

# Project 4: Queens College Course Database

Members: Theo, Jacelin, Rahat, Mateo, Sameeha, Prachi,  
Unsa

# Table of Contents

01

Project Overview

04

Entity Relationship Diagram (ERD)

02

Dataset Issues & Adjustments

05

Propositions + Queries

03

Database Design & Normalized Tables

06

Conclusion

# 01: Project Overview

# Project Purpose & Teamwork

This project asks us to design/implement a relational database system which manages class scheduling in a college. The database is to adhere to “industry” standards, enforced naming convention, schema segregation, and the use of udt’s for every column in the system. This will allow for the database to be highly maintainable and self documenting for future development, all ensuring 3NF compliance.

Data Integrity is to be assured through the use of primary, foreign, and alternative keys.

Team collaboration:

- Tools we used (Dbeaver, ADS, SQLDBM, etc)
- How we split the work:  
Each group member was assigned with the implementation of a table.
- Challenges we faced, if any

# 02: Dataset Issues & Adjustments

# Problems in the Original Dataset

## Main Issues:

- Data was stored in a generic table
- No primary or foreign keys
- Missing or Incomplete Data
- Non-atomic columns
- Unenforced Constraints

Notice the issues with the data

```
SELECT TOP (1000) [Semester]
,[Sec]
,[Code]
,[Course (hr, crd)]
,[Description]
,[Day]
,[Time]
,[Instructor]
,[Location]
,[Enrolled]
,[Limit]
,[Mode of Instruction]
FROM [QueensClassSchedule].[Uploadfile].[CurrentSemesterCourseOfferings]
```

	Current Semester	U1	11844 ANTH 2801 J...	Language and Social I...	T, TH	3:10 PM - 4:20 PM	Hodnquez Poonte, J...	PM 114	49	33	Web Enhanced
184	Current Semester	01	62926 ANTH 2953 (...	Ind Std-Anth	-		Pechenkina, Ekateri...	1	1	1	In-Person
185	Current Semester	02	62476 ANTH 2953 (...	Ind Std-Anth			Tache, Karine	1	1	1	In-Person
186	Current Semester	01	48129 ANTH 302 (3...	Ecology And Culture	T, TH	10:45 AM - 11:59 AM	Moore, James	PH 311	20	18	In-Person
187	Current Semester	01	11840 ANTH 354 (3...	Time	T, TH	1:40 PM - 2:55 PM	Bith, Kevin	PH 351	15	15	In-Person
188	Current Semester	01	11837 ANTH 361 (3...	Human Variation	M, W	10:45 AM - 12:00 PM	Madimenos, Felicia	PH 311	18	17	Web Enhanced
189	Current Semester	01	56480 ANTH 390 (3...	Senior Honor Thesis	-		Swedell, Larissa		1	1	In-Person
190	Current Semester	03	62480 ANTH 390 (3...	Senior Honor Thesis	-		Pechenkina, Ekateri...		1	1	In-Person
191	Current Semester	02	56863 ANTH 390 (3...	Senior Honor Thesis	-		Strassler, Karen		1	1	In-Person
192	Current Semester	1	57969 ANTH 3953 (...	Directed Studies	-		Pechenkina, Ekateri...		1	1	In-Person
193	Current Semester	01	44971 ARAR 101 M A	Class Arabic 1	T, TH	10:00 AM - 11:55 AM	Salaimani, Kamal	OH 240C	25	25	In-Person

# Data Cleansing & Fixes

- We decomposed the generic table into 7 specific tables: Department, Instructor, Course, Class, Room, Building Location, Mode of Instruction
- For each table we defined Primary Keys and Foreign Keys to enforce relationships between entities
- Created column names that are atomic
- Implemented constraints to prevent invalid data
- We defined User-Defined Data Types
- Created a Schema Name for the Tables: ClassSchedule

# 03: Database Design & Normalized Tables

# Department Table

What this table stores:

Key columns:

DepartmentID which is used as the Primary Key

DepartmentCode serves as the Alternative Key

DepartmentName is another attribute/column

Relationships: The Department Table connects to Course as a parent table. It links to InstructorDepartment.

Naming conventions & schemas used: ClassSchedule is the schema for which all departmental data was organized under.

Constraints & Data integrity:

Ensured by DepartmentID as the Primary Key with Constraint: Identity(1,1)

DepartmentName Constraint: NOT NULL

DepartmentCode Constraint: UNIQUE and NOT NULL

UDTs used: SurrogateKeyInt, NameNVARCHAR100, CodeNCHAR4

# Department Table

Department	
123	DepartmentID SurrogateKeyInt NOT NULL
A-Z	DepartmentCode DepartmentCode NOT NULL
123	UserAuthorizationKey SurrogateKeyInt NOT NULL
⌚	DateAdded DateAdded NOT NULL
⌚	DateOfLastUpdate DateOfLastUpdate NOT NULL

# Department Table continued...

Cardinality:

One Department → Many Courses (1 to Many): A single department can manage and offer different courses.

One Department → Many Assigned Instructors (1 to Many): One department can be the base for different instructors through the InstructorDepartment table.

One User → Many Departments (1 to Many): A single user in the UserAuthorization table can manage and update several department records.

One Department → Exactly One DepartmentID (1 to 1): Each unique department record is identified by exactly one primary key used as a reference throughout the schema.

# Instructor Table

Instructor Table stores the names and unique identification data for each faculty member.

Key Columns:

Primary Key: InstructorID is a surrogate key using SurrogateKeyInt to uniquely identify each instructor.

Attributes: FirstName and LastName display the legal name of a member of the faculty.

Relationships: Connects to this bridge table to associate instructors with their respective academic departments.

Connects to the Class table to pinpoint which instructor is teaching a specific section.

Connects to the security schema for auditing of records.

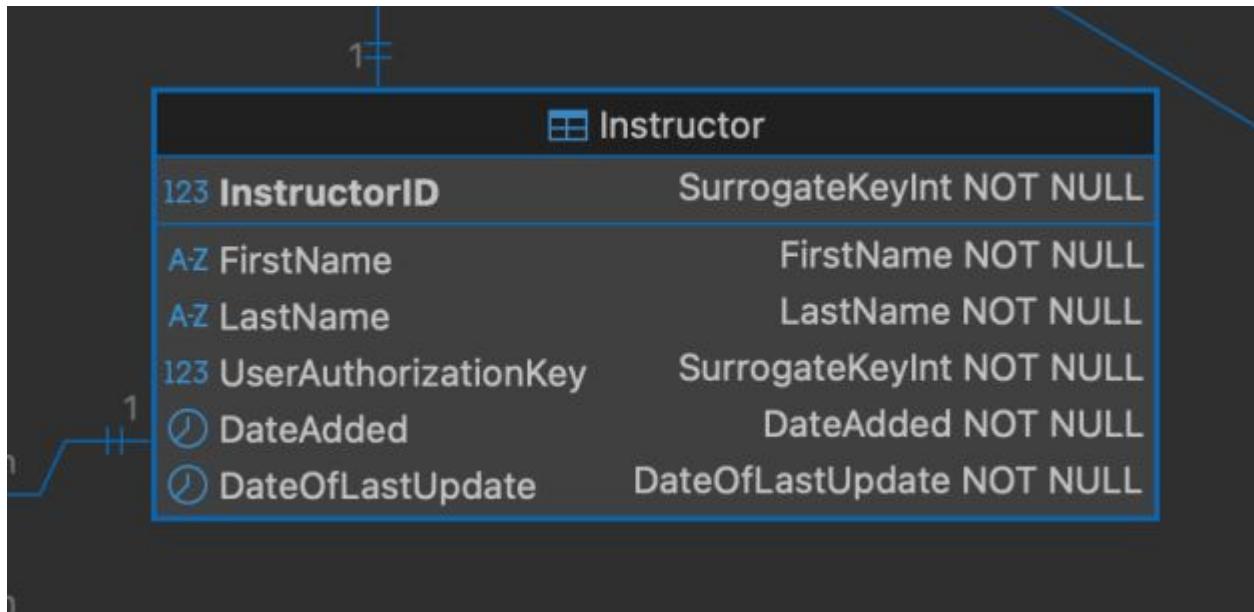
The table is contained within the ClassSchedule schema to keep academic data isolated from security or general system data.

PascalCase for all identifiers to ensure readability and consistency..

Constraints & Data integrity: To keep the data clean we made it impossible to leave the name fields empty because a teacher record isn't useful without a name. We also made sure the ID number is always unique so that every teacher is accounted for correctly.

UDTs used: Instead of using standard database types, we used custom "labels" for our data types called UDTs. Our team shared this frame. For example, if we decide to change how long a name can be, we only have to change it once in the template, updating every table in the database automatically.

# Instructor Table



# Instructor Table

Cardinality (Based on ERD):

- One Instructor → Many Classes (1 to Many): One teacher can be assigned to lead several different classes throughout the day.
- One Instructor → Many Department Assignments (1 to Many): A teacher might work for more than one department (like teaching both Math and Computer Science).
- One User → Many Instructors (1 to Many): One office worker can enter the profiles for many different teachers into the system.
- One Instructor → Exactly One InstructorID (1 to 1): Every teacher gets one unique ID number that belongs only to them.

# Course Table

This table acts as a list for all academic classes offered by the college. It keeps track of details like the course name, how many credits it is worth, and which department is responsible for its teaching.

- CourseID → This is the primary identification number for every course. It uses a "surrogate key", a unique number, to ensure every entry is unique.
- CourseCode and CourseName → These columns store the shorthand code and the full title of the class
- DepartmentID → The link between a course and its home department; foreign key.
- Relationships → Connects to the Department table to show who owns the class and to the Class table to show the sections being taught. In order to trace who made the course record Course links to the security schema.
- Naming Conventions/Schemas → Course resides in the ClassSchedule Schema. It also follows PascalCase for all names to ensure consistency.
- Constraints and Data Integrity → CourseCredits and CourseHours are mandatory so that no one can accidentally add a course without defining its academic value. It is also ensured that every CourseID is unique so there are no duplicates in the system.
- UDTs → Changed how we store "Course Credits" later on, we only have to update it in one place in our udt schema, and it will fix the entire database automatically.

# Course Table

## Relationships:

- Links to Department in order to see which specific department is in charge of the course in question.
- Links to Class in order to see specific rooms and time sections for the course being taught.
- Links to User Authorization in order to display the staff member added a class to the system.

# Course Table

```
CREATE TABLE Course (
    CourseID SurrogateKeyInt IDENTITY(1,1) NOT NULL,
    DepartmentID uniqueidentifier NOT NULL,
    CourseCode varchar(20) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    CourseName varchar(200) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    CourseHours CourseHour NOT NULL,
    CourseCredits CourseCredit NOT NULL,
    UserAuthorizationKey SurrogateKeyInt NOT NULL,
    DateAdded DateAdded DEFAULT sysdatetime() NOT NULL,
    DateOfLastUpdate DateOfLastUpdate DEFAULT sysdatetime() NOT NULL,
    CONSTRAINT PK_ClassSchedule_CourseID PRIMARY KEY (CourseID),
    CONSTRAINT FK_Course_Department_DepartmentID FOREIGN KEY (DepartmentID) REFERENCES
    Department(DepartmentID)
);
CREATE UNIQUE NONCLUSTERED INDEX IX_Course_CourseName_CourseCode ON
QueensClassScheduleThisCurrentSemester.ClassSchedule.Course (CourseName, CourseCode);
```

# Class Table – Purpose

- The **Class** table represents the individual classes (**sections**) that are being offered for the current semester
- Each entry in the table represents one specific section and includes details such as when the class meets, how many students are enrolled and more.
- This table was needed to keep specific class details separate from other data like Courses, Instructors, and Departments
- Separating this data helps reduce redundancy, improve data integrity and makes it easier to connect each class to related tables

# Class Table – Key Columns & Indexes

**Primary Key:** `ClassID`

**Foreign Keys:**

`CourseID` → Course

`InstructorID` → Instructor

`RoomID` → Room

`ModeOfInstructionID` → Mode of Instruction

**Alternate Keys:** `Semester`, `CourseID`,  
`Section`

**Core Attributes:**

- `Semester` and `Section` identify when and which section is being offered
- `MeetingDays`, `StartTime`, and `EndTime` describe the class schedule
- `Enrolled` and `EnrollmentLimit` track class capacity and enrollment status

**Indexes:**

`IX_Class_CourseID_Semester_Section`

`IX_Class_StartTime_EndTime_Section`

# Class Table – Table Relationships

## Connected Tables:

- **Course** → identifies which course the class section belongs to
- **Instructor** → identifies who teaches the class
- **Room** → identifies where the class meets
- **Mode of Instruction** → how the class meets (online, hybrid, in-person, etc)

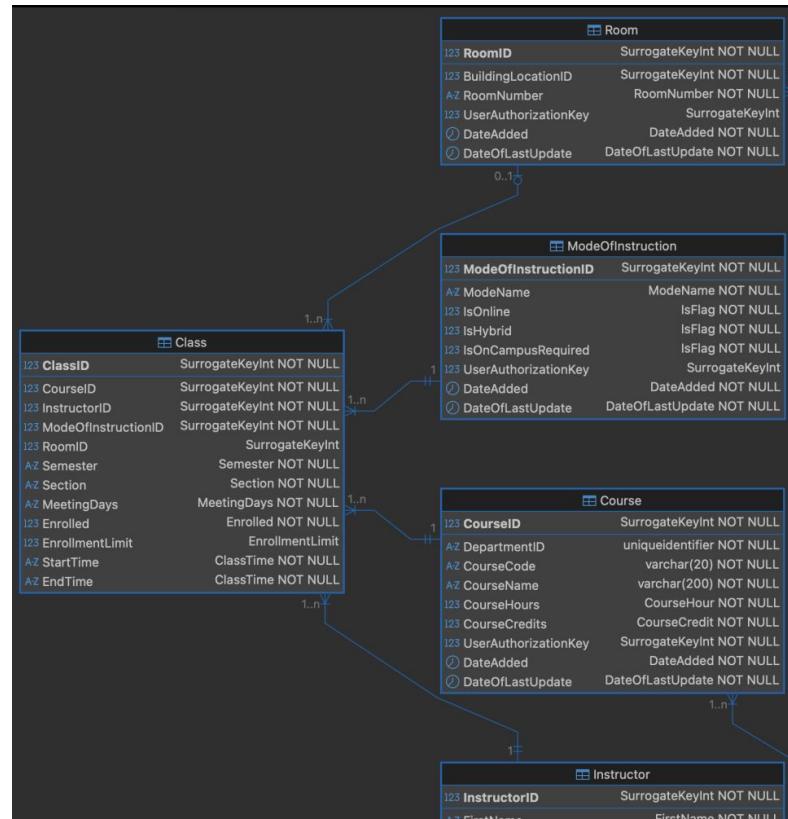
## Cardinality:

One Course → Many Classes (**1 to Many**): A single course can have multiple class sections in a semester

One Instructor → Many Classes (**1 to Many**): An instructor can teach multiple class sections

One Room → Many Classes (**1 to Many**): A room can host multiple classes at different times

One Mode of Instruction → Many Classes (**1 to Many**): A mode of instruction, such as online or in-person, can be used by multiple class sections



# Class Table – Constraints

## NOT NULL Constraints

```
[Section] NOT NULL, Enrolled IsFlag NOT NULL,
```

## CHECK Constraints

```
ADD CONSTRAINT CK_Class_Enrolled_NonNegative CHECK(([Enrolled]>=(0)));
```

```
ADD CONSTRAINT CK_Class_EnrollmentLimit_NonNegative CHECK(([EnrollmentLimit] IS NULL OR [EnrollmentLimit]>=(0)));
```

## UNIQUE (Alternate Key) Constraint

```
CONSTRAINT AK_Class_Semester_CourseID_Section UNIQUE (Semester,CourseID,[Section])
```

# Room Location Table

## 1) Key columns

RoomID (PK), BuildingLocation (FK), Room number and IsActive

## 1) Relationships:

Links to Building Location table

## 2) Naming conventions & schemas used

dbo schema, PascalCase column names

## 3) Constraints & Data integrity

Primary key on RoomID

Foreign Key on BuildingLocationID

## 4) UDTs used

User- defined data types for consistency

## 5) Design Decisions or fixes you made

Separated rooms from buildings to avoid redundancy

# Building Location Table

Stores campus building information

- 1) Key columns
  - a) BuildingLocationID (PK), BuildingCode, BuildingName, CampusName, IsActive
- 2) Relationships:
  - a) Referenced by room table
- 3) Naming conventions & schemas used
  - a) Dbo schema
- 4) Constraints & Data integrity
  - a) Primary Key on BuildingLocationID
  - b) Unique constraint on BuildingCode
- 5) UDTs used
  - a) User-defined data types for building attributes
- 6) Design Decisions or fixes you made
  - a) User-surrogate key instead of building code as PK

	Description	BuildingCode	RoomNumber	Location
1	Fin & Mgr Acct	KY	419	KY 419
2	Fin & Mgr Acct	HH	17	HH 17
3	Fin & Mgr Acct	KY	419	KY 419
4	Int Theo & Prac Acct 1	RA	201	RA 201
5	Int Theo & Prac Acct 1	PH	204	PH 204
6	Int Theo & Prac Acct 1	PH	110	PH 110
7	Int Theo & Prac Acct 1	PH	212	PH 212
8	Int Theo & Prac Acct 1	PH	110	PH 110
9	Int Theo & Prac Acct 1	PH	130	PH 130
10	Int Theo & Prac Acct 1	SB	D133	SB D133
11	Int Theo & Prac Acct 1	RO	230	RO 230
12	Int Theo & Prac Acct 1	PH	130	PH 130
13	Int Theo & Prac Acct 1	PH	118	PH 118
14	Int Theo & Prac Acct 1	RA	201	RA 201
15	Int Theo & Prac Acct 1	PH	130	PH 130
16	Intro Theo & Prac Acct 2	RA	201	RA 201
17	Intro Theo & Prac Acct 2	PH	116	PH 116

# Mode of Instruction Table: Purpose

## Purpose

- Defines how classes are delivered at Queens College
- Acts as a **lookup / reference table**

## What this table stores

- Standardized list of valid course delivery modes
- Each row represents one instruction type

## Examples

- In-Person
- Online Asynchronous
- Online Synchronous
- Hybrid

Prevents repeating delivery mode text in the Class table.

# Mode of Instruction Table: Key Columns & Index Design

## Key Columns

- **ModeOfInstructionID**
  - Primary Key
  - Surrogate key using **IDENTITY(1,1)**
- **ModeName**
  - Descriptive name of the instruction mode

## Primary Key & Index Design

- **Primary Key:** ModeOfInstructionID
  - Ensures uniqueness and efficient joins
- **Alternate Key / UNIQUE index:** ModeOfInstructionName
  - Prevents duplicate instruction modes
  - Improves lookup performance from Class

# Mode of Instruction Table: Relationships

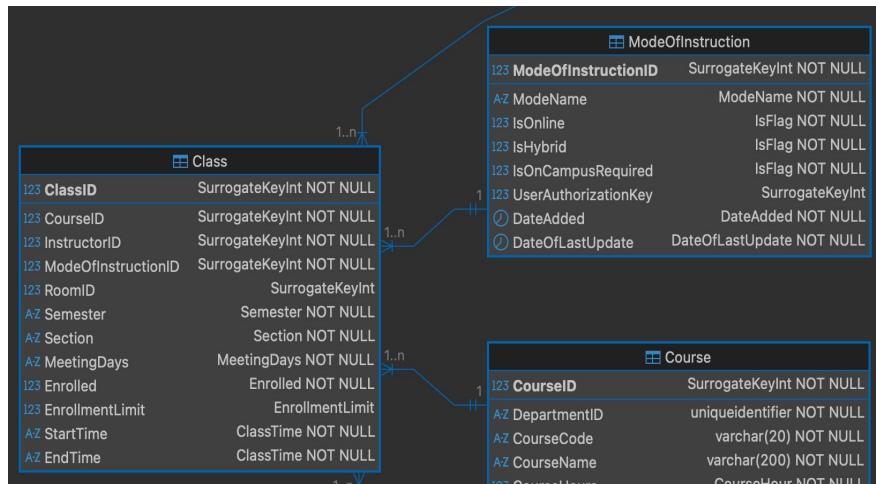
## Connected Tables

- **Class → ModeOfInstruction**
  - Class table references ModeOfInstructionID
  - (*Relationship shown in ERD Diagram*)

## Cardinality

- One ModeOfInstruction → Many Classes (1:M)
- Each class has exactly one mode of instruction

Ensures consistent delivery methods across all class sections.



# Mode of Instruction Table – Naming & Schema

## Naming Conventions

- PascalCase used for:
  - Table names
  - Column names
  - Constraint names
- No underscores used

## Schema Used

- Table exists under the **ClassSchedule** schema
- Schema separation improves clarity and maintainability

# Mode of Instruction Table – Constraints & Data Integrity

## NOT NULL Constraints

- ModeName must always be provided
- Instruction mode flags must always be defined

```
ModeName NOT NULL,  
IsOnline IsFlag NOT NULL,  
IsHybrid IsFlag NOT NULL,  
IsOnCampusRequired IsFlag NOT NULL;
```

## UNIQUE (Alternate Key) Constraint

- Ensures each instruction mode name is defined only once
- Prevents duplicate delivery modes from being inserted

```
CONSTRAINT AK_ModeOfInstruction_ModeName  
UNIQUE (ModeName);
```

## Data Integrity Summary

- Enforces valid and complete instruction modes
- Guarantees consistency before being referenced by the Class table

# Mode of Instruction Table – UDTs Used

## User Defined Datatypes

- **Udt.SurrogateKeyInt**
  - Used for ModeOfInstructionID
- **Udt.NameNVARCHAR100**
  - Used for ModeName

## Hierarchy & Reuse

- These UDTs are reused across the ClassSchedule schema
- Improves consistency and self-documentation
- Allows centralized datatype changes without rewriting tables

Table name

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ▲ Move Up ▼ Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
=	ModeOfInstructionID	Udt.SurrogateKeyInt	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
=	ModeName	Udt.NameNVARCHAR100	<input type="checkbox"/>	<input type="checkbox"/>		
=	IsOnline	Udt.IsFlag	<input type="checkbox"/>	<input type="checkbox"/>		
=	IsHybrid	Udt.IsFlag	<input type="checkbox"/>	<input type="checkbox"/>		
=	IsOnCampusRequired	Udt.IsFlag	<input type="checkbox"/>	<input type="checkbox"/>		
=	UserAuthorizationKey	Udt.SurrogateKeyInt	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
=	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
=	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

# Mode of Instruction Table – Design Decisions & Fixes

## Design Decisions

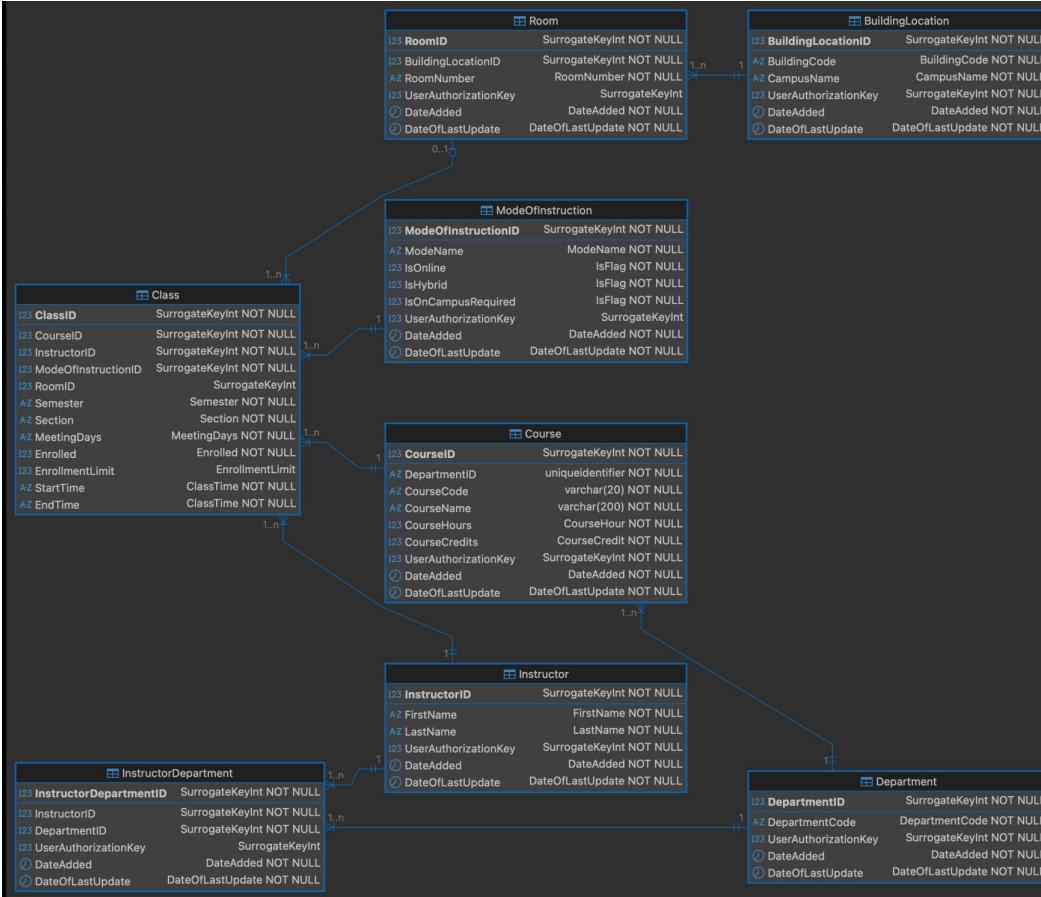
- Implemented as a lookup table instead of storing text in Class
- Used a surrogate key instead of a natural key
- Enforced uniqueness through constraints rather than application logic

## Fixes from Original Dataset

- Removed inconsistent instruction values
- Eliminated non-atomic delivery mode data
- Standardized delivery methods across all classes

# 04: Entity Relationship Diagrams

# Conceptual Data Model (CDM)



# Physical Data Model (PDM)

Table name Room

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ▲ Move Up ▼ Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
=	RoomID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
=	BuildingLocationID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	RoomNumber	Udt.RoomNumber	<input type="checkbox"/>	<input type="checkbox"/>		
=	UserAuthorizationKey	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
=	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
=	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

MateoCh\_Indigo - 2024

Table name ModeOfInstruction

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ^ Move Up v Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
==	ModeOfInstructionID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
==	ModeName	Udt.ModeName	<input type="checkbox"/>	<input type="checkbox"/>		
==	IsOnline	Udt.IsFlag	<input type="checkbox"/>	<input type="checkbox"/>		
==	IsHybrid	Udt.IsFlag	<input type="checkbox"/>	<input type="checkbox"/>		
==	IsOnCampusRequired	Udt.IsFlag	<input type="checkbox"/>	<input type="checkbox"/>		
==	UserAuthorizationKey	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
==	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
==	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

Table name

InstructorDepartment

[Columns](#) [Primary Key](#) [Foreign Keys](#) [Check Constraints](#) [Indexes](#) [General](#)[+ New Column](#) [^ Move Up](#) [v Move Down](#)

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
=	InstructorDepartmentID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
=	InstructorID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	DepartmentID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	UserAuthorizationKey	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
=	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
=	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

Table name

**Columns** Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ▲ Move Up ▼ Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
==	InstructorID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
==	FirstName	Udt.FirstName	<input type="checkbox"/>	<input type="checkbox"/>		
==	LastName	Udt.LastName	<input type="checkbox"/>	<input type="checkbox"/>		
==	UserAuthorizationKey	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
==	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
==	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

Table name

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ^ Move Up v Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
=	DepartmentID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
=	DepartmentCode	Udt.DepartmentC...	<input type="checkbox"/>	<input type="checkbox"/>		
=	UserAuthorizationKey	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
=	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

Table name Course

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ▲ Move Up ▼ Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
==	CourseID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
==	DepartmentID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
==	CourseCode	varchar(20)	<input type="checkbox"/>	<input type="checkbox"/>		
==	CourseName	varchar(200)	<input type="checkbox"/>	<input type="checkbox"/>		
==	CourseHours	Udt.CourseHour	<input type="checkbox"/>	<input type="checkbox"/>		
==	CourseCredits	Udt.CourseCredit	<input type="checkbox"/>	<input type="checkbox"/>		
==	UserAuthorizationKey	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
==	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
==	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

Table name

**Columns** Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ▲ Move Up ▼ Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
=	BuildingLocationID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
=	BuildingCode	Udt.BuildingCode	<input type="checkbox"/>	<input type="checkbox"/>		
=	CampusName	Udt.CampusName	<input type="checkbox"/>	<input type="checkbox"/>		
=	UserAuthorizationKey	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	DateAdded	Udt.DateAdded	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	
=	DateOfLastUpdate	Udt.DateOfLastUp...	<input type="checkbox"/>	<input type="checkbox"/>	(sysdatetime())	

Table name

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column ^ Move Up v Move Down

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove
=	ClassID	Udt.SurrogateKeyl...	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
=	CourseID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	InstructorID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	ModeOfInstructionID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input type="checkbox"/>		
=	RoomID	Udt.SurrogateKeyl...	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
=	Semester	Udt.Semester	<input type="checkbox"/>	<input type="checkbox"/>		
=	Section	Udt.Section	<input type="checkbox"/>	<input type="checkbox"/>		
=	MeetingDays	Udt.MeetingDays	<input type="checkbox"/>	<input type="checkbox"/>		
=	Enrolled	Udt.Enrolled	<input type="checkbox"/>	<input type="checkbox"/>		
=	EnrollmentLimit	Udt.EnrollmentLimit	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
=	StartTime	Udt.ClassTime	<input type="checkbox"/>	<input type="checkbox"/>		
=	EndTime	Udt.ClassTime	<input type="checkbox"/>	<input type="checkbox"/>		

```
-- 1. All instructors teaching in multiple departments
SELECT
    i.FirstName,
    i.LastName,
    COUNT(DISTINCT d.DepartmentID) AS DepartmentCount
FROM ClassSchedule.Instructor i
JOIN ClassSchedule.InstructorDepartment id
    ON i.InstructorID = id.InstructorID
JOIN ClassSchedule.Department d
    ON id.DepartmentID = d.DepartmentID
GROUP BY i.FirstName, i.LastName
HAVING COUNT(DISTINCT d.DepartmentID) > 1;
```

# 05: Propositions + Queries

# Proposition 1

Proposition: Instructors who are teaching in classes in multiple departments

```
-- 1. All instructors teaching in multiple departments
SELECT
    i.FirstName,
    i.LastName,
    COUNT(DISTINCT d.DepartmentID) AS DepartmentCount
FROM ClassSchedule.Instructor i
JOIN ClassSchedule.InstructorDepartment id
    ON i.InstructorID = id.InstructorID
JOIN ClassSchedule.Department d
    ON id.DepartmentID = d.DepartmentID
GROUP BY i.FirstName, i.LastName
HAVING COUNT(DISTINCT d.DepartmentID) > 1;
```

## Proposition 2

Proposition: How many instructors in each department

-- 2. How many instructors are in each department

SELECT

```
d.DepartmentCode,  
COUNT(DISTINCT id.InstructorID) AS InstructorCount  
FROM ClassSchedule.Department d  
LEFT JOIN ClassSchedule.InstructorDepartment id  
    ON d.DepartmentID = id.DepartmentID  
GROUP BY d.DepartmentCode;
```

# Proposition 3

Proposition: Classes that are being taught that semester grouped by course and aggregating the total enrollment, total class limit and the percentage of enrollment.

```
-- 3. Classes by course with enrollment for the semester
SELECT
    crs.CourseCode,
    crs.CourseName,
    c.Semester,
    COUNT(c.ClassID) AS ClassCount,
    SUM(c.Enrolled) AS TotalEnrolled,
    SUM(c.EnrollmentLimit) AS TotalCapacity,
    CAST(
        (SUM(c.Enrolled) * 100.0) / NULLIF(SUM(c.EnrollmentLimit), 0)
        AS DECIMAL(5,2)
    ) AS EnrollmentPercentage
FROM ClassSchedule.Class c
JOIN ClassSchedule.Course crs
    ON c.CourseID = crs.CourseID
GROUP BY
    crs.CourseCode,
    crs.CourseName,
    c.Semester
ORDER BY crs.CourseCode;
```

# 06: Conclusion

# Conclusion

What we accomplished:

- Designed a fully normalized database
- Identified and fixed data anomalies using constraints and proper table design
- Created reusable User Defined Data Types (UDTs) for consistency and clarity

What we learned:

- How to analyze messy data and redesign it
- The importance of normalization, constraints, and data integrity

Why this matters:

- The skills we used are similar to the ones used in data analysis jobs