Design Database Schema for a system like Netflix with following Use Cases.
Use Cases

1. **Netflix has users.**
2. **Every user has an email and a password.**
3. **Users can create profiles to have separate independent environments**.
4. **Each profile has a name and a type. Type can be KID or ADULT.**
5. **There are multiple videos on netflix.**
6. **For each video, there will be a title, description and a cast.**
7. **A cast is a list of actors who were a part of the video. For each actor we need to know their name and list of videos they were a part of.**
8. **For every video, for any profile who watched that video, we need to know the status (COMPLETED/ IN PROGRESS).**
9. **For every profile for whom a video is in progress, we want to know their last watch timestamp.**

User (**userId**, Email, Password)
PK: userId
FK : No
Index: No

Profile (**profile_id,**Name, Type, userId)
PK: profile_id
FK: userId
Index: No

ProfileType (**TypeId**, Value - KID/ADULT)
PK: TypeId
FK : NO
Index: NO

Videos (**VideoId**,Title, Description)
PK: VideoId
FK: No
Index: no

Actor (**actorId**, Name)

PK: actorId
FK: no
Index: no

Video_Actors(**videoId, ActorId**)

PK: videoId + ActorId
FK: videoId, actorId

- Index 1: videoId + ActorId (default) OR videoId
- Index 2: ActorId

StatusType(**StatusTypeId**, Value - COMPLETED/INPROGRESS)

PK: statusTypeId
FK: No
Index: No

VideoWatch(**profileId,videoId,** watchTimeStamp, StatusTypeId)

PK: profileId + videoId
FK: profileId, videoId, statusTypeId
Index: (profileId, videoId)
AND (videoId) depending upon requirements (**Most watch videos**)

Both (profileId, videoId) or (videoId, ProfileId) can act as primary keys because Combination of both will be always unique

But a default index is created on primary key,
Thinking of a use case a you want to show list video that a particular profile was watching on login -> better option will be **(profileId, videoId)**

```
CREATE TABLE Orders (

    OrderID int NOT NULL,

    OrderNumber int NOT NULL,

    PersonID int,

   PRIMARY KEY (OrderID),

   FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```