “auto\_increment”. Esta también cuenta con la clave **date** en “fecha\_contratacion”. Aquí podemos ver la fila “Puesto”, esta está constituida por los cargos de la empresa, los cuales veremos más adelante

*Continuamos con la tabla de cliente:*

```
create table clientes (  
    id_cliente int auto_increment primary key,  
    nombre varchar(30) not null,  
    apellidos varchar(30),  
    email varchar(50),  
    telefono char(10)  
);
```

Esta tabla consiste por un **id\_cliente** pero no tiene nada nuevo, en ella se hospedarán toda la información de los clientes los cuales hagan alguna compra en nuestra empresa, estas compras se registrarán en la tabla de ventas

*Continuamos con la tabla ventas:*

```
create table ventas (  
    id_venta int auto_increment primary key,  
    id_cliente int,  
    id_camion int,  
    fecha_venta date,  
    precio_venta decimal(10, 2),  
    foreign key (id_cliente) references clientes(id_cliente),  
    foreign key (id_camion) references camiones(id_camion)  
);
```

Esta tabla consiste por un **id\_venta** con el valor `auto_increment`, `id_cliente` haciendo referencia a la tabla de cliente a través de **foreign key** y **references**. Esto lo podemos ver también con `id_camion` haciendo referencia a la tabla `camion`. En esta tabla se registrarán todas las ventas

*Continuamos con la tabla Tipos de mercancía:*

```
create table tipos_mercancia (  
    id_mercancia int auto_increment primary key,  
    nombre varchar(30) not null  
);
```

Esta tabla consiste por un **id\_mercancia** con el valor “`auto_increment`” y con el nombre, aquí usaremos los nombre de los tipos de mercancía que podemos distribuir, en nuestro caso usaremos: Electrónicos, Material de construcción, Muebles, Productos químicos, Ropa)

Continuamos con la tabla envíos:

```
create table envios (  
    id_envio int auto_increment primary key,  
    id_camion int,  
    tipo_mercancia int,  
    origen varchar(40),  
    destino varchar(40),  
    fecha_envio date,  
    foreign key (id_camion) references camiones(id_camion),  
    foreign key (tipo_mercancia) references tipos_mercancia(id_mercancia)  
);
```

Esta tabla consiste por un **id\_envio** con el valor auto\_increment, una foreign key haciendo referencia a la tabla **camión** y otra haciendo referencia a la tabla **tipo\_mercancia** visto anteriormente. Esta tabla contará con todos los envíos realizados por camiones, añadiendo así su transportista y su destino

Continuamos con la tabla mantenimiento:

```
create table mantenimiento (  
    id_mantenimiento int auto_increment primary key,  
    id_camion int,  
    descripcion varchar(200),  
    precio decimal(10, 2),  
    fecha_mantenimiento date,  
    foreign key (id_camion) references camiones(id_camion)  
);
```

Esta tabla consiste por un **id\_mantenimiento** con el valor auto\_increment, una foreign key haciendo referencia a la tabla **camión**, podemos ver que en esta tabla existe una fila llamada Descripción con **varchar(200)**, este varchar funciona igual que varchar(30) solo que al ser una descripción necesitamos más espacio y he puesto 200 caracteres

Continuamos con la tabla Historia de pagos (pagos):

```
create table pagos (  
    id_pago int auto_increment primary key,  
    id_venta int,  
    total_pagado decimal(10, 2),  
    fecha_pago date,  
    foreign key (id_venta) references ventas(id_venta)  
);
```

En esta tabla he acertado el nombre, pero igualmente consiste en un **id\_pago** y con una serie de filas para los mismo, por único, recalcar que hace referencia a la tabla ventas a través de la **foreign key (id\_venta)**.

Continuamos con la tablas detalles de envios:

```
create table detalles_envios (  
    id_detalle int auto_increment primary key,  
    id_envio int,  
    cantidad int,  
    peso decimal(10, 2),  
    foreign key (id_envio) references envios(id_envio)  
);
```

Esta tabla consiste de un **id\_detalle**.

Continuamos con la tabla comprobación:

```
create table comprobacion (  
    id_comprobacion int auto_increment primary key,  
    id_mantenimiento int,  
    id_empleado int,  
    foreign key (id_mantenimiento) references mantenimiento(id_mantenimiento),  
    foreign key (id_empleado) references empleados(id_empleado)  
);
```

En la tabla de comprobación que está formada por un **id\_comprobacion** y dos referencias a las tablas de mantenimiento y empleado, en ella daremos por confirmada la comprobación del mantenimiento realizado, ya que muchas veces puede salir algún vehículo en mal estado.

### **3. Dibuja el modelo de BD acorde a las especificaciones del punto anterior.**

En las bases de datos nos podemos encontrar con 3 tipos de relaciones que son:

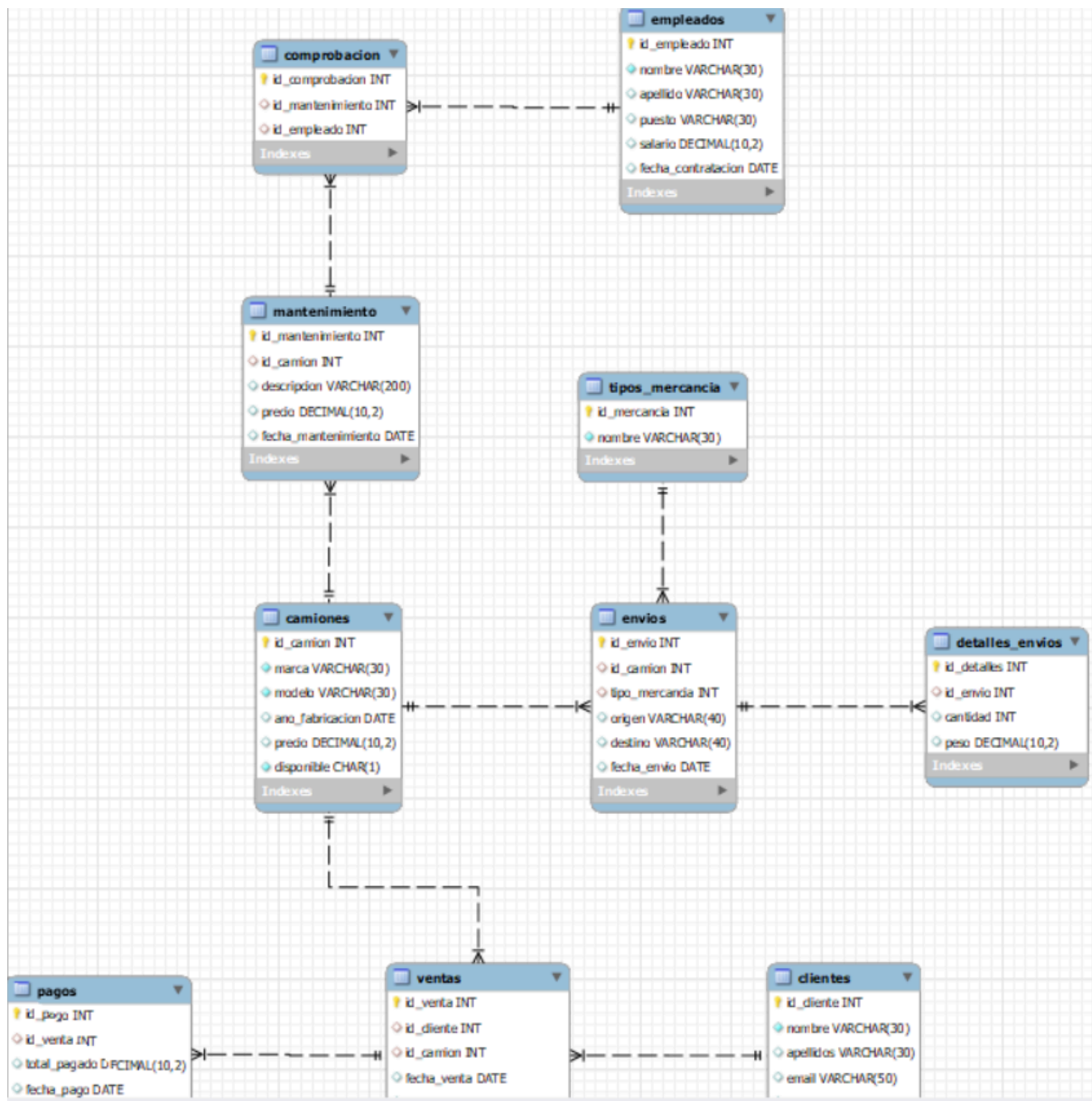
- 1. Relación de uno a uno:** Cada fila de una tabla está vinculada a una sola fila de otra tabla y viceversa
- 2. Relación de uno a muchos:** Una fila de una tabla puede estar relacionada con múltiples filas de otra tabla, pero una fila de la segunda tabla solo puede estar relacionada con una fila de la primera tabla.
- 3. Relación de muchos a muchos:** Las filas de una tabla pueden estar relacionadas con múltiples filas de otra tabla y viceversa

El modelo de esta base de datos consiste en un eje central que es: Camiones, dentro de ella nos encontramos con las siguientes relaciones

- **Relaciones de camiones:**
  1. Tiene una relación uno a muchos con la tabla de ventas.
  2. Tiene una relación uno a muchos con la tabla mantenimiento.
  3. Tiene una relación uno a muchos con la tabla envios.
  
- **Relaciones de ventas:**
  1. Tiene una relación muchos a uno con la tabla clientes.
  2. Tiene una relación muchos a uno con la tabla camiones..
  3. Tiene una relación uno a muchos con la tabla pagos
  
- **Relaciones de pagos:**
  1. Tiene una relación muchos a uno con la tabla de ventas.
  
- **Relaciones de clientes:**
  1. Tiene una relación uno a muchos con la tabla ventas.
  
- **Relaciones de envios:**
  1. Tiene una relación muchos a uno con la tabla camiones.
  2. Tiene una relación muchos a uno con la tabla tipos\_mercancia.
  3. Tiene una relación uno a muchos con la tabla detalles\_envios.

- **Relaciones de detalles\_envios:**
  1. Tiene una relación muchos a uno con la tabla envios.
  
- **Relaciones de tipos\_mercancia:**
  1. Tiene una relación uno a muchos con la tabla envios
  
- **Relaciones de mantenimiento:**
  1. Tiene una relación muchos a uno con la tabla camiones..
  2. Tiene una relación uno a muchos con la tabla comprobacion
  
- **Relaciones de comprobacion:**
  1. Tiene una relación muchos a uno con la tabla mantenimiento.
  2. Tiene una relación muchos a uno con la tabla empleados
  
- **Relaciones de empleados:**
  1. Tiene una relación uno a muchos con la tabla de comprobacion

Al terminar de unir nuestras tablas deberíamos de tener una imagen de la base de datos parecida a la siguiente:



Puede que no sea exactamente igual, pero no te preocupes ya que lo puedes ordenar de la forma más eficiente. Lo único que tenemos que tener igual son las tablas y sus relaciones



## 4. Crea la Base de Datos

### PASOS PARA CREAR UNA BASE DE DATOS MYSQL:

#### 1. PRIMER PASO:

Usando Workbench crearemos nuestro propio modelo de base de datos, el cual usaremos para comprobar las relaciones entre sí, como hemos visto en la imagen anterior

#### 2. SEGUNDO PASO:

Una vez creado el diseño entraremos en cmd y usaremos el comando:

- `sudo mysql -u root -p`

#### 3. TERCER PASO:

Crearemos un usuario con contraseña, o mejor dicho, el nombre de la actividad para realizar la actividad dentro de él, con el comando:

- `mysql> create user ejemplo1 identified by "ejemplo1" ;`
- `mysql> create user prueba1 identified by "camiones" ;`

#### 4. CUARTO PASO:

Aplicaremos privilegios al propio usuario

- `mysql> grant all privileges on ejemplo1.* to ejemplo1;`
- `mysql> grant all privileges on empresa_camiones.* to prueba1;`

#### 5. QUINTO PASO:

Una vez aplicado, entraremos dentro del usuario, para ello cerraremos la terminal y usaremos el comando

- `mysql -u empresa_camiones -p`

Pedirá la contraseña y usaremos la anteriormente aplicada (vehículos)

#### 6. SEXTO PASO:

Una vez dentro necesitaremos crear la base de datos con el comando:

- `create database empresa_camiones ;`

#### 7. SÉPTIMO PASO

Si no tenemos ningún error ya podremos usar la base de datos con el comando

- `use empresa_camiones`

**Una vez creada la base de datos, desde la misma terminal añadiremos los siguientes datos:**

```
create table empleados (  
    id_empleado int auto_increment primary key,  
    nombre varchar(30) not null,  
    apellido varchar(30),  
    puesto varchar(30),  
    salario decimal(10, 2),  
    fecha_contratacion date  
);
```

```
create table clientes (  
    id_cliente int auto_increment primary key,  
    nombre varchar(30) not null,  
    apellidos varchar(30),  
    email varchar(50),  
    telefono char(10)  
);
```

```
create table camiones (  
    id_camion int auto_increment primary key,  
    marca varchar(30) not null,  
    modelo varchar(30) not null,  
    ano_fabricacion date,  
    precio decimal(10, 2),  
    disponible char(1) not null default 'n'  
);
```

```
create table ventas (  
    id_venta int auto_increment primary key,  
    id_cliente int,  
    id_camion int,  
    fecha_venta date,  
    precio_venta decimal(10, 2),  
    foreign key (id_cliente) references clientes(id_cliente),  
    foreign key (id_camion) references camiones(id_camion)  
);
```

```
create table tipos_mercancia (  
    id_mercancia int auto_increment primary key,  
    nombre varchar(30) not null  
);
```

```

create table envios (
  id_envio int auto_increment primary key,
  id_camion int,
  tipo_mercancia int,
  origen varchar(40),
  destino varchar(40),
  fecha_envio date,
  foreign key (id_camion) references camiones(id_camion),
  foreign key (tipo_mercancia) references tipos_mercancia(id_mercancia)
);

```

```

create table mantenimiento (
  id_mantenimiento int auto_increment primary key,
  id_camion int,
  descripcion varchar(200),
  precio decimal(10, 2),
  fecha_mantenimiento date,
  foreign key (id_camion) references camiones(id_camion)
);

```

```

create table pagos (
  id_pago int auto_increment primary key,
  id_venta int,
  total_pagado decimal(10, 2),
  fecha_pago date,
  foreign key (id_venta) references ventas(id_venta)
);

```

```

create table detalles_envios (
  id_detalle int auto_increment primary key,
  id_envio int,
  cantidad int,
  peso decimal(10, 2),
  foreign key (id_envio) references envios(id_envio)
);

```

```

create table comprobacion (
  id_comprobacion int auto_increment primary key,
  id_mantenimiento int,
  id_empleado int,
  foreign key (id_mantenimiento) references mantenimiento(id_mantenimiento),
  foreign key (id_empleado) references empleados(id_empleado)
);

```

**Una vez copiados todos los datos, podremos entrar en Workbench para comprobar que la base de datos empresa\_camiones ya tiene todas las tablas creadas correctamente**

## 5. Inserta datos que aporten sentido a la Base de Datos

**Para ello copiaremos los siguientes datos y desde la misma terminal podemos pegarlos. Una vez pegados y comprobados que no haya ningún error, entraremos en Workbench para comprobar que los datos están correctos**

```
insert into empleados (nombre, apellido, puesto, salario, fecha_contratacion) values
('Juan', 'González', 'Chofer', 2500.00, '2023-01-15'),
('María', 'López', 'Mecánico', 2800.00, '2022-08-20'),
('Pedro', 'Martínez', 'Gerente', 3500.00, '2021-05-10'),
('Laura', 'Pérez', 'Recepcionista', 2000.00, '2023-03-01'),
('Carlos', 'García', 'Chofer', 2400.00, '2023-02-10');
```

```
insert into clientes (nombre, apellidos, email, telefono) values
('Ana', 'Ruiz', 'ana@gmail.com', 612345678),
('Luis', 'Sánchez', 'luis@gmail.com', 655432189),
('Elena', 'Gómez', 'elena@gmail.com', 699887766),
('Pablo', 'Díaz', 'pablo@gmail.com', 671234567),
('Sofía', 'Fernández', 'sofia@gmail.com', 688998877);
```

```
insert into camiones (marca, modelo, ano_fabricacion, precio, disponible) values
('Volvo', 'vnl 860', '2020-01-01', 120000.00, 'N'),
('Kenworth', 't680', '2019-05-15', 110000.00, 'N'),
('Freightliner', 'cascadia', '2021-03-10', 130000.00, 'N'),
('Peterbilt', '579', '2018-12-20', 105000.00, 'N'),
('Mack', 'anthem', '2022-02-28', 125000.00, 'N');
```

```
insert into ventas (id_cliente, id_camion, fecha_venta, precio_venta) values
(1, 1, '2024-04-10', 115000.00),
(3, 3, '2024-03-25', 125000.00),
(2, 4, '2024-02-15', 107000.00),
(4, 2, '2024-01-05', 112000.00),
(5, 5, '2024-04-01', 120000.00);
```

```
insert into tipos_mercancia (nombre) values
('Electrónicos'),
('Materiales de construcción'),
('Muebles'),
('Productos químicos'),
('Ropa');
```

```
insert into envios (id_camion, tipo_mercancia, origen, destino, fecha_envio) values
(1, 1, 'Madrid', 'Barcelona', '2024-04-20'),
(3, 3, 'Sevilla', 'Valencia', '2024-03-30'),
(2, 2, 'Bilbao', 'Alicante', '2024-02-25'),
(4, 4, 'Málaga', 'Zaragoza', '2024-01-15'),
(5, 5, 'Valencia', 'Valladolid', '2024-04-05');
```

```
insert into mantenimiento (id_camion, descripcion, precio, fecha_mantenimiento) values
(1, 'cambio de aceite y filtros', 200.00, '2024-03-01'),
(2, 'revisión del sistema de frenos', 300.00, '2024-02-10'),
(3, 'reparación de motor', 800.00, '2024-01-20'),
(4, 'alineación y balanceo', 150.00, '2024-04-05'),
(5, 'cambio de llantas', 600.00, '2024-02-28');
```

```
insert into pagos (id_venta, total_pagado, fecha_pago) values
(1, 115000.00, '2024-04-12'),
(3, 125000.00, '2024-03-28'),
(2, 107000.00, '2024-02-18'),
(4, 112000.00, '2024-01-08'),
(5, 120000.00, '2024-04-03');
```

```
insert into detalles_envios (id_envio, cantidad, peso) values
(1, 100, 500.00),
(3, 50, 300.00),
(2, 80, 700.00),
(4, 120, 1000.00),
(5, 200, 1500.00);
```

```
insert into comprobacion (id_mantenimiento, id_empleado) values
(1, 2),
(2, 3),
(3, 1),
(4, 4),
(5, 5);
```

## 6. Diseña varias consultas, que tendrás que describir y hacer:

### 1. Al menos 4 consultas simples

Esta consulta selecciona y devuelve todas las columnas y todas las filas de la tabla empleados

```
select * from empleados;
```

1 • `select * from empleados;`

2

#	id_empleado	nombre	apellido	puesto	salario	fecha_cont
1	1	Juan	González	Chofer	2500.00	2023-01-15
2	2	María	López	Mecánico	2800.00	2022-08-20
3	3	Pedro	Martínez	Gerente	3500.00	2021-05-10
4	4	Laura	Pérez	Recepcionista	2000.00	2023-03-01
5	5	Carlos	García	Chofer	2400.00	2023-02-10

Esta consulta selecciona las columnas id\_cliente, nombre, apellidos, email y telefono de la tabla clientes y con order by las ordena alfabéticamente

```
select id_cliente, nombre, apellidos, email, telefono
from clientes
order by nombre;
```

3 • `select id_cliente, nombre, apellidos, email, telefono`

4 `from clientes`

5 `order by nombre;`

#	id_cliente	nombre	apellidos	email	telefono
1	1	Ana	Ruiz	ana@gmail.com	612345678
2	3	Elena	Gómez	elena@gmail.com	699887766
3	2	Luis	Sánchez	luis@gmail.com	655432189
4	4	Pablo	Díaz	pablo@gmail.com	671234567
5	5	Sofía	Fernández	sofia@gmail.com	688998877

Esta consulta selecciona las columnas nombre, apellido, puesto y salario de la tabla empleados y filtra los resultados para incluir sólo aquellos empleados cuyo salario es **mayor a 2500**.

```
select nombre, apellido, puesto, salario
from empleados
where salario > 2500;
```

```
7 • select nombre, apellido, puesto, salario
8   from empleados
9   where salario > 2500;
```

Result Grid				
Filter Rows: <input type="text"/>				
Export: <input type="button" value="Export"/>				
#	nombre	apellido	puesto	salario
1	María	López	Mecánico	2800.00
2	Pedro	Martínez	Gerente	3500.00

Esta consulta cuenta el número total de registros en la tabla ventas y asigna el resultado a la columna total\_ventas. El resultado proporciona el número total de ventas realizadas y el valor total de todas esas ventas

```
select count(*) as total_ventas, sum(precio_venta) as valor_total_ventas
from ventas;
```

```
11 • select count(*) as total_ventas, sum(precio_venta) as valor_total_ventas
12   from ventas;
13
```

Result Grid		
Filter Rows: <input type="text"/>		
Export: <input type="button" value="Export"/>		
Wrap Cell Content: <input type="button" value="Wrap"/>		
#	total_ventas	valor_total_venta
1	5	579000.00

## 2. Al menos 4 consultas de agrupación

Esta consulta muestra cada marca y la cantidad de camiones correspondientes a cada una

```
select marca, count(*) as cantidad_de_camiones
from camiones
group by marca;
```

```
1 • select marca, count(*) as cantidad_de_camiones
2   from camiones
3   group by marca;
```

#	marca	cantidad_de_camione
1	Volvo	1
2	Kenworth	1
3	Freightliner	1
4	Peterbilt	1
5	Mack	1

Esta consulta muestra cada id\_cliente y el total de ventas realizadas a cada cliente

```
select id_cliente, sum(precio_venta) as total_ventas
from ventas
group by id_cliente;
```

```
5 • select id_cliente, sum(precio_venta) as total_ventas
6   from ventas
7   group by id_cliente;
```

#	id_cliente	total_ventas
1	1	115000.00
2	2	107000.00
3	3	125000.00
4	4	112000.00
5	5	120000.00



*Esta consulta muestra cada puesto y el salario promedio asociado a ese puesto*

```
select puesto, avg(salario) as salario_promedio
from empleados
group by puesto;
```

```
9 • select puesto, avg(salario) as salario_promedio
10 from empleados
11 group by puesto;
12
```

#	puesto	salario_promedio
1	Chofer	2450.000000
2	Mecánico	2800.000000
3	Gerente	3500.000000
4	Recepcionista	2000.000000

*Esta consulta muestra el id\_cliente y el total gastado por el cliente que más ha gastado.*

```
select id_cliente, sum(precio_venta) as total_gastado
from ventas
group by id_cliente
order by total_gastado desc
limit 1;
```

```
13 • select id_cliente, sum(precio_venta) as total_gastado
14 from ventas
15 group by id_cliente
16 order by total_gastado desc
17 limit 1;
```

#	puesto	salario_promedio
1	Chofer	2450.000000
2	Mecánico	2800.000000
3	Gerente	3500.000000
4	Recepcionista	2000.000000

*Esta consulta muestra el año de fabricación y el precio promedio de los camiones fabricados en ese año*

```
select year(ano_fabricacion) as año, avg(precio) as precio_promedio
from camiones
group by year(ano_fabricacion);
```

```
19 • select year(ano_fabricacion) as año, avg(precio) as precio_promedio
20   from camiones
21   group by year(ano_fabricacion);
22
```

#	año	precio_promedio
1	2020	120000.000000
2	2019	110000.000000
3	2021	130000.000000
4	2018	105000.000000
5	2022	125000.000000

### 3. Al menos 2 consultas OUTER JOIN

*Esta consulta muestra el nombre del tipo de mercancía, junto con el ID del envío, origen, destino y fecha de envío correspondientes. Si no hay envíos para un tipo de mercancía, los valores de las columnas de la tabla envíos serán NULL.*

```
select tm.nombre as tipo_mercancia, e.id_envio, e.origen, e.destino, e.fecha_envio
from tipos_mercancia tm
left join envios e on tm.id_mercancia = e.tipo_mercancia;
```

```
1 • select tm.nombre as tipo_mercancia, e.id_envio, e.origen, e.destino, e.fecha_envio
2   from tipos_mercancia tm
3   left join envios e on tm.id_mercancia = e.tipo_mercancia;
```

#	tipo_mercancia	id_envio	origen	destino	fecha_envio
1	Electrónicos	1	Madrid	Barcelona	2024-04-20
2	Materiales de construcción	3	Bilbao	Alicante	2024-02-25
3	Muebles	2	Sevilla	Valencia	2024-03-30
4	Productos químicos	4	Málaga	Zaragoza	2024-01-15
5	Ropa	5	Valencia	Valladolid	2024-04-05

*Esta consulta muestra el nombre de cada cliente junto con el total de ventas realizadas a cada uno*

```
select c.nombre, sum(v.precio_venta) as total_ventas
from ventas v
inner join clientes c on v.id_cliente = c.id_cliente
group by c.nombre;
```

```
5 • select c.nombre, sum(v.precio_venta) as total_ventas
6   from ventas v
7   inner join clientes c on v.id_cliente = c.id_cliente
8   group by c.nombre;
9
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
#	nombre	total_ventas			
1	Ana	115000.00			
2	Elena	125000.00			
3	Luis	107000.00			
4	Pablo	112000.00			
5	Sofía	120000.00			

*Esta consulta muestra el modelo de cada camión junto con el precio total de mantenimiento asociado a cada uno*

```
select c.modelo, sum(m.precio) as precio_total_mantenimiento
from camiones c
left join mantenimiento m on c.id_camion = m.id_camion
group by c.modelo;
```

```
10 • select c.modelo, sum(m.precio) as precio_total_mantenimiento
11   from camiones c
12   left join mantenimiento m on c.id_camion = m.id_camion
13   group by c.modelo;
14
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
#	modelo	precio_total_mantenimien			
1	VNL 860	200.00			
2	T680	300.00			
3	Cascadia	800.00			
4	579	150.00			
5	Anthem	600.00			

#### 4. Al menos 2 consultas con subconsultas

Esta consulta devolverá todos los empleados que tienen el salario máximo en la tabla de empleados.

```
select *
from empleados
where salario = (select max(salario) from empleados);
```

```
1 • select *
2   from empleados
3   where salario = (select max(salario) from empleados);
```

Result Grid

Filter Rows:

Edit:

Export/Im

#	id_emplead	c nombre	apellido	puesto	salario	fecha_contratacio
1	3	Pedro	Martínez	Gerente	3500.00	2021-05-10

Esta consulta devolverá todos los detalles de envío cuya cantidad sea mayor que el promedio de todas las cantidades de envío en la tabla detalles\_envios.

```
select id_detalle, id_envio, cantidad, peso
from detalles_envios
where cantidad >
    ( select avg(cantidad)
      from detalles_envios
    );
```

```
5 • select *
6   from detalles_envios
7   where cantidad >
8     ( select avg(cantidad)
9       from detalles_envios
10     );
```

Result Grid




Filter Rows:

#	id_detalle	id_envio	cantidad	peso
1	4	4	120	1000.00
2	5	5	200	1500.00
*	NULL	NULL	NULL	NULL

*Esta consulta devolverá todos los camiones cuyo precio sea igual al precio máximo de camiones de la misma marca*

```
select *
from camiones c
where precio = (
    select max(precio)
    from camiones
    where marca = c.marca
);
```

```
13 • select *
14   from camiones c
15   where precio = (
16       select max(precio)
17       from camiones
18       where marca = c.marca
19   );
20
```

Result Grid		Filter Rows: <input type="text"/>			Edit:   		Export
#	id_camion	marca	modelo	ano_fabricacion	precio	disponible	
1	1	Volvo	VNL 860	2020-01-01	120000.00	N	
2	2	Kenworth	T680	2019-05-15	110000.00	N	
3	3	Freightliner	Cascadia	2021-03-10	130000.00	N	
4	4	Peterbilt	579	2018-12-20	105000.00	N	
5	5	Mack	Anthem	2022-02-28	125000.00	N	

## 7. Describe y crea 5 funciones y/o procedimientos (al menos uno de cada)

La función llamada `realizar_venta`, se utiliza para simular la realización de ventas por parte de los clientes.

**function realizar\_venta(a\_id\_cliente int, a\_id\_camion int, a\_fecha\_venta date, a\_precio\_venta decimal(10, 2)) returns int**

drop function if exists realizar\_venta;

delimiter //

create function realizar\_venta(a\_id\_cliente int, a\_id\_camion int, a\_fecha\_venta date, a\_precio\_venta decimal(10, 2)) returns int

reads sql data

begin

declare v\_venta int;

insert into ventas (id\_cliente, id\_camion, fecha\_venta, precio\_venta)

values (a\_id\_cliente, a\_id\_camion, a\_fecha\_venta, a\_precio\_venta);




set v\_venta = last\_insert\_id();

return v\_venta;

end //

delimiter ;

```
1 • select realizar_venta(1, 2, '2024-05-14', 45.99);
```

Result Grid   Filter Rows:  Export:  Wrap Cell Con

#	realizar_venta(1, 2, '2024-05-14', 45.99)
1	6

```
3 • select * from ventas ;
```

Result Grid		Filter Rows:			Edit:
#	id_venta	id_cliente	id_camion	fecha_venta	precio_ven
1	1	1	1	2024-04-10	115000.00
2	2	3	3	2024-03-25	125000.00
3	3	2	4	2024-02-15	107000.00
4	4	4	2	2024-01-05	112000.00
5	5	5	5	2024-04-01	120000.00
6	6	1	2	2024-05-14	45.99

La funcion `f_realizar_envio` sirve para poder hacer la simulación de como se realizaria un envio, aplicando el camion, el tipo de mercancía, origen, destino y su fecha

**funcion `f_realizar_envio(a_id_camion int, a_tipo_mercancia int, a_origen varchar(40), a_destino varchar(40), a_fecha_envio date ) returns int`**

`drop function if exists f_realizar_envio;`

`delimiter //`

`create function f_realizar_envio(a_id_camion int, a_tipo_mercancia int, a_origen varchar(40), a_destino varchar(40), a_fecha_envio date ) returns int`

`reads sql data`

`begin`

`declare v_envio int;`

`insert into envios (id_camion, tipo_mercancia, origen, destino, fecha_envio)`

`values (a_id_camion, a_tipo_mercancia, a_origen, a_destino, a_fecha_envio);`

`set v_envio = last_insert_id();`

`return v_envio;`

`end //`

`delimiter ;`

```
select f_realizar_envio(1, 3, 'Madrid', 'Barcelona', '2024-05-17');
```

6 • `select * from envios;`

	id_envio	id_camion	tipo_mercancia	origen	destino	fecha_envio
▶	1	1	1	Madrid	Barcelona	2024-04-20
	2	3	3	Sevilla	Valencia	2024-03-30
	3	2	2	Bilbao	Alicante	2024-02-25
	4	4	4	Málaga	Zaragoza	2024-01-15
	5	5	5	Valencia	Valladolid	2024-04-05
	6	1	3	Madrid	Barcelona	2024-05-17
	NULL	NULL	NULL	NULL	NULL	NULL

*Esta función devuelve el precio total de mantenimiento asociado a ese camión*

**funcion f\_mantenimiento(a\_id\_camion int) returns decimal(10, 2)**

drop function if exists f\_mantenimiento;

delimiter //

create function f\_mantenimiento(a\_id\_camion int) returns decimal(10, 2)

reads sql data

begin

declare total\_mantenimiento decimal(10, 2);

select sum(precio) into total\_mantenimiento

from mantenimiento

where id\_camion = a\_id\_camion;

if total\_mantenimiento is null then

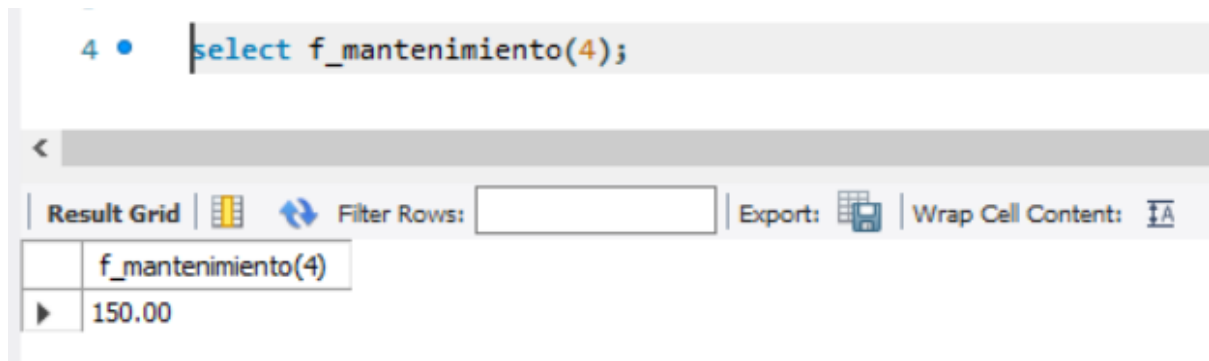
set total\_mantenimiento = 0;

end if;

return total\_mantenimiento;

end //

delimiter ;





*Esta función devuelve el salario promedio de cada empleado*

**funcion f\_salario\_promedio(a\_id\_empleado int) returns decimal(10, 2)**

drop function if exists f\_salario\_promedio;

delimiter //

create function f\_salario\_promedio(a\_id\_empleado int) returns decimal(10, 2)

reads sql data

begin

declare salario\_promedio decimal(10, 2);

select avg(salario) into salario\_promedio

from empleados

where id\_empleado = a\_id\_empleado;

if salario\_promedio is null then

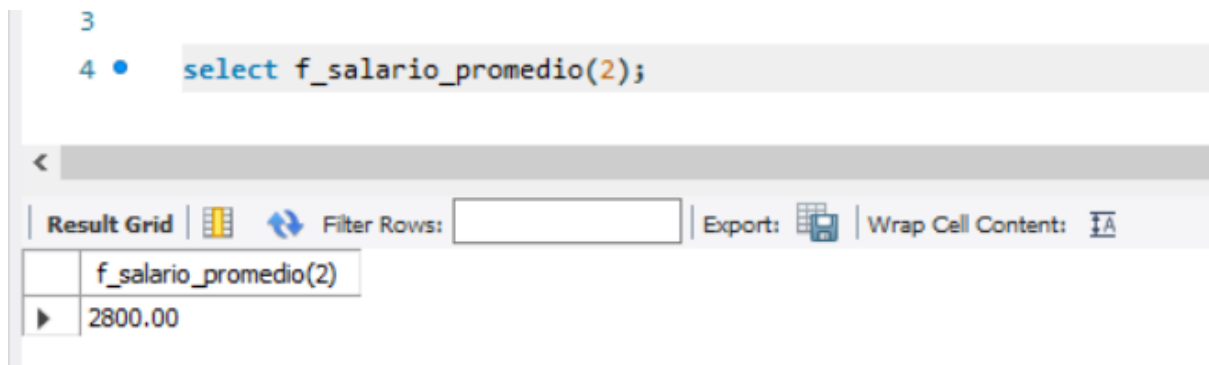
set salario\_promedio = 0;

end if;

return salario\_promedio;

end //

delimiter ;



The screenshot shows a SQL IDE interface. At the top, a query is entered in a text editor: `select f_salario_promedio(2);`. Below the editor, a toolbar contains icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. The 'Result Grid' is active, displaying a single row with the value '2800.00' under the column header 'f\_salario\_promedio(2)'.

f_salario_promedio(2)
2800.00

La entrega final será:

- La memoria, en formato WORD, WRITER o PDF, que tenga (al menos) los siguientes puntos:
  - Introducción (explicas el motivo de la base de datos)
  - Modelo de datos: explicas las entidades y relaciones, las claves primarias que eliges, las columnas, etc... Incluye el modelo de datos diseñado acorde a las especificaciones.
  - Creación de la Base de Datos: incluye el SQL así como los INSERT que hagan falta para nutrir la BD con información necesaria.
  - Consultas: describe las consultas, escribe el SQL y haz una muestra del resultado de cada una de ellas.
  - Funciones/procedimientos: describe su funcionamiento y escribe el SQL y realiza y documenta las pruebas de funcionamiento realizadas.
- El fichero MWB del Workbench
- Un MYSQL-DUMP de la BD diseñada (recuerda usar --routines)