**Data Modelling is the process of creating a model which stores data in database. Four steps to design the database are:**

**1. Understand the requirements.**

**Three types of Data Models below are:**

**2. Build a Conceptual Model.**

**3. Build a Logical Model.**

**4. Build a Physical Model.**

**-> Build a Conceptual Model:**

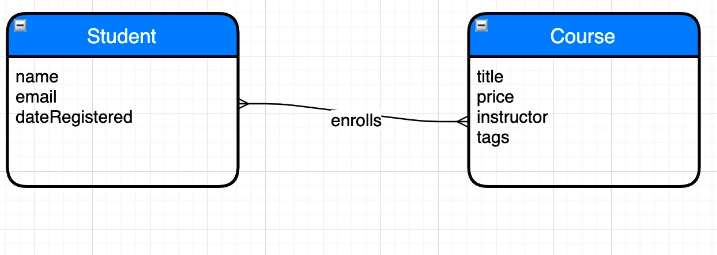
**Represents the entities and their relationships there is no structure defined here.**

**Two ways to build conceptual modelling: Entity**

1. **Relationship (ER) diagram,**
2. **UML diagrams.**

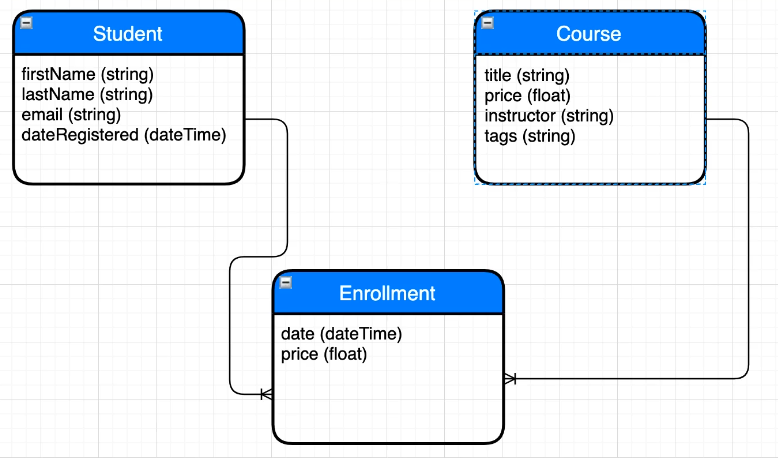
**Modelling tools are Microsoft Visio, draw.io, LucidCharts.**

**Eg. of Conceptual Model:**

****

**-> Build a Logical Model:**

**Logical model is more informative than Conceptual model in this the skeleton structure of the database is designed.**

****

**-> Build a Physical Model:**

**A physical model is the implementation of logical model with specific database technology.**

**>Table- is a container to store entities.**

**>Primary key- is the column that uniquely identify each record in the table.**

**>Composite primary key - it has multiple columns**

**>A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another table.**

**The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.**

**The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.**

**Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies.**

## **Normal Forms : All the normal forms helps us to reduce duplication and increase Data Integrity.**

## **Rules for First Normal Form**

**The first normal form expects you to follow a few simple rules while designing your database, and they are:**

#### **Rule 1: Single Valued Attributes**

**Each column of your table should be single valued which means they should not contain multiple values. We will explain this with help of an example later, let's see the other rules for now.**

**Example shown in below diagram as 1 course can have multiple tags so instead of creating single tags column, we created tags table so that single valued cells can be stored to follow 1NF.**

#### **Rule 2: Attribute Domain should not change**

**This is more of a "Common Sense" rule. In each column the values stored must be of the same kind or type.**

**For example: If you have a column dob to save date of births of a set of people, then you cannot or you must not save 'names' of some of them in that column along with 'date of birth' of others in that column. It should hold only 'date of birth' for all the records/rows.**

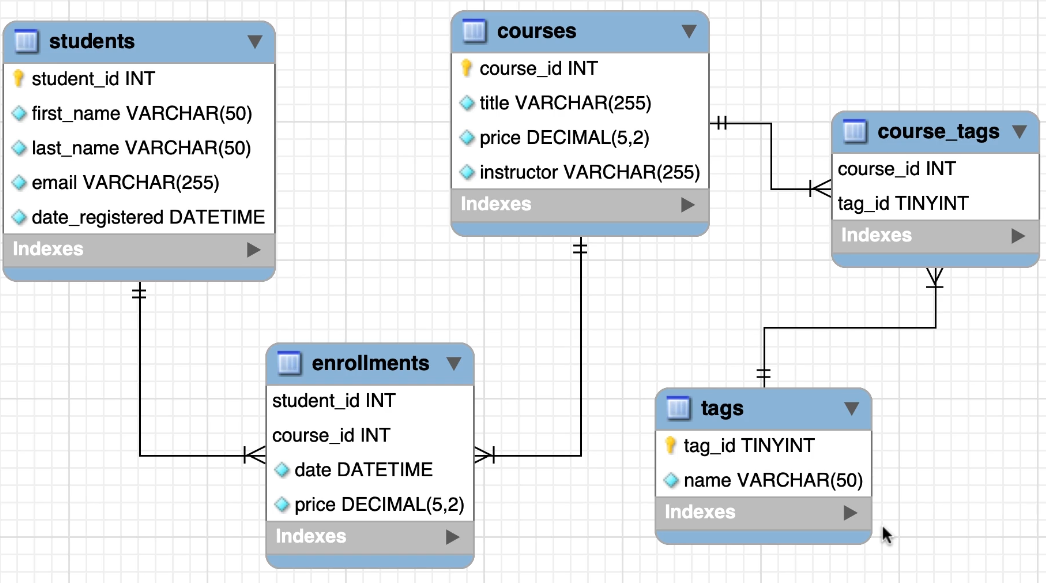
#### **Rule 3: Unique name for Attributes/Columns**

**This rule expects that each column in a table should have a unique name. This is to avoid confusion at the time of retrieving data or performing any other operation on the stored data.**

**If one or more columns have same name, then the DBMS system will be left confused.**

#### **Rule 4: Order doesn't matters**

**This rule says that the order in which you store the data in your table doesn't matter.**

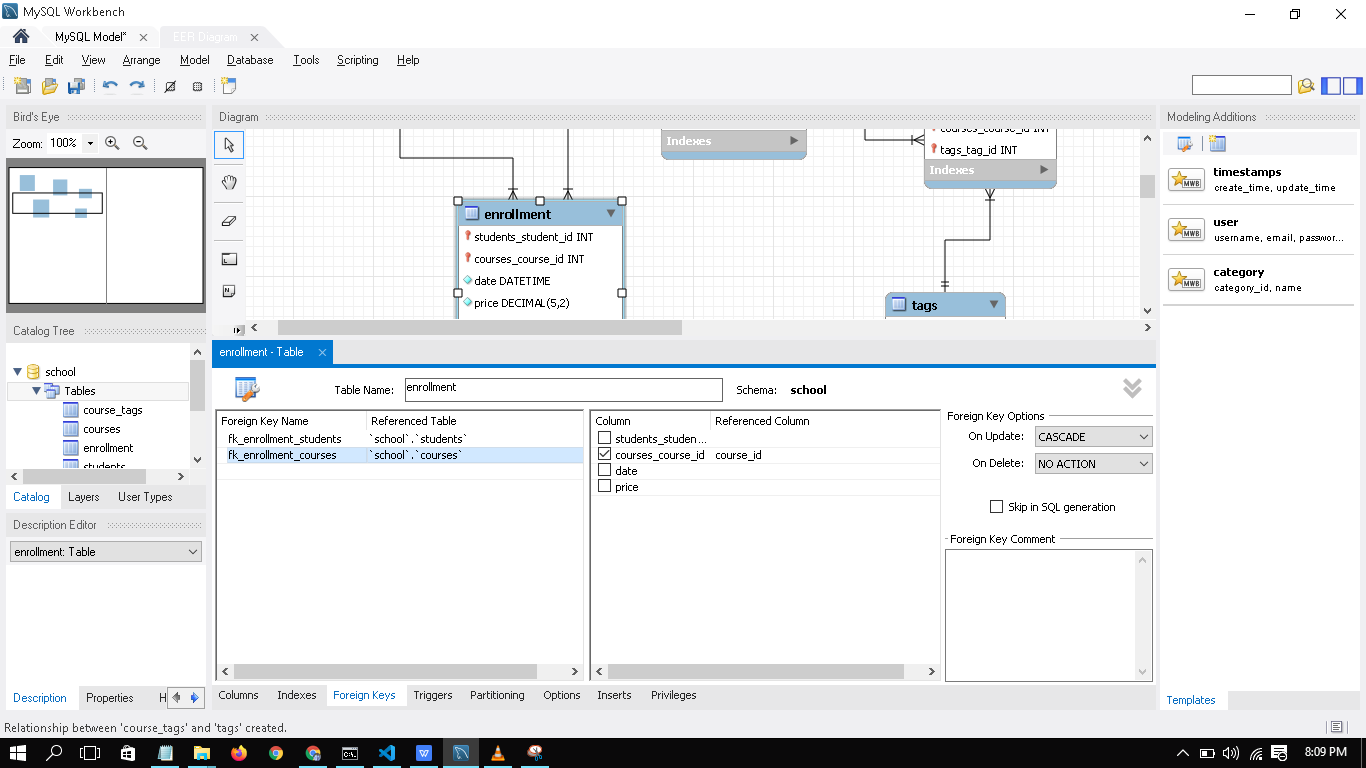
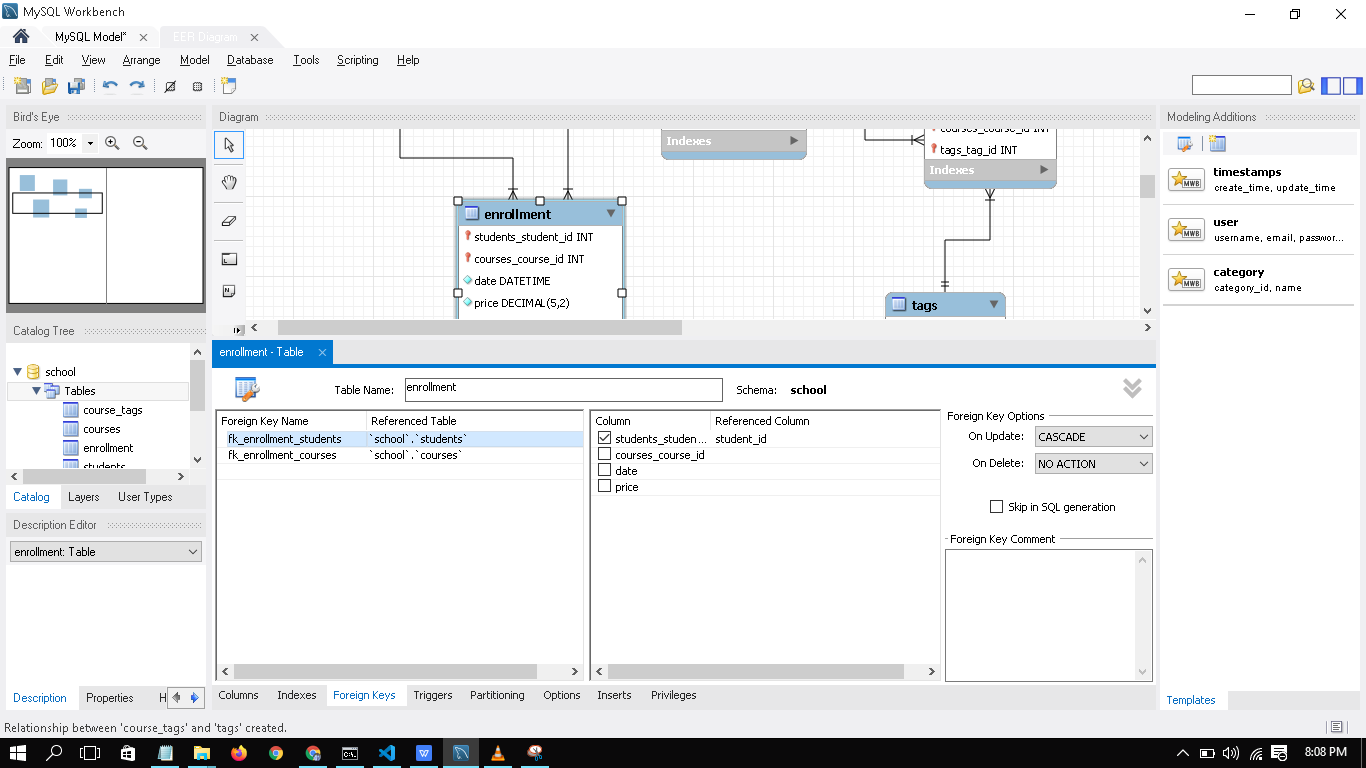
****

**As in relational databases one-to-many relationship cannot be formed so we have to build relation with two one-to-many relationships. In above diagram, enrollments and course\_tags tables are used to form one-to-many relation.**

**Student\_id and course\_id together form composite primary keys.**

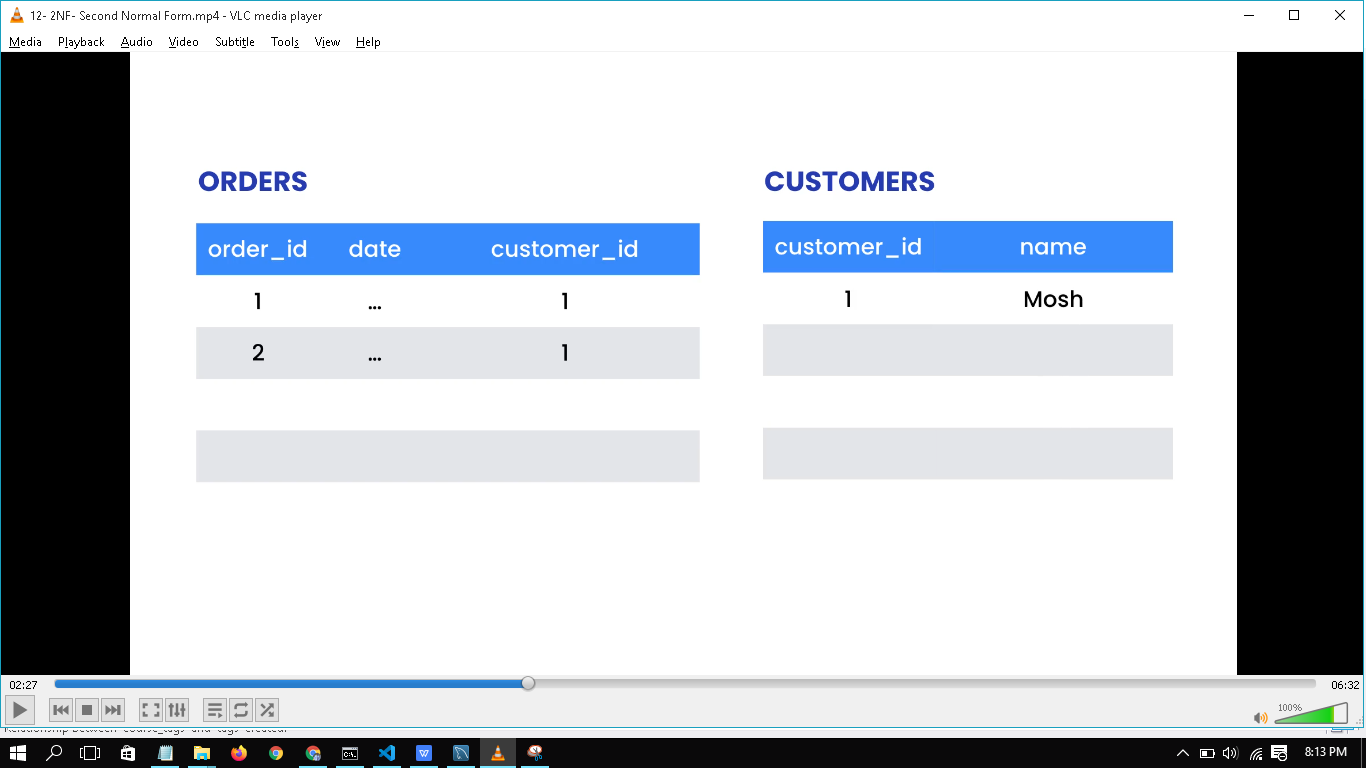
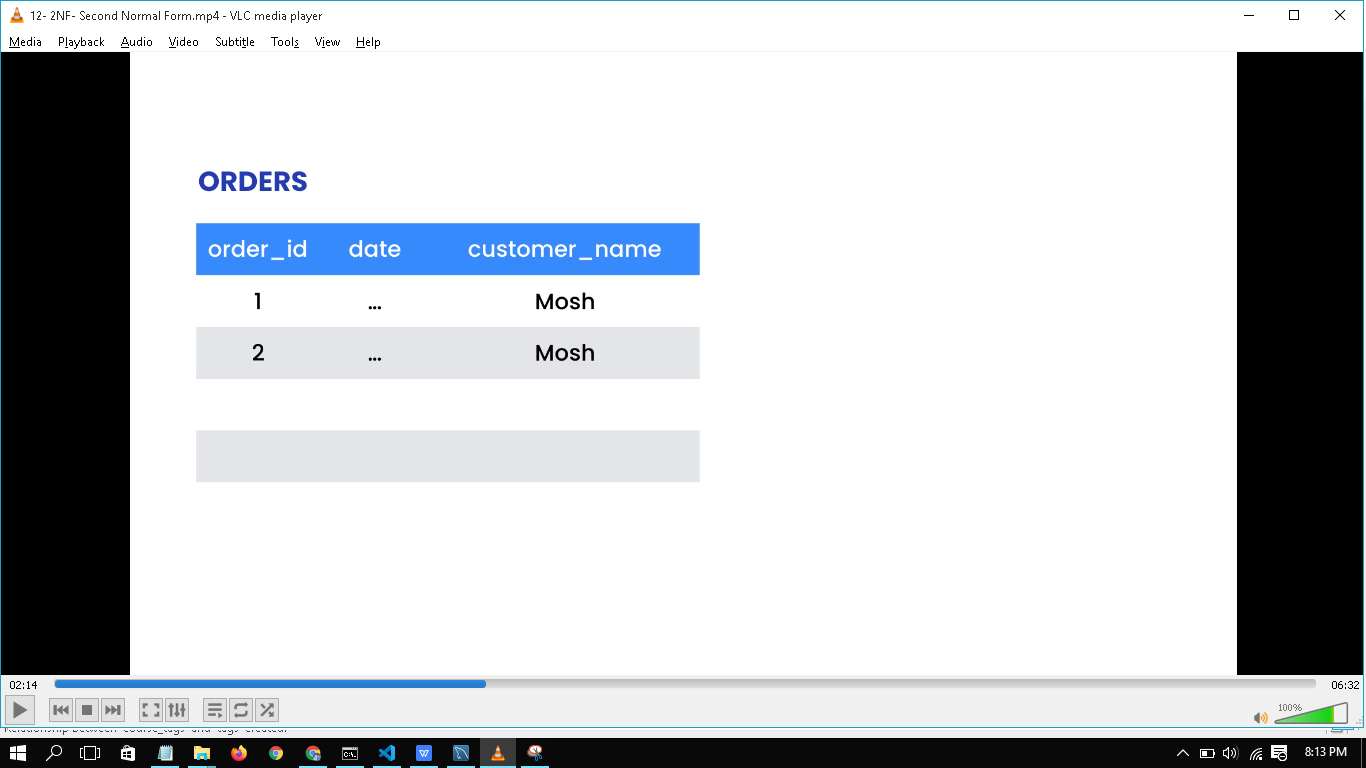
**Course\_id and tag\_id also form composite primary keys.**

**Instead of composite primary keys we can also create individual primary key in tables but using composite primary key we can remove data redundancy.**

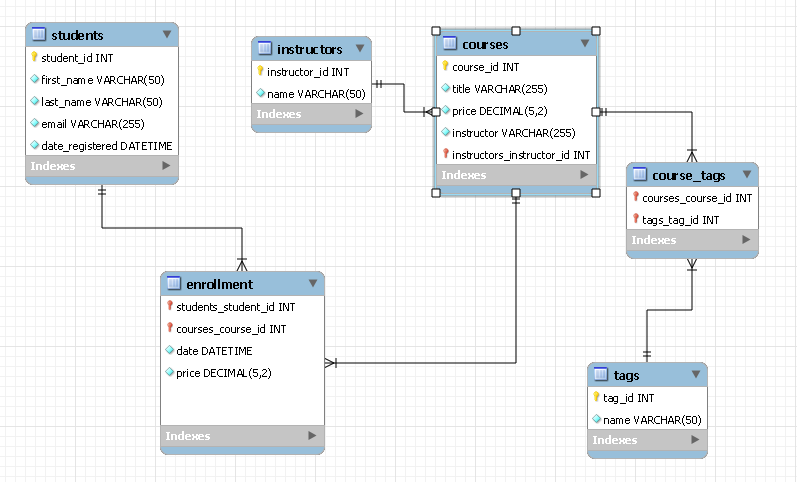
****

**Rules for Second Normal Form:**

1. **It must follow First Normal Form**
2. **Attributes/Columns must match the table. I.e shown in the below pictures. Customer name should not be present in orders table as in orders table only order detail should be present.**

****

**After following 2NF: making instructors as a new table. Since there is no need of keeping instructor column in courses table as in courses table only course related info should be there.**

****

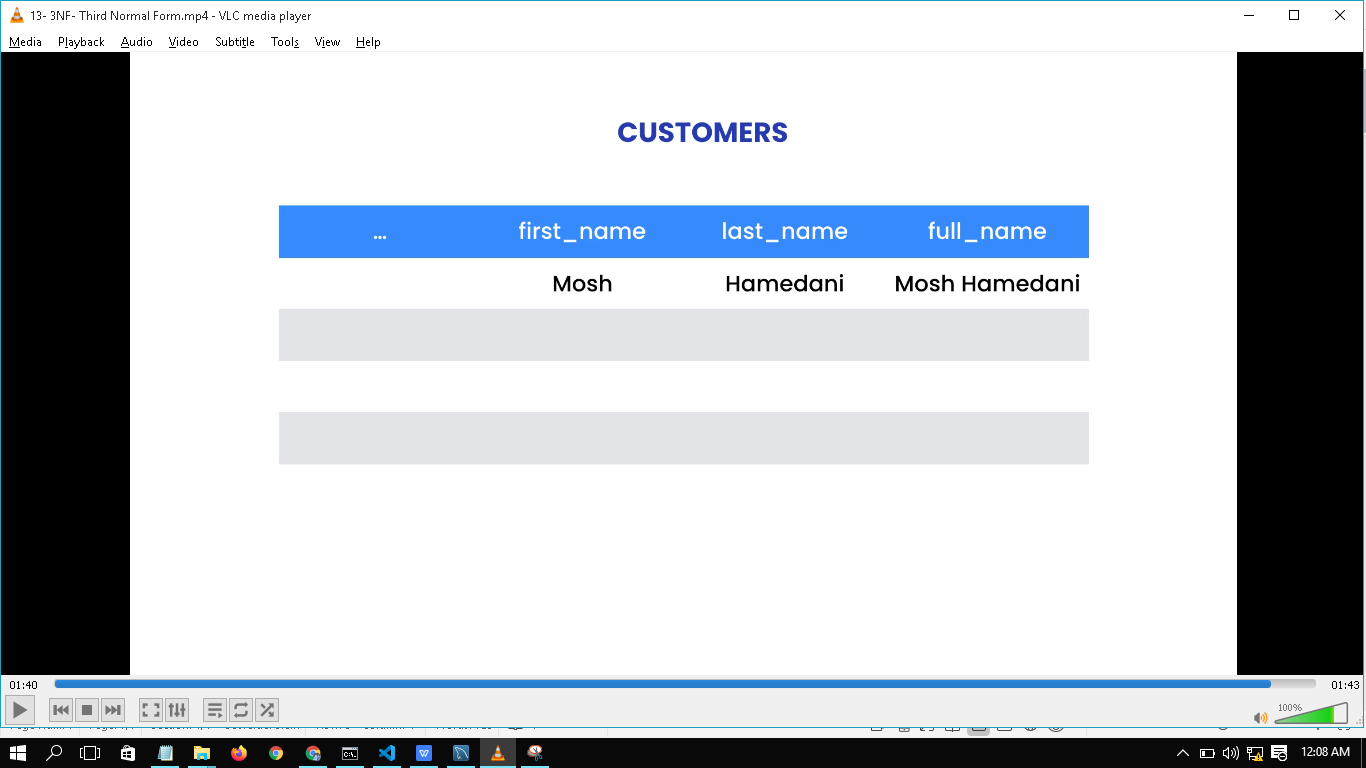
**Third Normal Form:**

**A column in the table should not be derived from other columns.**

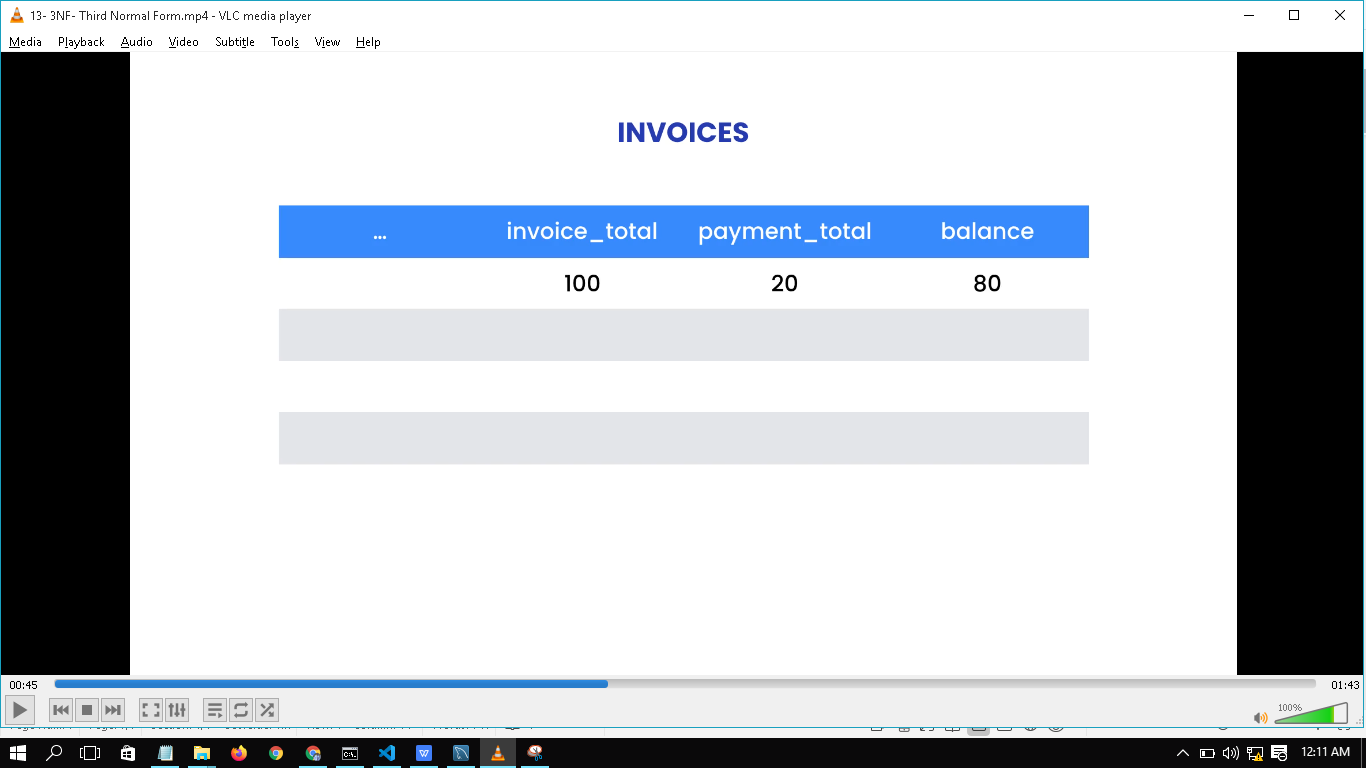
**Rules for Third Normal Form:**

1. **It must follow Second Normal Form.**

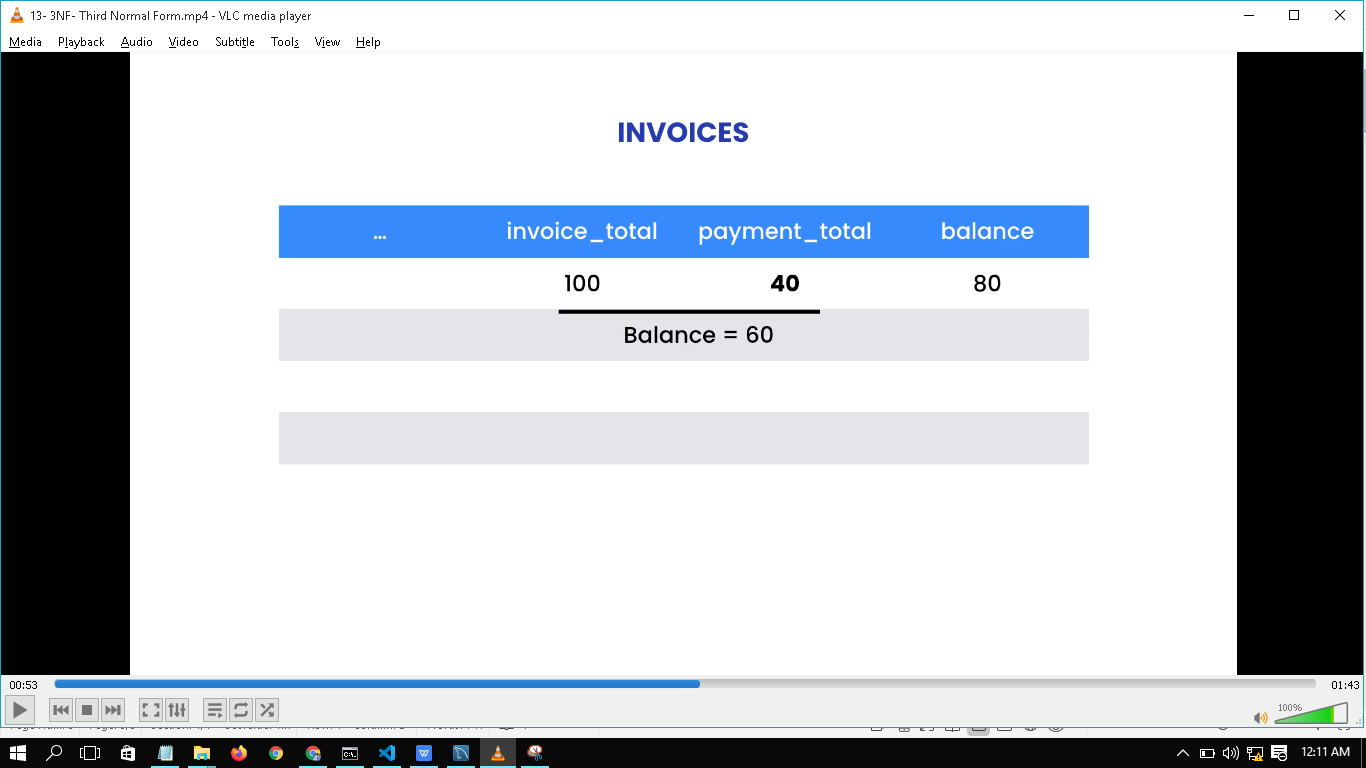
**2. A column in the table should not be derived from other columns.**

****

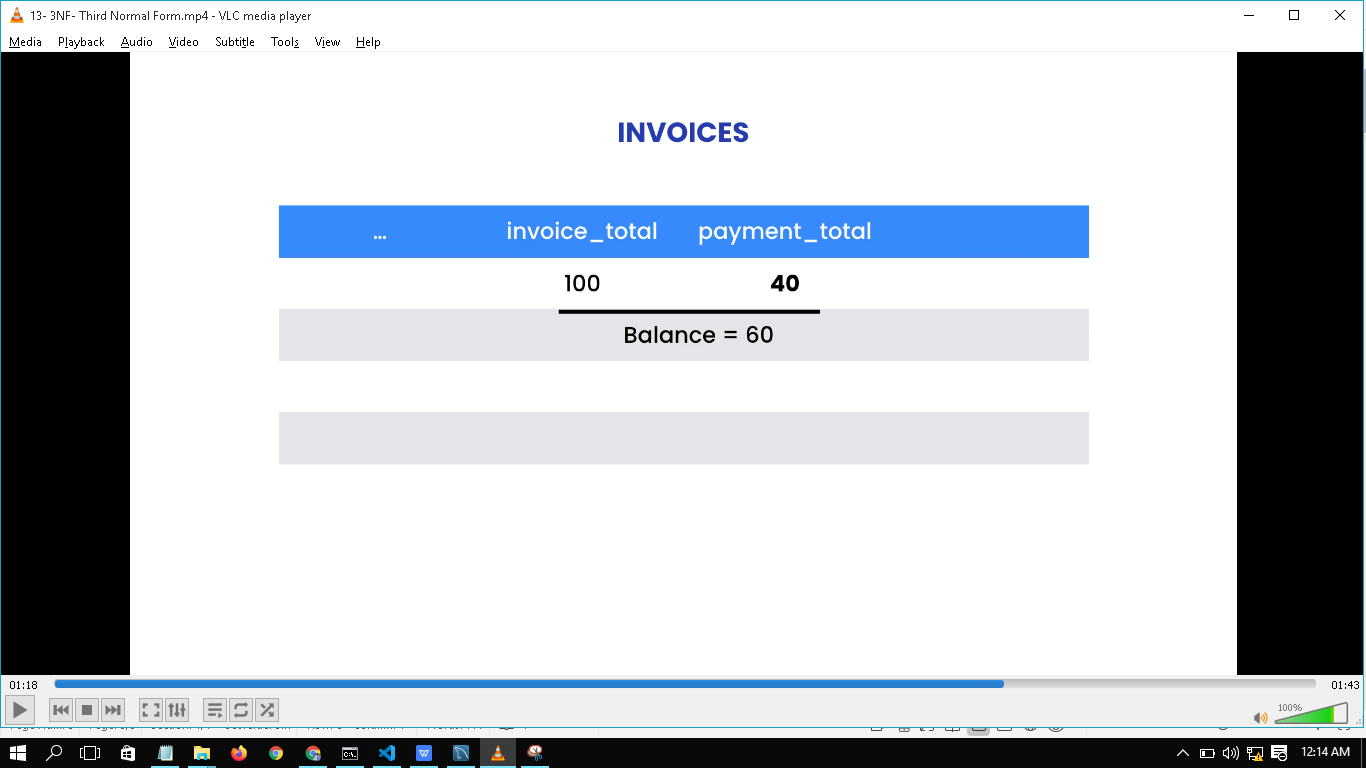
**Full\_name column should not be in table above as it goes against with the third normal form.**

****

**If payment\_total value gets changed from 20 to 40 and we forget to update balance column than it will be loss to business.**

****

**So we can drop the balance column as per the 3rd Normal Form.**

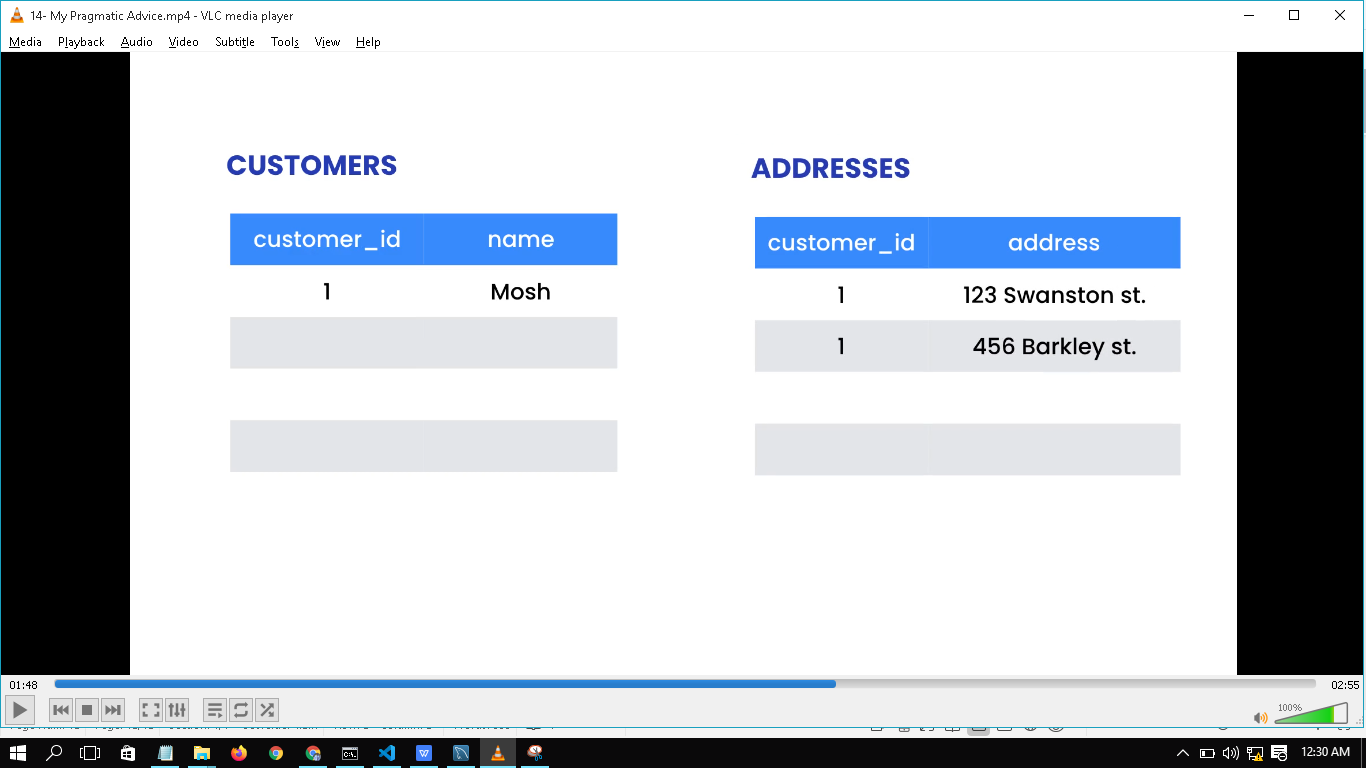
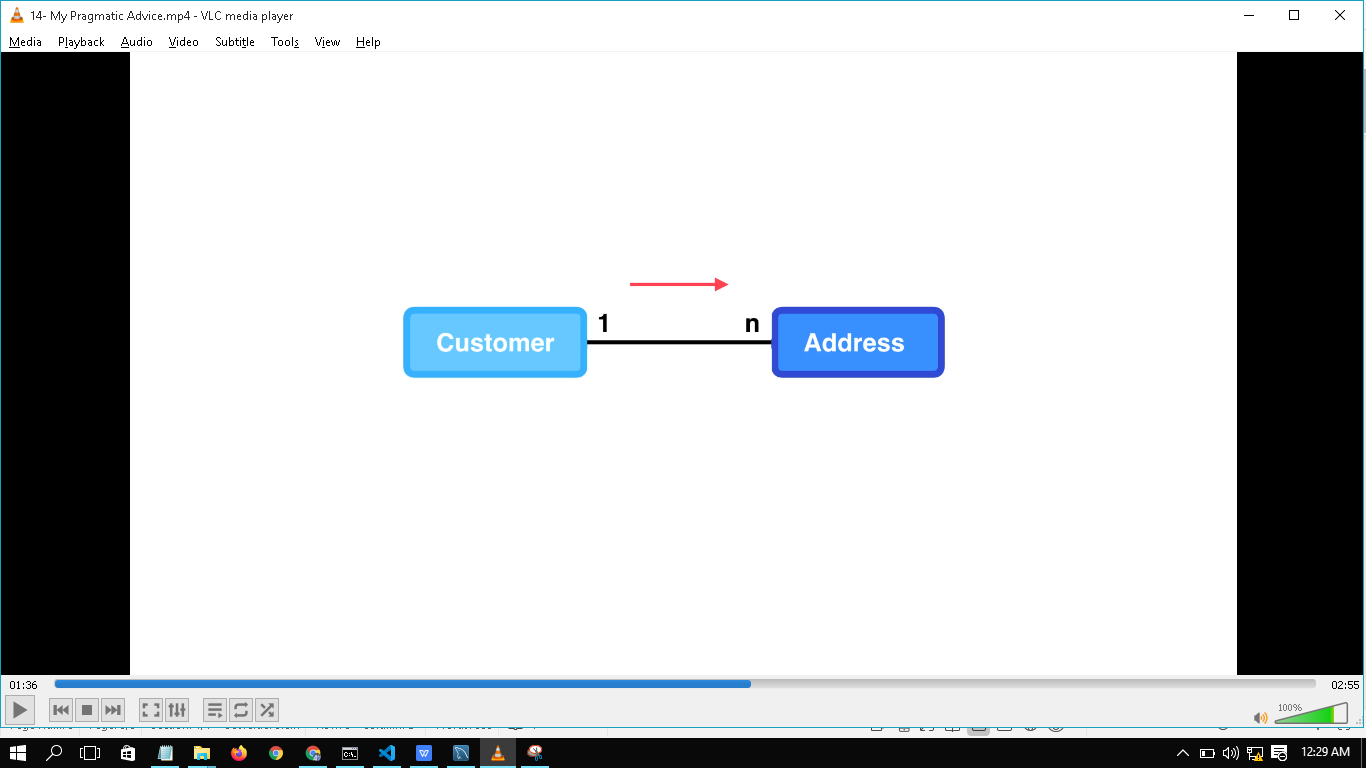
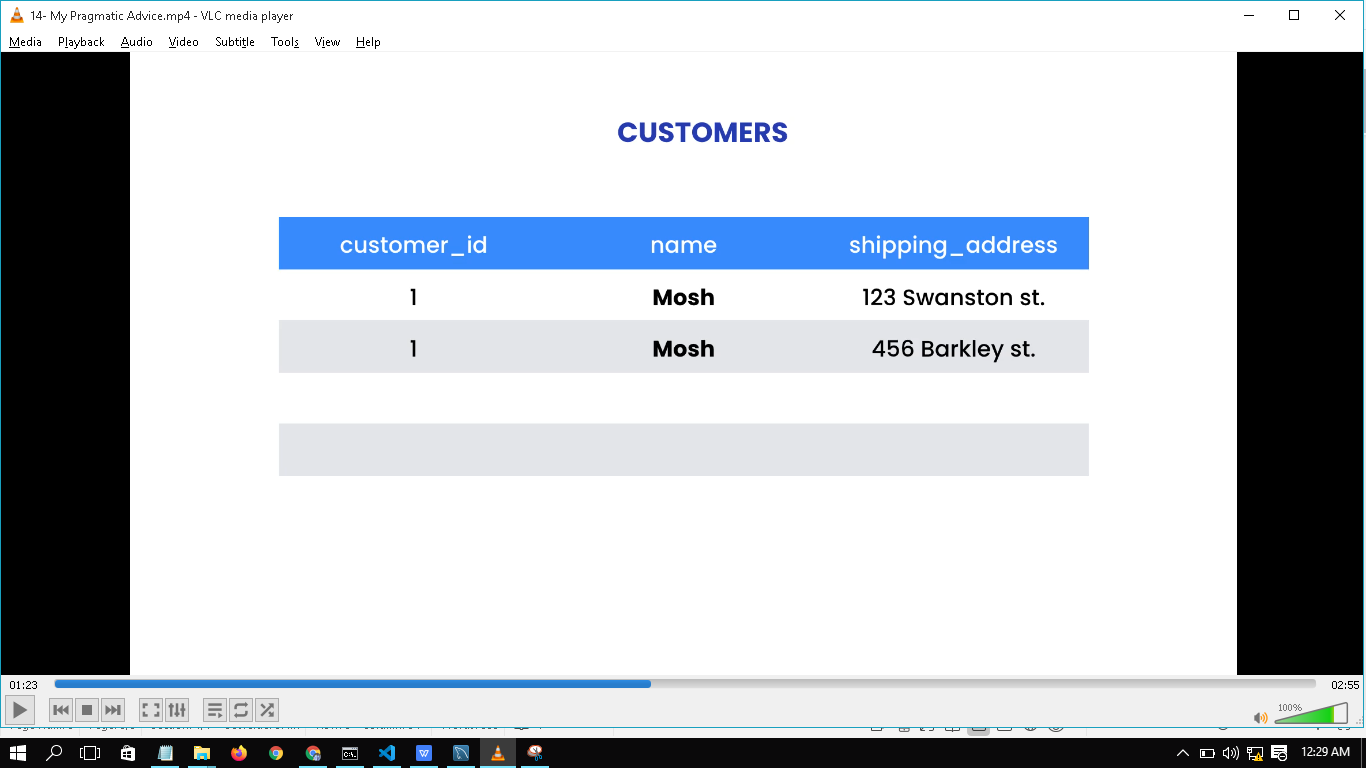
****

**NOTE:**

**>In real world, no one will ask you whether you have followed any normalization to build database. So just focus on removing data redundancy.**

**>Don’t jump directly to build database, first do conceptual or logical model and then build it real.**

**>Build only new tables to follow 3NF if there is any business needs otherwise don’t.**

****

**In the above pics there is requirement that single customer can have multiple addresses. So we made new table Addresses to follw 3NF.**

**If there is no requirement that the single customer can have multiple addresses then there is no need to create new Adresses table.**

**In MYSQL WORKBENCH,**

**TO CONVERT DATA MODEL TO THE REAL DATABASE:**

**In top menu bar, select Database -> Forward Engineer… -> click on Next,Next,Next till its done.**

**SYNCHRONICING A MODEL WITH A DATABASE:**

**In top menu bar, select Database -> Synchronize Model… -> click on Next,Next,Next till its done.**

**TO CONVERT A REAL DATABASE TO A DATA MODEL:**

**In top menu bar, select Database -> Reverse Engineer… -> click on Next,Next,Next till its done.**

**Data Model is used to easily visualize all the nerves of the databases.**

**Design database for Flight booking portal:**

