Running Example:

Input:

Input dataset:

>>> students.csv

StudentID	FirstName	LastName	Course	Professor	ProfessorEmail	CourseStart	CourseEnd
101	John	Doe	Math101	Dr.Smith	smith@mst.edu	1/1/2023	5/30/2023
102	Jane	Roe	Math101	Dr.Smith	smith@mst.edu	1/1/2023	5/30/2023
103	Arindam	Khanda	CS101	Dr.Yeung	syeung@mst.edu	2/1/2023	6/15/2023
104	Jose	Franklin	Bio101	Dr.Watson	watson@mst.edu	3/1/2023	7/20/2023
105	Ada	Lovelace	CS101	Dr.Yeung	syeung@mst.edu	2/1/2023	6/15/2023

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

>>> StudentID -> FirstName, LastName

>>> Course -> CourseStart, CourseEnd, Professor

>>> Professor -> ProfessorEmail

>>> exit

** 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):

>>> 3

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 3NF:

CREATE TABLE Students (

```
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(255),
    LastName VARCHAR(255)
);

CREATE TABLE Courses (
    Course VARCHAR(50) PRIMARY KEY,
    CourseStart DATE,
    CourseEnd DATE,
    Professor VARCHAR(50),
    FOREIGN KEY (Professor) REFERENCES Professors(ProfessorName)
);
CREATE TABLE Professors (
    ProfessorName VARCHAR(50) PRIMARY KEY,
    ProfessorEmail VARCHAR(100)
);
CREATE TABLE StudentCourses (
    StudentID INT,
    Course VARCHAR(255),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (Course) REFERENCES Courses(Course)
);
```

Highest normal form of the input table: 1NF

Logical deduction:

1NF:

- The table already satisfies 1NF. Every attribute contains atomic values.

2NF (Removing partial dependencies):

- The given composite key is (StudentID, Course)
- CourseStart, CourseEnd, and Professor 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 give 2 tables Courses(Course (PK), CourseStart, CourseEnd, Professor), Students(StudentID(PK), FirstName, LastName).

3NF (Removing transitive dependencies):

- Transitive dependencies exist as ProfessorEmail depends on Professor.
- Removing this dependency gives the hint of a new table Professors(Professor (PK), ProfessorEmail).

Revisiting the table hints:

- Check if an attribute X is a primary key in a table T1 and it is a non-key attribute in another table T2. Then in T2, X will be a foreign key referencing T1.
    - E.g., Professor is a PK in Professors table, but it is a non-key attribute in Courses table. So Professor should be a foreign key in Courses table.
      So new table hint Courses(Course (PK), CourseStart, CourseEnd, Professor(FK))
- Check if all the tables are directly or indirectly connected.
    - E.g., Students table has no attribute common with any other table. So this table is disconnected. In actual input table (StudentID, Course) were the composite key. So, we can use it to create a new table StudentCourses(StudentID, Course) to create the connection. Here, StudentID will be a foreign key referencing to StudentID in table Students and Course will be another foreign key referencing to Course in table Courses.
- So, the final tables are:
    - Students(StudentID(PK), FirstName, LastName)
    - Courses(Course (PK), CourseStart, CourseEnd, Professor(FK))
    - Professors(Professor (PK), ProfessorEmail)
    - StudentCourses(StudentID(FK), Course(FK))