Final Project Phase II

University of Nevada, Reno
Department of Computer Science and Engineering
CS 457 Database Management Systems

Team 02:
Connor James, Sasha Koroleva,
Robb Northrup, Ethan Partain,
Farzana Tanni

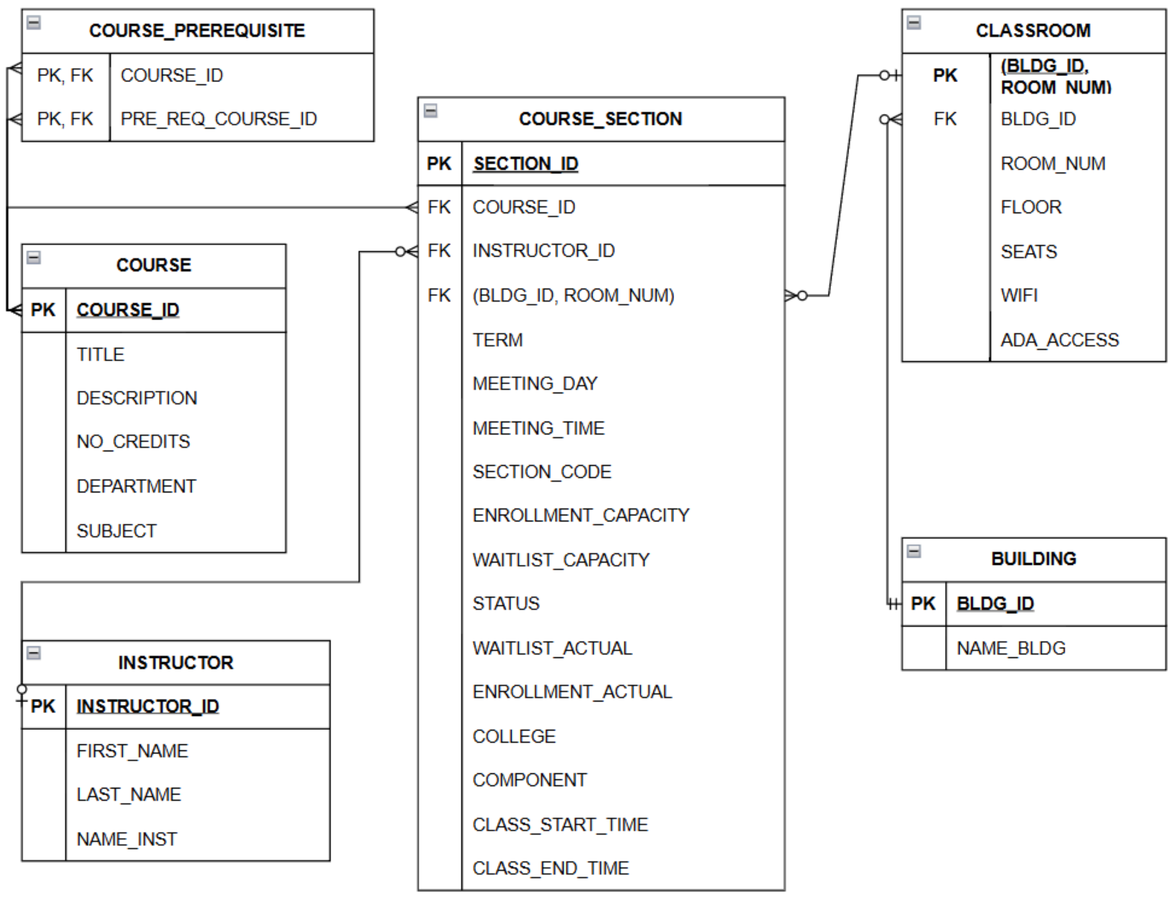
Instructors:
Jordan T. Hastings, Professor
Muhammed Ayaan, TA

May 12, 2025

Database Efficiency

The existing class search database is outdated and no longer meets the functional requirements of modern academic systems. This project focuses on developing a replacement by designing a well-structured and efficient database. The design process utilized both an Entity-Relationship Diagram (ERD) and a detailed glossary to establish a solid framework for the system. The ERD illustrates a relational model that promotes efficient data storage while addressing the limitations commonly found in non-normalized databases, such as data redundancy and update anomalies. The glossary serves as a quick reference guide, providing clear definitions for each attribute within the system. While the current ERD successfully outlines key relationships, further normalization could reduce attribute redundancy and enhance the overall design efficiency.

ERD



SLN notation

Courses
CourseID: TEXT, PRIMARY KEY
Title: TEXT, NOT NULL
Description: TEXT
No_Credits: INTEGER, NOT NULL
Department: TEXT
Subject: TEXT

Course_Prerequisites
CourseID: TEXT, NOT NULL, FOREIGN KEY => Courses.CourseID
Pre_Req_Course_ID: TEXT, NOT NULL, FOREIGN KEY => Courses.CourseID
PRIMARY KEY (Course_ID, Pre_Req_COurse_ID)

Course Sections
Section_ID: TEXT, PRIMARY KEY
Course_ID: TEXT, NOT NULL, FOREIGN KEY => Courses.CourseID
Instructor_ID: TEXT, FOREIGN KEY => Instructors.InstructorID
Bldg_ID: TEXT, NOT NULL
Room_Num: TEXT, NOT NULL
Term: TEXT, NOT NULL
College: TEXT
Component: TEXT
Meeting_Day: TEXT, NOT NULL
Meeting_Time: TEXT, NOT NULL
Section_Code: TEXT, NOT NULL
Enrollment_Capacity: INTEGER, NOT NULL
Waitlist_Capacity: INTEGER, NOT NULL
Status: TEXT, DEFAULT 'Open'
Waitlist_Actual: INTEGER, DEFAULT 0
Enrollment_Actual: INTEGER, DEFAULT 0
FOREIGN KEY (Bldg_ID, Room_Num)
REFERENCES Classrooms (Bldg_ID, Room_Num)

Instructors
InstructorID: TEXT, PRIMARY KEY
First_Name: TEXT
Last_Name: TEXT
Name_Inst: TEXT

Classrooms

Bldg_ID: TEXT, NOT NULL, FOREIGN KEY => Buildings.Bldg_ID
Room_Num: TEXT, NOT NULL
Floor: INTEGER, NOT NULL
Seats: INTEGER, NOT NULL
WiFi: TEXT, DEFAULT ‘No’
ADA_Access: TEXT, DEFAULT ‘No’
PRIMARY KEY (Bldg_ID, Room_Num)

Buildings
BldgID: TEXT, PRIMARY KEY
Name_Bldg: TEXT, NOT NULL

SLN Glossary

Attribute	Definition
CourseID	Unique numeric identifier for each course.
Title	Full course title as displayed in the catalog.
Description	Brief summary of the course content.
No_Credits	Number of credit hours assigned to the course.
Department	Academic department offering the course.
Subject	Subject code portion of the catalog number (e.g. “CS”).
Course_ID(in Course_Prerequisites)	Foreign key referencing the dependent course in the Course table.
Pre_Req_Course_ID	Foreign key reference to the Courses table (the prerequisite course).
Section_ID	Unique numeric identifier for each section.
Course_ID(in Course_Section)	Foreign key referencing the catalog course this section belongs to.
Instructor_ID	Foreign key reference to the Instructors table.
Bldg_ID	Foreign key referencing the building where the classroom is located.
Room_Num	Room number within the building.

Term	Academic term when the section is offered.
College	College or division offering the section.
Component	Course component type, (e.g. “Lecture, “Lab”).
Meeting_Day	Days of the week the class meets.
Meeting_Time	Time of day the class meets.
Class_Start_Time	Scheduled meeting start time (e.g. “09:00).
Class_End_Time	Scheduled meeting end time (e.g. “09:50).
Section_Code	Section code as assigned by the registrar.
Enrollment_Capacity	Maximum number of students allowed to enroll.
Waitlist_Capacity	Maximum number of students allowed on the waitlist.
Status	Current enrollment status.
Waitlist_Actual	Current number of students on the waitlist.
Enrollment_Actual	Current number of students enrolled.
Instructor_ID (in Instructors)	Unique text identifier for each instructor.
First_Name	Instructor’s given name.
Last_Name	Instructor’s family name.
Name_Inst	Concatenated instructor name for display.
Bldg_ID (in Buildings)	Unique text identifier for each building (e.g. “SEM).
Name_Bldg	Official name of the building.
Floor	Floor number where the classroom is located.
Seats	Number of available seats in the classroom.
WiFi	Indicates if WiFi is available.
ADA_Access	Indicates if the classroom is ADA accessible.

ETL Overview

Our ETL was implemented as a Python process that is designed from the ground up to convert the data of the raw class schedules from an excel file (form the UNR Registrar) to a well structured, cleaned format for being inserted into the SQLite database. The program utilizes the pandas library to parse the datasheets. Once this step is complete, the data is cleaned and normalized: fields split or merged to match database schema, mapping codes to their names, missing values are identified and either flagged or corrected, and tables are decomposed into their respective pieces: courses, section, instructor, buildings, classrooms, etc. The data is finally loaded into the SQLite database using SQL INSERT statements.

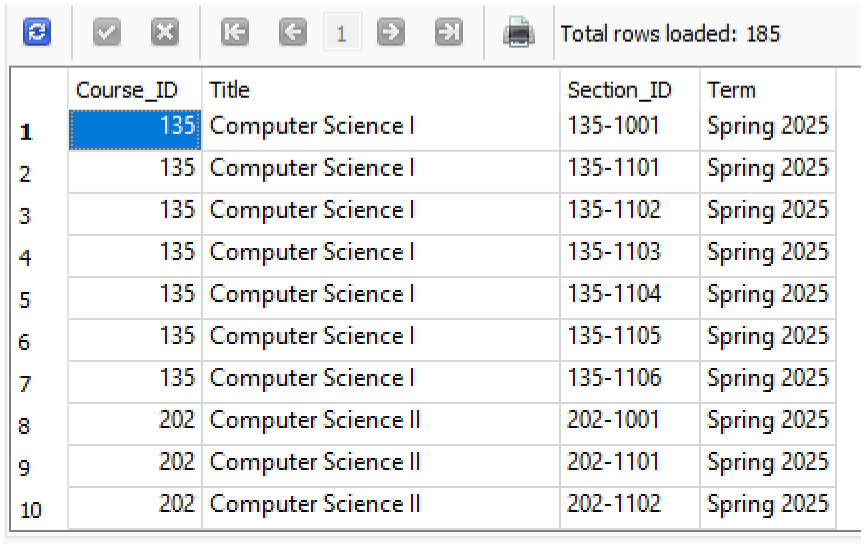
Just as well, to keep the data on the database up to date with the edited information on the excel file, the database is auto-updated by the ETL.py program being run in the background. This is accomplished by utilizing a Bash script to automatically run the entire process and ensure that the database properly reflects latest class offerings.

Database Queries

With the database fully designed, created, and populated, the final step involved validating its functionality through a series of structured SQL queries. These queries were carefully crafted to test the integrity of the database design, ensuring that all tables, relationships, and constraints functioned as intended. Additionally, the ETL process successfully transformed and prepared the raw data for seamless integration, resulting in accurate and organized data loading. The executed queries produced the expected results, confirming that the database meets the project’s design specifications and is capable of supporting reliable and efficient data retrieval for class search operations.

1. All classes/sections offered in Spring

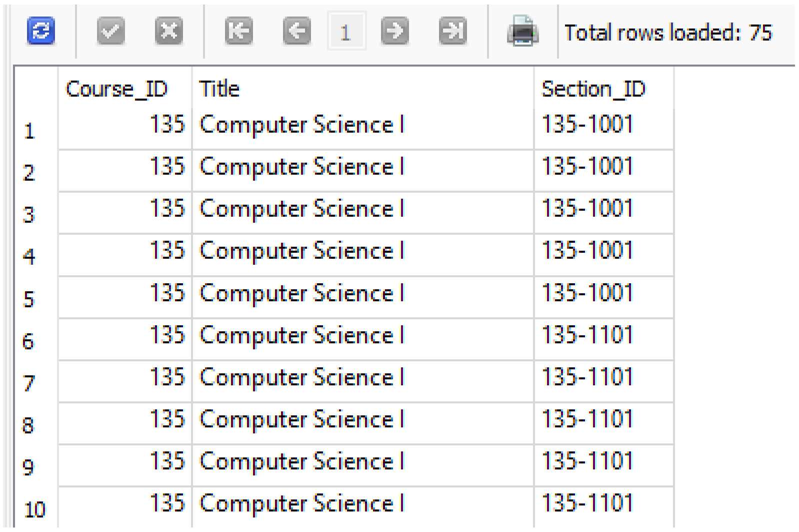
```
SELECT c.Course_ID, c.Title, cs.Section_ID, cs.Term
FROM Course c
JOIN Course_Section cs ON c.Course_ID = cs.Course_ID
WHERE cs.Term = 'Spring 2025';
```


Total rows loaded: 185

	Course_ID	Title	Section_ID	Term
1	135	Computer Science I	135-1001	Spring 2025
2	135	Computer Science I	135-1101	Spring 2025
3	135	Computer Science I	135-1102	Spring 2025
4	135	Computer Science I	135-1103	Spring 2025
5	135	Computer Science I	135-1104	Spring 2025
6	135	Computer Science I	135-1105	Spring 2025
7	135	Computer Science I	135-1106	Spring 2025
8	202	Computer Science II	202-1001	Spring 2025
9	202	Computer Science II	202-1101	Spring 2025
10	202	Computer Science II	202-1102	Spring 2025

2. All lower-division classes/sections

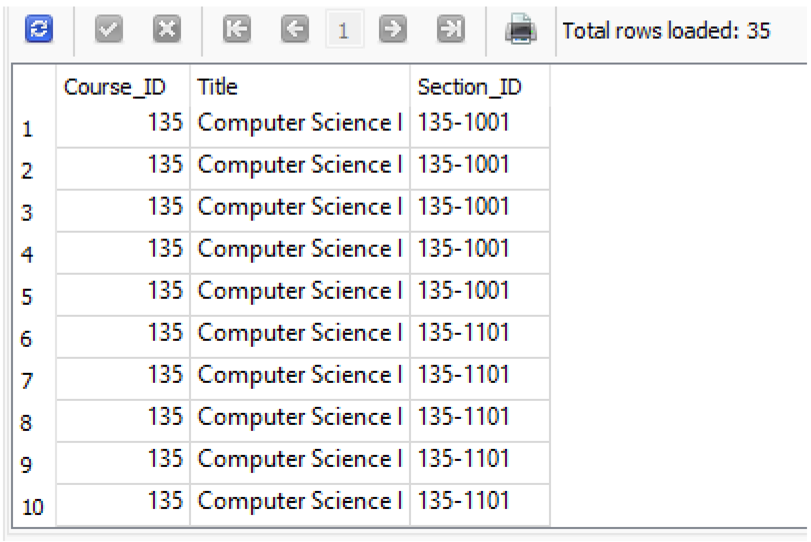
```
SELECT c.Course_ID, c.Title, cs.Section_ID
FROM Course c
JOIN Course_Section cs ON c.Course_ID = cs.Course_ID
WHERE CAST(c.Course_ID AS INTEGER) < 300;
```


Total rows loaded: 75

	Course_ID	Title	Section_ID
1	135	Computer Science I	135-1001
2	135	Computer Science I	135-1001
3	135	Computer Science I	135-1001
4	135	Computer Science I	135-1001
5	135	Computer Science I	135-1001
6	135	Computer Science I	135-1101
7	135	Computer Science I	135-1101
8	135	Computer Science I	135-1101
9	135	Computer Science I	135-1101
10	135	Computer Science I	135-1101

3. All CS135-related lectures & labs

```
SELECT c.Course_ID, c.Title, cs.Section_ID
FROM Course c
JOIN Course_Section cs ON c.Course_ID = cs.Course_ID
WHERE c.Course_ID = '135';
```



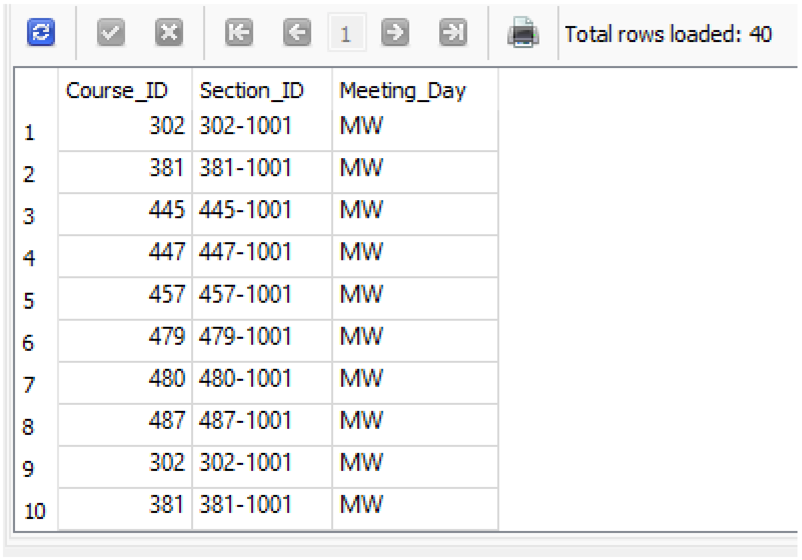
Total rows loaded: 35

	Course_ID	Title	Section_ID
1	135	Computer Science I	135-1001
2	135	Computer Science I	135-1001
3	135	Computer Science I	135-1001
4	135	Computer Science I	135-1001
5	135	Computer Science I	135-1001
6	135	Computer Science I	135-1101
7	135	Computer Science I	135-1101
8	135	Computer Science I	135-1101
9	135	Computer Science I	135-1101
10	135	Computer Science I	135-1101

4. All upper-division classes on MW / TH / MWF

MW:

```
SELECT c.Course_ID, cs.Section_ID, cs.Meeting_Day
FROM Course c
JOIN Course_Section cs ON c.Course_ID = cs.Course_ID
WHERE CAST(c.Course_ID AS INTEGER) >= 300
  AND cs.Meeting_Day = 'MW';
```



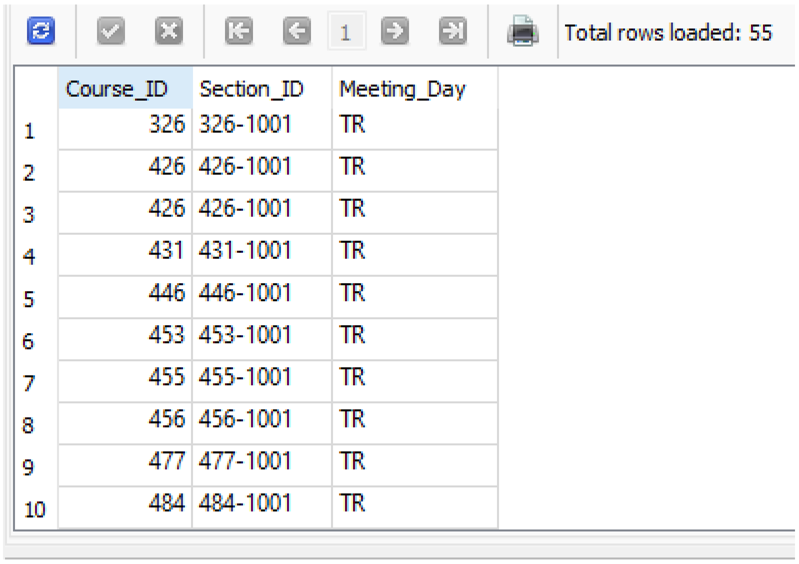
Total rows loaded: 40

	Course_ID	Section_ID	Meeting_Day
1	302	302-1001	MW
2	381	381-1001	MW
3	445	445-1001	MW
4	447	447-1001	MW
5	457	457-1001	MW
6	479	479-1001	MW
7	480	480-1001	MW
8	487	487-1001	MW
9	302	302-1001	MW
10	381	381-1001	MW

TH:

```
SELECT c.Course_ID, cs.Section_ID, cs.Meeting_Day
FROM Course c
JOIN Course_Section cs ON c.Course_ID = cs.Course_ID
WHERE CAST(c.Course_ID AS INTEGER) >= 300
```

```
  AND cs.Meeting_Day = 'TR';
```

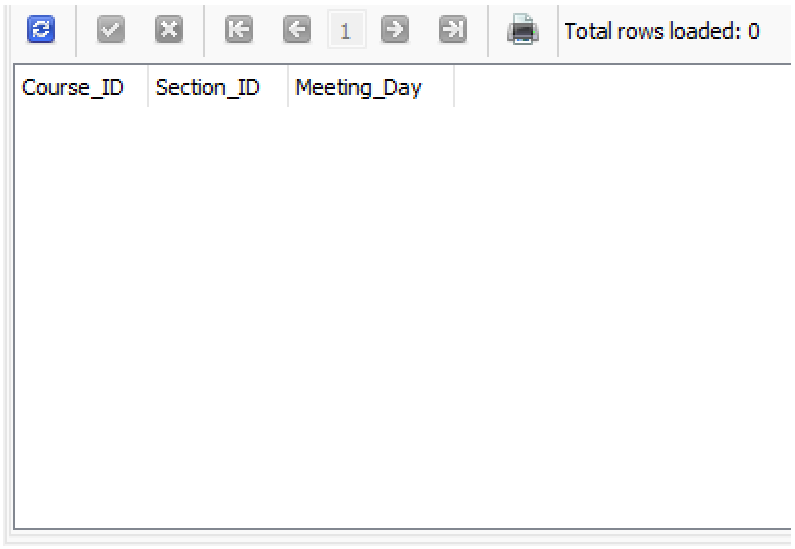


Total rows loaded: 55

	Course_ID	Section_ID	Meeting_Day
1	326	326-1001	TR
2	426	426-1001	TR
3	426	426-1001	TR
4	431	431-1001	TR
5	446	446-1001	TR
6	453	453-1001	TR
7	455	455-1001	TR
8	456	456-1001	TR
9	477	477-1001	TR
10	484	484-1001	TR

MWF:

```
SELECT c.Course_ID, cs.Section_ID, cs.Meeting_Day
FROM Course c
JOIN Course_Section cs ON c.Course_ID = cs.Course_ID
WHERE CAST(c.Course_ID AS INTEGER) >= 300
  AND cs.Meeting_Day = 'MWF';
```

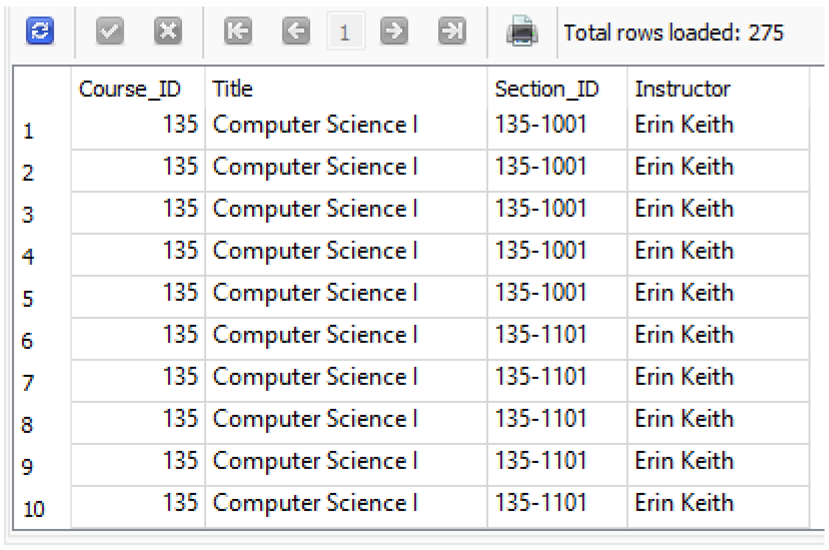


Total rows loaded: 0

Course_ID	Section_ID	Meeting_Day

5. All classes instructed by Prof. Keith, Prof. Hastings

```
SELECT c.Course_ID, c.Title, cs.Section_ID, i.Name AS Instructor
FROM Course c
JOIN Course_Section cs ON c.Course_ID = cs.Course_ID
JOIN Instructor i ON cs.Instructor_ID = i.Instructor_ID
WHERE i.Name LIKE '%Keith%' OR i.Name LIKE '%Hastings%';
```



Total rows loaded: 275

	Course_ID	Title	Section_ID	Instructor
1	135	Computer Science I	135-1001	Erin Keith
2	135	Computer Science I	135-1001	Erin Keith
3	135	Computer Science I	135-1001	Erin Keith
4	135	Computer Science I	135-1001	Erin Keith
5	135	Computer Science I	135-1001	Erin Keith
6	135	Computer Science I	135-1101	Erin Keith
7	135	Computer Science I	135-1101	Erin Keith
8	135	Computer Science I	135-1101	Erin Keith
9	135	Computer Science I	135-1101	Erin Keith
10	135	Computer Science I	135-1101	Erin Keith

```
BEGIN TRANSACTION;

-- Table: Building
CREATE TABLE IF NOT EXISTS Building (
   BLDG_ID TEXT PRIMARY KEY,
   NAME_BLDG    TEXT NOT NULL
);

-- Table: Classroom
CREATE TABLE IF NOT EXISTS Classroom (
   Bldg_ID      TEXT NOT NULL REFERENCES Building (BLDG_ID) ON DELETE CASCADE,
   Room_Num     TEXT NOT NULL,
   Floor        INTEGER NOT NULL,
   Seats        INTEGER NOT NULL,
   WiFi         TEXT DEFAULT 'No',
   ADA_Access   TEXT DEFAULT 'No',
   PRIMARY KEY (Bldg_ID, Room_Num)
);

-- Table: Course
CREATE TABLE IF NOT EXISTS Course (
   Course_ID   TEXT PRIMARY KEY,
   Title       TEXT NOT NULL,
   Description TEXT,
   No_Credits  INTEGER NOT NULL,
   Department  TEXT,          -- Added for Department/Subject Section
   Subject     TEXT           -- Added for Department/Subject Section
);
```

```
-- Table: Course_Prerequisite
CREATE TABLE IF NOT EXISTS Course_Prerequisite (
   Course_ID        TEXT NOT NULL REFERENCES Course (Course_ID) ON DELETE CASCADE,
   Pre_Req_Course_ID TEXT NOT NULL REFERENCES Course (Course_ID) ON DELETE
CASCADE,
   PRIMARY KEY (Course_ID, Pre_Req_Course_ID)
);

-- Table: Course_Section
CREATE TABLE IF NOT EXISTS Course_Section (
   Section_ID         TEXT PRIMARY KEY,
   Course_ID          TEXT NOT NULL REFERENCES Course (Course_ID) ON DELETE CASCADE,
   Instructor_ID      TEXT NOT NULL REFERENCES Instructor (Instructor_ID) ON DELETE SET
NULL,
   Bldg_ID            TEXT NOT NULL,
   Room_Num           TEXT NOT NULL,
   Term               TEXT NOT NULL,
   College            TEXT,              -- Added for College Section
   Component          TEXT,              -- Added for Component (Lab/Lec/etc.)
   Meeting_Day        TEXT NOT NULL,
   CLASS_START_TIME   TEXT,              -- Added for CLASS_START_TIME
   CLASS_END_TIME     TEXT,              -- Added for CLASS_END_TIME
   Meeting_Time       TEXT NOT NULL,
   Section_Code       TEXT NOT NULL,
   Enrollment_Capacity INTEGER NOT NULL,
   Waitlist_Capacity  INTEGER NOT NULL,
   Status             TEXT DEFAULT 'Open',
   Waitlist_Actual    INTEGER DEFAULT 0,
   Enrollment_Actual  INTEGER DEFAULT 0,
   FOREIGN KEY (Bldg_ID, Room_Num) REFERENCES Classroom (Bldg_ID, Room_Num) ON
DELETE SET NULL
);

-- Table: Instructor
CREATE TABLE IF NOT EXISTS Instructor (
   Instructor_ID TEXT PRIMARY KEY,
   FIRST_NAME    TEXT,              -- Added
   LAST_NAME     TEXT,              -- Added
   NAME_INST     TEXT               -- Renamed from NAME for clarity
);

COMMIT TRANSACTION;
```

Links

https://github.com/NorthrupRobert/CS_457.1001_Final_Project_Phase_II/blob/main/College.sql