1. Introducción

En este trabajo se diseñó una base de datos para un sistema de gestión de hospital.
El objetivo es poder administrar pacientes, médicos y turnos, y que la información quede organizada sin datos repetidos ni inconsistencias.

2. Entidades del sistema

A partir del enunciado, se identificaron tres entidades principales:

Paciente

Guarda la información personal de cada paciente.
Atributos principales: id_paciente (PK), nombre, apellido, dni, fecha_nacimiento, teléfono, email y fecha_alta.

Medico

Contiene los datos de los doctores del hospital.
Atributos: id_medico (PK), nombre, apellido, dni, especialidad, teléfono, email y fecha_alta.

Turno

Registra los turnos programados entre pacientes y médicos.
Atributos: id_turno (PK), fecha, hora, estado, motivo y fecha_creacion.

3. Relaciones

A partir de estas entidades se definieron dos relaciones:

1. **Paciente – Turno**: un paciente puede tener varios turnos.
   → Relación 1 a N.
2. **Medico – Turno**: un médico también puede atender muchos turnos.
   → Relación 1 a N.

Cada turno queda así asociado a un único médico y un único paciente.

4. Normalización

El diseño se llevó a 3NF para evitar redundancia y problemas de actualización.

- **1NF**: todos los atributos son simples, no hay listas ni valores repetidos.
- **2NF**: todas las tablas tienen clave primaria simple, así que no hay dependencias parciales.
- **3NF**: no hay atributos que dependan de otros atributos que no son clave.
  Por ejemplo, la especialidad depende solo del médico, no de otro atributo.

5. Restricciones e integridad

- Las PK son autoincrementales para simplificar la gestión.
- Los DNI tanto de pacientes como de médicos son **UNIQUE** para evitar duplicados.
- Las relaciones se implementaron con claves foráneas en la tabla turnos.
- Para no perder historial, se usó **ON DELETE RESTRICT** al borrar pacientes o médicos.
- También se agregó una restricción UNIQUE en (id_medico, fecha, hora) para evitar superposición de turnos del mismo médico.

6. Índices

Se agregaron índices en:

- **especialidad** (tabla médico)
- **fecha** (tabla turno)

Esto mejora las búsquedas y los reportes que se piden en el proyecto.

7. Procedimientos y funciones

Se incluyeron:

- Un procedimiento para cancelar turnos por rango de fechas.
- Un procedimiento para obtener los 3 médicos con más turnos.
- Una función que devuelve la cantidad de turnos de un médico.

Estos elementos resuelven las operaciones más complejas del sistema y cumplen con los requisitos del TP.

8. Conclusión

El diseño final permite manejar correctamente pacientes, médicos y turnos.
La base de datos queda normalizada, con restricciones adecuadas, sin datos repetidos y preparada para consultas, reportes y operaciones desde la aplicación en Python.