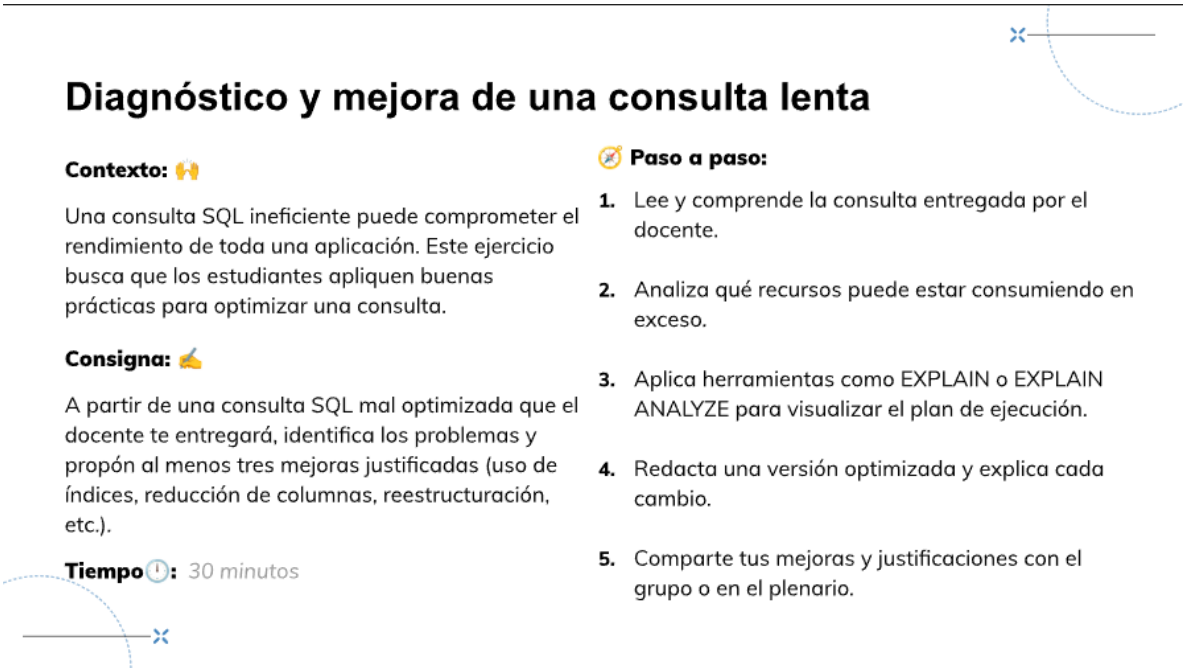


Consulta SQL original (ineficiente):

```
SELECT *
FROM empleados
WHERE UPPER(departamento) = 'VENTAS'
  AND salario > 50000;
```

Problemas identificados:

1. Uso de SELECT * trae todas las columnas innecesariamente.
2. Uso de UPPER() en la cláusula WHERE impide el uso de índice en departamento.
3. Falta de índices en columnas filtradas (departamento, salario).
4. No hay límite de resultados (peligroso si la tabla es grande).

Análisis de recursos:

- CPU: se desperdicia procesando todas las columnas con SELECT *.
- Disco: más I/O por leer toda la tabla.
- Plan de ejecución (con EXPLAIN): indica secuential scan, no se usan índices.

Mejora 1: evitar SELECT *

```
SELECT nombre, apellido, salario
FROM empleados
WHERE UPPER(departamento) = 'VENTAS'
  AND salario > 50000;
```

Mejora 2: evitar funciones en filtros

- Asumimos que los datos están normalizados (departamento en mayúsculas):

```
SELECT nombre, apellido, salario
FROM empleados
WHERE departamento = 'VENTAS'
  AND salario > 50000;
```

Mejora 3: uso de índices

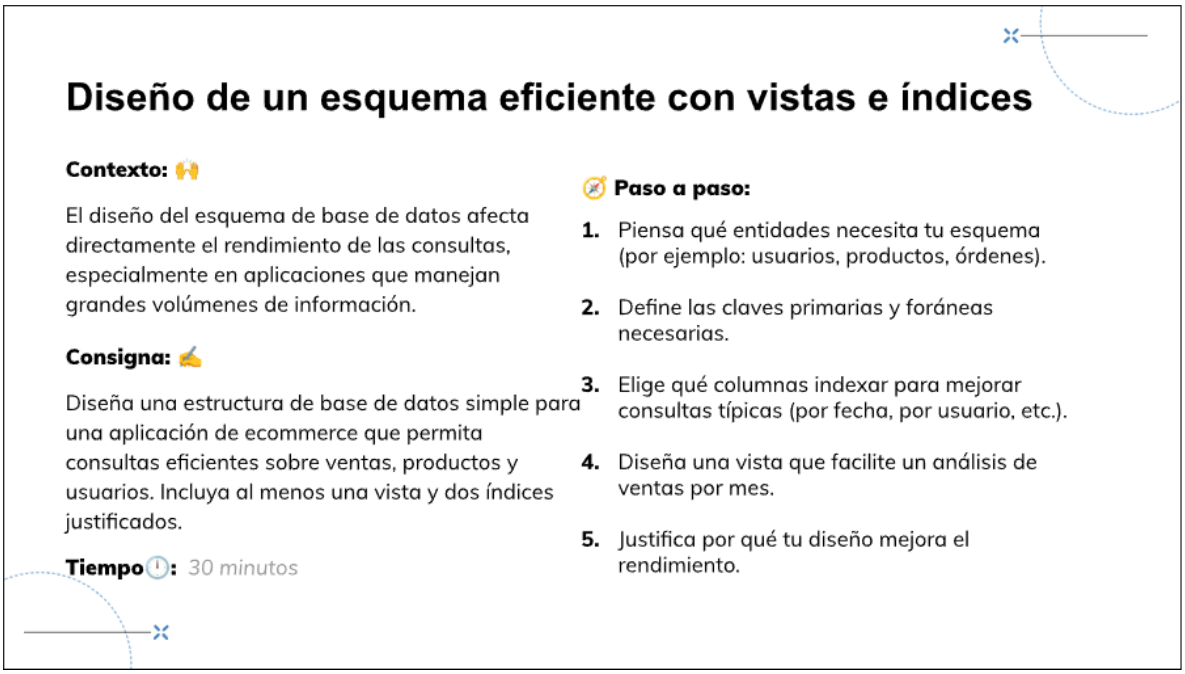
```
CREATE INDEX idx_departamento ON empleados(departamento);
CREATE INDEX idx_salario ON empleados(salario);
```

Resultado final optimizado:

```
SELECT nombre, apellido, salario
FROM empleados
WHERE departamento = 'VENTAS'
  AND salario > 50000;
```

Beneficios:

- Menos columnas = menor carga.
- Se pueden usar índices.
- Mejora el plan de ejecución y reduce el tiempo de respuesta.



1. Entidades necesarias:

- Usuarios
- Productos
- Órdenes
- Detalle_Orden (tabla intermedia que relaciona productos y órdenes)

2. Claves primarias y foráneas:

```
-- Tabla Usuarios
CREATE TABLE Usuarios (
  id_usuario INT PRIMARY KEY,
  nombre VARCHAR(100),
  correo VARCHAR(100)
);

-- Tabla Productos
CREATE TABLE Productos (
  id_producto INT PRIMARY KEY,
  nombre VARCHAR(100),
  precio DECIMAL(10, 2)
);
```

```
-- Tabla Órdenes
CREATE TABLE Ordenes (
  id_orden INT PRIMARY KEY,
  id_usuario INT,
  fecha TIMESTAMP,
  FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario)
);

-- Tabla Detalle_Orden
CREATE TABLE Detalle_Orden (
  id_detalle INT PRIMARY KEY,
  id_orden INT,
  id_producto INT,
  cantidad INT,
  FOREIGN KEY (id_orden) REFERENCES Ordenes(id_orden),
  FOREIGN KEY (id_producto) REFERENCES Productos(id_producto)
);
```

3. Índices justificados:

1. CREATE INDEX idx_fecha_orden ON Ordenes(fecha);
   - **Justificación:** mejora las consultas por rangos de fechas (por mes, año, etc.).
2. CREATE INDEX idx_usuario_orden ON Ordenes(id_usuario);
   - **Justificación:** mejora las búsquedas de órdenes por cliente/usuario.

4. Vista para análisis de ventas por mes:

```
CREATE VIEW Ventas_Mensuales AS
SELECT
  DATE_TRUNC('month', o.fecha) AS mes,
  SUM(p.precio * d.cantidad) AS total_ventas
FROM
  Ordenes o
JOIN Detalle_Orden d ON o.id_orden = d.id_orden
JOIN Productos p ON d.id_producto = p.id_producto
GROUP BY mes
ORDER BY mes;
```

5. Justificación del diseño:

- El modelo relacional está normalizado, lo que evita redundancias.
- Los índices aceleran las consultas típicas (por usuario y fecha).
- La vista permite consultar rápidamente las ventas mensuales sin repetir lógica en múltiples queries.
- El diseño mejora el rendimiento en grandes volúmenes de datos gracias al uso de claves, relaciones claras y consultas agregadas optimizadas.