# Exercises: Code-First

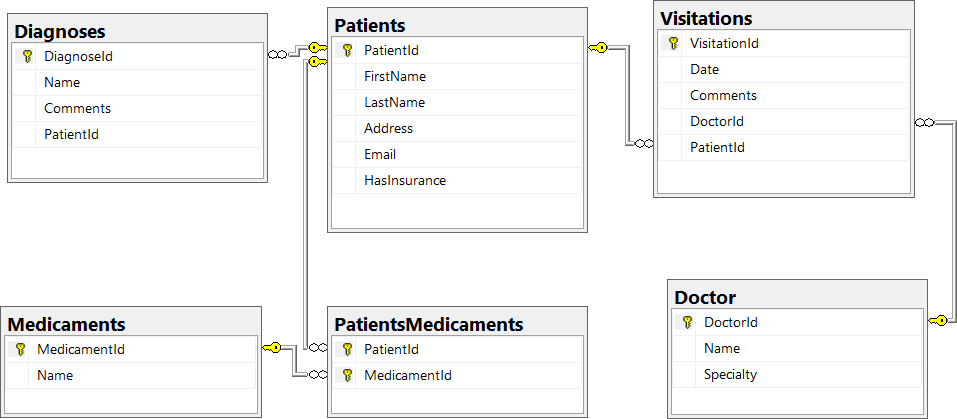
This document defines the **exercise assignments** for the [Databases Advanced - Entity Framework course @ SoftUni](https://softuni.bg/trainings/4234/entity-framework-core-october-2023)  
You can check your solutions in [Judge](https://judge.softuni.org/Contests/3923/Code-First)

## Hospital Database

You went to your GP for your annual exam and you told him that you've started working as a Junior Database App Developer. It turned out he was looking for someone to make an app, which he could use to manage and store data about his patients.

Your task is to design a database using the **Code** **First** approach. The GP needs to keep information about their **patients**. Each patient has **first name**, **last name**, **address**, **email**, information whether he has **medical** **insurance** or not, and should keep history about all his **visitations**, **diagnoses** and **prescribed** **medicaments**. Each visitation has **date** and **comments**. Each **diagnose** has **name** and **comments** for it. Each **medicament** has **name**. **Validate** all data before inserting it in the database.

Your Database should look something like this:



### Constraints

Your **namespaces** should be:

* P01\_HospitalDatabase – for your **StartUp** class, if you have one
* P01\_HospitalDatabase.Data – for your **DbContext**
* P01\_HospitalDatabase.Data.Models – for your models

Your **classes** should be:

* HospitalContext – your DbContext
* Patient
  + **PatientId**
  + **FirstName** (up to **50** characters, unicode)
  + **LastName** (up to **50** characters, unicode)
  + **Address** (up to **250** characters, unicode)
  + **Email** (up to **80** characters, not unicode)
  + **HasInsurance**
* Visitation
  + **VisitationId**
  + **Date**
  + **Comments** (up to **250** characters, unicode)
  + **Patient**
* Diagnose
  + **DiagnoseId**
  + **Name** (up to 50 characters, unicode)
  + **Comments** (up to 250 characters, unicode)
  + **Patient**
* Medicament
  + **MedicamentId**
  + **Name** (up to **50** characters, unicode)
* PatientMedicament – mapping class between **Patients** and **Medicaments**

The **collections** of mapping classes (ICollection<PatientMedicament>) must be named **Prescriptions**!

**Note**: Don't forget to remove the **Tools** package before uploading your project to Judge, if you have used it!

### \* Bonus Task

Make a console-based user interface, so the doctor can easily use the database.

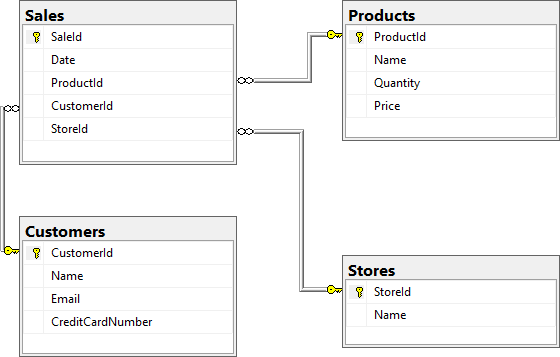
## Hospital Database Modification

Your GP bragged around in the hospital about the cool software you made for him. Now the hospital administration wants to modify your program so they can use it, too. They want to store information about the **Doctors** (**Name** and **Speciality**). Each doctor can perform **many** **Visitations**. Make the necessary changes in the **database** to satisfy the new needs of the hospital administration.

### Constraints

Keep the **namespaces** fromthe previous task and only add the class **Doctor** and change the class **Visitation** accordingly. The doctor's **name** and **specialty** should be up to **100** **characters** long, **unicode**.

## Sales Database

Create a database for storing data about sales using the Code First approach. The database should look like this:  
  


### **Constraints**

Your **namespaces** should be:

* P03\_SalesDatabase
* P03\_SalesDatabase.Data
* P03\_SalesDatabase.Data.Models

Your **classes** should be:

* SalesContext – your DbContext
* Product
  + **ProductId**
  + **Name** (up to **50** characters, unicode)
  + **Quantity** (real number)
  + **Price**
  + **Sales**
* Customer
  + **CustomerId**
  + **Name** (up to **100** characters, unicode)
  + **Email** (up to **80** characters, not unicode)
  + **CreditCardNumber** (**string**)
  + **Sales**
* Store
  + **StoreId**
  + **Name** (up to **80** characters, unicode)
  + **Sales**
* Sale
  + **SaleId**
  + **Date**
  + **Product**
  + **Customer**
  + **Store**

### \*Bonus Task

Write a **seed method** that fills the database with sample data (randomly generated).