Running Example:

Assumption:

Each class can be taught by multiple professors.

Each professor can teach multiple courses.

A class can have multiple sections i.e., multiple classroom.

A student can take multiple courses but not the same course more than once.

A single course with multiple sections has same start and end date.

Input:

Input dataset:

>>> students.csv

Student ID	First Name	Last Name	Course	Professor	ProfessorEmail	CourseStart	CourseEnd	classRoom
101	John	Doe	Math101	Dr.Smith	smith@mst.edu	1/1/2023	5/30/2023	M1
101	John	Doe	CS101	Dr.Jones	jones@mst.edu	2/1/2023	6/15/2023	C1
102	Jane	Roe	Math101	Dr.Smith	smith@mst.edu	1/1/2023	5/30/2023	M1
102	Jane	Roe	CS101	Dr.Smith	smith@mst.edu	2/1/2023	6/15/2023	C2
103	Arindam	Khanda	CS101	Dr.Jones	jones@mst.edu	2/1/2023	6/15/2023	C1
104	Jose	Franklin	Bio101	Dr.Watson	watson@mst.edu	3/1/2023	7/20/2023	B1
105	Ada	Lovelace	CS101	Dr.Jones	jones@mst.edu	2/1/2023	6/15/2023	C1

Input Functional Dependencies (type “exit” and hit enter to complete your dependency list):

>>> StudentID -> FirstName, LastName

>>> Course, Professor -> classroom

>>> Course -> CourseStart, CourseEnd

>>> Professor -> ProfessorEmail

>>> exit

Input Multi-valued Dependencies (type “exit” and hit enter to complete your dependency list):

>>> Course ->> Professor

>>> Course ->> classroom

>>> StudentID ->> Course

>>> StudentID ->> Professor

** NOTE: You may store the functional dependencies in a .txt file and provide the .txt file as input also**

Choice of the highest normal form to reach (1: 1NF, 2: 2NF, 3: 3NF, B: BCNF, 4: 4NF, 5: 5NF):

>>> 4

Find the highest normal form of the input table? (1: Yes, 2: No):

>>> 1

NOTE: If we cannot decide how to find the key in the input table we may take the key as input

Key (can be composite):

>>> StudentID, Course

Output:

SQL queries to create 4NF:

```
CREATE TABLE Student (
   StudentID INT PRIMARY KEY,
   FirstName VARCHAR(255) NOT NULL,
   LastName VARCHAR(255) NOT NULL
);
CREATE TABLE CourseDetails (
   Course VARCHAR(255) PRIMARY KEY,
   CourseStart DATE NOT NULL,
   CourseEnd DATE NOT NULL
);
CREATE TABLE ProfessorDetails (
   Professor VARCHAR(255) PRIMARY KEY,
   ProfessorEmail VARCHAR(255) NOT NULL
);
CREATE TABLE ClassRoomDetails (
```

```
    Course VARCHAR(255),
    Professor VARCHAR(255),
    classRoom VARCHAR(255) NOT NULL,
    PRIMARY KEY (Course, Professor),
    FOREIGN KEY (Course) REFERENCES CourseDetails(Course),
    FOREIGN KEY (Professor) REFERENCES ProfessorDetails(Professor)
);
CREATE TABLE Enrollment (
    StudentID INT,
    Course VARCHAR(255),
    Professor VARCHAR(255),
    PRIMARY KEY (StudentID, Course),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (Course) REFERENCES CourseDetails(Course),
    FOREIGN KEY (Professor) REFERENCES ProfessorDetails(Professor)
);
```

Highest normal form of the input table: 1NF

Logical deduction:

1NF:

- The table already satisfies 1NF. Every attribute contains atomic values.
- This table doesn't handle nested relations in 1NF.
- We take care of the multi-valued attributes in 4NF.

2NF (Removing partial dependencies):

>>> StudentID -> FirstName, LastName

>>> Course, Professor -> classroom

>>> Course -> CourseStart, CourseEnd

- The given composite key is (StudentID, Course)
- Since Course-> CourseStart, CourseEnd, CourseStart, CourseEnd are dependent on Course. So, partial dependencies exist.
- FirstName, LastName depend only on StudentID. So, partial dependencies exist.
- So, the input table is not in 2NF and it was in 1NF.
- Breaking the partial dependencies gives the tables Student(StudentID (Primary Key), FirstName, LastName), CourseDetails(Course (Primary Key), CourseStart, CourseEnd), Enrollment(StudentID (Composite Key), Course (Composite Key), Professor, ProfessorEmail , classRoom)

3NF (Removing transitive dependencies):

- Transitive dependencies are there only in table Enrollment as Professor -> ProfessorEmail transitively depends on the primary key.
- So, the new tables will be
    - Student(StudentID (Primary Key), FirstName, LastName),
    - CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
    - Enrollment(StudentID (Composite Key), Course (Composite Key), Professor, classRoom)
    - ProfessorDetails(Professor(Primary Key), ProfessorEmail)
-

BCNF (Eliminate other anomalies):

- Student(StudentID (Primary Key), FirstName, LastName),
- CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
- Enrollment(StudentID (Composite Key), Course (Composite Key), Professor, classRoom)
    - Course, Professor -> classroom
    - StudentID, Course -> Professor, classRoom
- ProfessorDetails(Professor(Primary Key), ProfessorEmail)

- The only BCNF violating functional dependency exists in Enrollment table only, since the left hand side is not a super key (Course, Professor -> classroom).
- Updated tables:
    - Student(StudentID (Primary Key), FirstName, LastName),
    - CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
    - Enrollment(StudentID (Composite Key), Course (Composite Key), Professor)
    - ClassRoomDetails(Course (Composite Key), Professor (Composite Key), classRoom)
    - ProfessorDetails(Professor(Primary Key), ProfessorEmail)

4NF (Remove multi-valued dependencies):

>>> Course ->> Professor

>>> Course ->> classroom

>>> StudentID ->> Course

>>> StudentID ->> Professor

    - Student(StudentID (Primary Key), FirstName, LastName),
    - CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
    - Enrollment(StudentID (Composite Key), Course (Composite Key), Professor)
    - ClassRoomDetails(Course (Composite Key), Professor (Composite Key), classRoom)
    - ProfessorDetails(Professor(Primary Key), ProfessorEmail)

- The creation of table ClassRoomDetails renders the multivalued dependencies Course ->> Professor and Course ->> classroom invalid since there exists a valid dependency between Professor and classroom: (Course, Professor) -> classroom. Forcing the 4NF decomposition will cause that dependency to be lost, which is a much needed constraint.

Similarly, Enrollment table renders the multivalued dependencies StudentID ->> Course and StudentID ->> Professor invalid since there exists a valid dependency between Course and Professor: (StudentID,Course) -> Professor.

So, the final tables are:

- Student(StudentID (Primary Key), FirstName, LastName),
- CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
- Enrollment(StudentID (Composite Key), Course (Composite Key), Professor)
- ClassRoomDetails(Course (Composite Key), Professor (Composite Key), classRoom)
- ProfessorDetails(Professor(Primary Key), ProfessorEmail)

All tables are in 4NF.