**Tarea 1: Creación de Modelos con SQLAlchemy**

**Curso:** Data Science and Machine Learning Applied to Financial Markets – Módulo III  
**Nombre:** Andrés Padrón Quintana  
**Fecha:** 13 de octubre de 2025

**Diseño del esquema de base de datos**

**1. Estudiante**

* **Atributos:**
  + id: clave primaria autoincremental.
  + nombre: obligatorio, máximo 100 caracteres.
  + email: obligatorio, único, con formato válido.
  + matricula: obligatorio, único, exactamente 8 caracteres alfanuméricos.
  + fecha\_inscripcion: fecha obligatoria.
* **Restricciones:**
  + Emails con formato validado mediante expresión regular.
  + Matrícula con longitud fija (8 caracteres).
  + Unicidad en email y matricula.

**2. Profesor**

* **Atributos:**
  + id: clave primaria.
  + nombre: obligatorio, máximo 100 caracteres.
  + email: único, obligatorio, validado.
  + departamento: obligatorio, máximo 50 caracteres.
* **Relaciones:**
  + Relación **1–N**: un profesor puede impartir varios cursos.

**3. Curso**

* **Atributos:**
  + id: clave primaria.
  + nombre: obligatorio, máximo 80 caracteres.
  + creditos: entero, restringido entre 1 y 10.
  + nivel: restringido a los valores ‘Licenciatura’, ‘Maestría’, ‘Doctorado’.
  + profesor\_id: clave foránea hacia Profesor.
* **Restricciones:**
  + CheckConstraint para validar créditos (1 ≤ créditos ≤ 10).
  + CheckConstraint para validar el nivel académico.
* **Relaciones:**
  + Cada curso pertenece a un profesor (relación inversa de 1–N).

**4. Inscripción (relación N–M)**

* **Atributos:**
  + id: clave primaria.
  + estudiante\_id: clave foránea hacia Estudiante.
  + curso\_id: clave foránea hacia Curso.
  + fecha\_inscripcion: fecha obligatoria.
  + calificacion: entero entre 0 y 100 (puede ser NULL si aún no hay nota).
* **Restricciones:**
  + Un estudiante no puede inscribirse dos veces en el mismo curso (UniqueConstraint).
  + Calificación debe estar en el rango [0, 100] (CheckConstraint).
* **Relaciones:**
  + Relación **N–M**:
    - Un estudiante puede tener varios cursos.
    - Un curso puede tener varios estudiantes.

**Creación de la base de datos**

Al final del script se ejecuta:

engine = create\_engine("sqlite:///universidad.db")

Base.metadata.create\_all(engine)

Esto genera un archivo universidad.db en la carpeta del proyecto.

* Si no existe, se crea automáticamente.
* Todas las tablas y restricciones definidas en las clases se traducen en un esquema físico dentro de la base de datos.

**Conclusión**

* El modelo diseñado en SQLAlchemy representa una estructura robusta y coherente para la gestión de información académica dentro de una institución universitaria. Cada tabla fue construida considerando la **integridad referencial**, las **restricciones de dominio** y las **relaciones entre entidades** que garantizan la consistencia de los datos.
* La implementación de validaciones mediante CheckConstraint, UniqueConstraint y el uso de decoradores @validates permite controlar desde el propio código los posibles errores de captura o duplicación de registros, asegurando así la **calidad y confiabilidad de la base de datos**.
* Además, la inclusión de relaciones 1–N (Profesor–Curso) y N–M (Estudiante–Curso) permite reflejar de manera realista el funcionamiento académico, donde los profesores imparten múltiples cursos y los estudiantes pueden inscribirse en varios de ellos.