Project: Dental clinic database.

Course: Database Systems

Instructor:

Msc. Anxhela Baraj

By:

Xhoni Shega,

Marvin Rexhepagaj,

Noelia Karameta,

Laerto Xhuti,

Leandro Pasha.

***1. Enterprise description:***

**a) Give a short description of the enterprise.**

Dental clinic database system involves the development of a database to manage dentist’s information. This system is designed to protect the privacy of the patients. It manages all information about clinics, appointments, patient treatments, products etc. The purpose of the project is to store the data correctly and to reduce the annual work of managing the clinic. It tracks all details about patients, treatments, appointments and invoices.

**b) Describe the functions that the system will perform.**

The goal of this project is to create a database for an existing business, a dental clinic, in the field of medicine for the daily use of the staff. The database will be based on all registrations made in the clinic, all patient treatments and the invoices. The system will display the main page where employees can see all appointments for the day, view all information about patients and all products offered by the clinic, and also book appointments.

**c) Define the end users.**

The end users of this system will be the dentists of the clinic. This system will let the dentists view their appointments, patient’s information and all their treatmetns. Some other end users would be the employees but they will not be able to have a full access to the system as the system was planned to be more for dentists work. The employees can view the products available in the clinic only.

**d) Define the Business Rules based on the above descriptions.**

The clinic sets up a log in form for employees to enter to their profile. An employee ID has one employee, and each employee has only one employee id.

The clinics have their products. So, every clinic has many products, and each product has only one clinic.

When the employees log in, some employees can be dentists. If they are dentists, they can only have only one single emplyee ID.

A dentist is scheduled for many appointments, and each appointment is scheduled with exactly 1 doctor.

A patient scheduels many appointments. Each appointment is scheduled by one patient.

A patient makes many patient treatments, and each patient treatment is made for one patient.

One treatment has many patient treatments, and each patient treatment can have only one treatment.

Many treatments can have one or zero prescription, and each percription can have many treatments or zero.

A treatments generates many invoice treatments, and each invoice treatment is generated by only one treatment.

Many invoice treatments apply only one invoice, and a single invoice applies many invoice treatments.

***2. Technical description.***

**Describe all the technologies used.**

There were several requirements for the project. First, the final product must be available on a Windows PC, as this is the default operating system used by the team members. Second, it must be fully compatible with desktop and smartphone so that the website can be accessed by employees with different devices.

The software/applications used by the team are:

Draw - ERD

Netbeans - Java

GUI

SQLServer - database implementation

Draw.io – normalization

***3.Entity Relationship Design:***

**a) Describe your entities. Define the strong and weak entities:**

**Product:** Product is a strong entity which contains all the necessary information about products that are needed for the clinic in order to be operational. The PRODUCT\_ID is the primary key for this entity.

**Clinic:** Clinic is a strong entity which contains the information about the clinic like the address, phone number, etc. The primary key for this entity is CLINIC\_ID.

**Employee:** Employee is a strong entity which contains all the required information about the employee of the clinic. The primary key for this entity is EMPLOYEE\_ID. Th

**Login\_Employee:** Login\_Employee is a weak entity which contains the information about login credentials that an employee needs to have in order to have certain access to the system. The primary of this entity is a composed primary key composed of EMPLOYEE\_USERNAME and EMPLOYEE\_ID.

**Dentist:** Dentist is a strong entity which contains the information about the dentist the role of the dentist on the clinic. The primary key for this entity is DENTIST\_ID.

**Appointment:** Appointment is a strong entity which contains the necessary information about an appointment. The primary key for this entity is Appointment\_ID.

**Patient:** Patient is a strong entity which contains the necessary information about the patient. The primary for this entity is PATIENT\_PERSONAL\_NUMBER.

**Patient\_Treatment:** Patient\_Treatment is a weak entity which serves as a table for the connection of the patient with the treatment. The primary key on this entity is composed of PATIENT\_PERSONAL\_NUMBER and TREATMENT\_ID.

**Treatment:** Treatment is a strong entity which contains the information about the treatment. The primary key for this entity is TREATMENT\_ID.

**Perscription:** Perscription is a weak entity which is existence is dependent on the treatment that the patient will take. The primary key for this entity is composed of PERSCRIPTION\_ID and TREATMENT\_ID.

**Invoice\_Treatment:** Invoice\_Treatment is a weak entity which is existence dependent on the invoice table and treatment table. The primary key is composed of INVOICE\_ID and TREATMENT\_ID.

**Invoice:** Invoice is a strong entity which contains the information about the invoice. The primary key of this entity is INVOICE\_ID.

**b) Describe the relationships between tables and their types.**

Clinic table is related with the product table, the type of the relationship is one-to-many where the entity product is the many part of the relationship and the table clinic is the one part of the relationship, the foreign key is CLINIC\_ID;

Clinic is also related with the Employee table, the type of the relationship is one-to-many to many relationship where the entity employee is the many part of the relationship and the entity clinic is the one part of the relationship, the foreign key is CLINIC\_ID;

Employee table is related with Login\_Employee table, the type of the relationship is one-to-one relationship, the foreign key is stored in Login\_Employee table since it is a weak entity and is existence dependent on the employee table, it is EMPLOYEE\_ID;

Employee is also related with the table dentist on a one-to-one relationship optional which means the an employee of the clinic optionally can be a dentist. The foreign key is stored on the dentist table and it is EMPLOYEE\_ID;

Dentist table is related with appointment table on a one-to-many relationship, where the Dentist is the one part of the relationship and the appointment is the many part of the relationship, the foreign key is stored on appointment table and it is DENTIST\_ID;

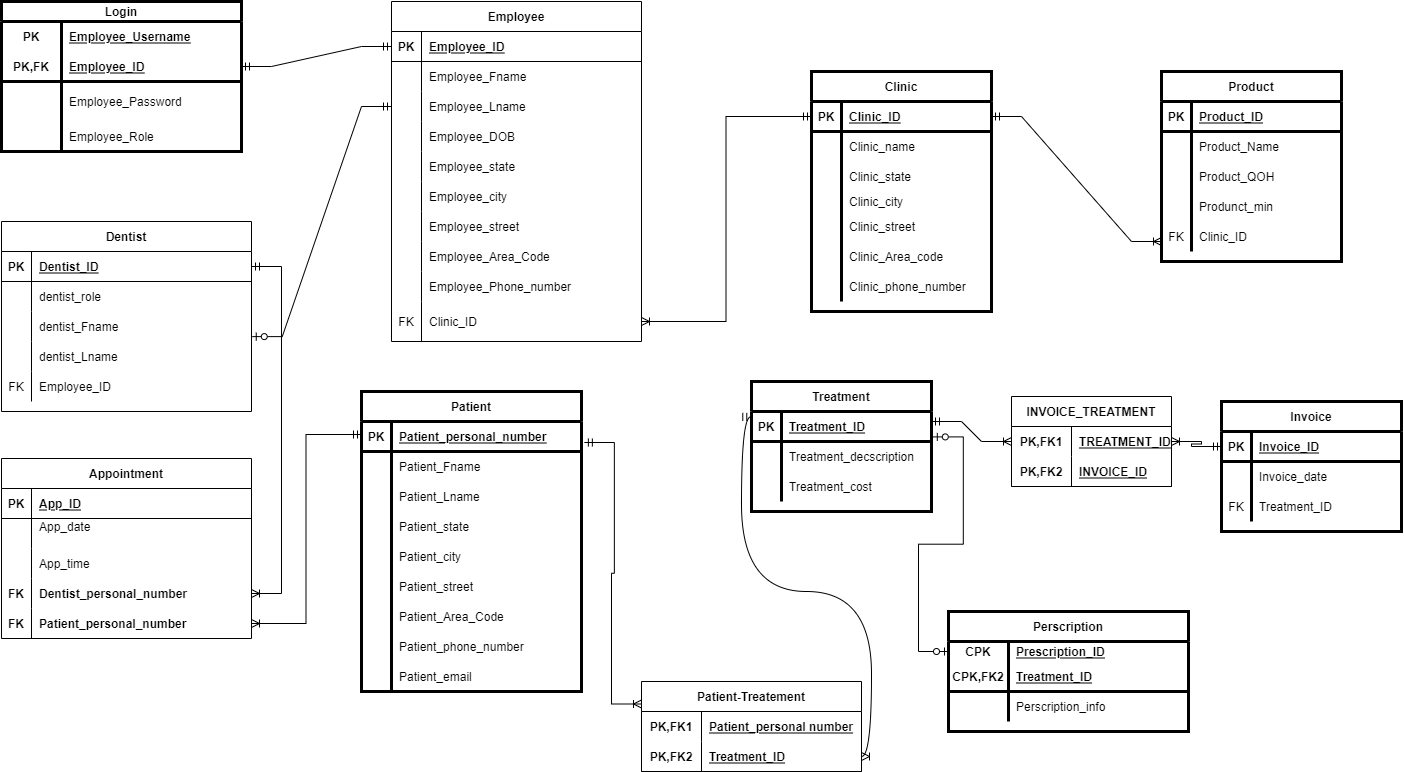
Appointment table is related with the patient table on a one-to-many relationship where patient is the one part of the relationship and the appointment is the many part of the relationship where the foreign key is stored which is, PATIENT\_PERSONAL\_NUMBER;

Patient table is related with the treatment table on a many-to-many relationship which is broken on two one-to-many relationship by using the table patient\_treatment.

Treatment table is also related with the prescription table on a one-to-one relationship optional where the foreign key is stored on the prescription table since it is a weak entity existence dependent on the treatment table.

Treatment table is also related with the invoice table on a many-to-many relationship which is broken down on two one-to-many relationships by using the table invoice\_treatment.

**c) Draw the Crow’s Foot ER Diagram for your database.**

****

**In the project file will also provide a xml file which can open the diagram on draw.io**

***4.*** ***4. Normalization:***

**Each base table MUST be in 3NF and must have the following information associated with it:**

**a) The Primary Keys.**

**b) A list of Foreign Keys.**

**c) A list of and functional dependencies that hold.**

UNF:

Clinic(clinic\_id, clinic\_state, clinic\_city, clinic\_are\_code, clinic\_phone\_number);

Product(product\_name, product\_QOH, Product\_min);

Dentist(Dentist\_id, dentist\_role, employee\_id, employee\_f\_name, employee\_l\_name, employee\_DOB, employee\_state, employee\_city, employee\_street, employee\_area\_code, employee\_phone\_number);

Appointment(Appointment\_ID, appointment\_time);

Patient\_Treatment(Patient\_personal\_number, Patient\_fname, Patient\_lname, Patient\_state, Patient\_city, Patient\_street, Patient\_area\_code, Patient\_phone\_number, Patient\_email, Treatement\_id, Treatment\_description, Treatment\_cost);

Invoice\_Treatment(Treatment\_id, Treatment\_description, Treatment\_cost, Invoice\_id, invoice\_date);

Perscription(Perscription\_id, Treatment\_id, prescription\_info);

1NF:

Clinic(**clinic\_id**, clinic\_state, clinic\_city,clinic\_street, clinic\_are\_code, clinic\_phone\_number);

Product(**product\_id**,product\_name, product\_QOH, Product\_min, product\_reorder,***clinic\_id***);

Dentist(**Dentist\_id**, dentist\_role, **employee\_id**, employee\_f\_name, employee\_l\_name, employee\_DOB, employee\_state, employee\_city, employee\_street, employee\_area\_code, employee\_phone\_number);

Appointment(**Appointment\_ID**, appointment\_time,***Dentist\_id***, ***Patient\_personal\_number***);

Patient\_Treatment(**Patient\_personal\_number**, Patient\_fname, Patient\_lname, Patient\_state, Patient\_city, Patient\_street, Patient\_area\_code, Patient\_phone\_number, Patient\_email, **Treatement\_id**, Treatment\_description, Treatment\_cost);

Invoice\_treatment(**Invoice\_id, Treatment\_id**)

Invoice(**Invoice\_id**, invoice\_date);

Perscription(**Perscription\_id**, ***Treatment\_id***, prescription\_info);

We have partial dependencies between:

Employee\_id 🡪 employee\_f\_name, employee\_l\_name, employee\_DOB, employee\_state, employee\_city, employee\_street, employee\_area\_code, employee\_phone\_number;

Dentist\_id 🡪 Dentist\_role, Employee\_id;

Patient\_personal\_number 🡪 Patient\_fname, Patient\_lname, Patient\_state, Patient\_city, Patient\_street, Patient\_area\_code, Patient\_phone\_number, Patient\_email;

**Treatement\_id**🡪 Treatment\_description, Treatment\_cost;

2NF and 3NF

We break the partial dependencies and we have:

Clinic(**clinic\_id**, clinic\_state, clinic\_city,clinic\_street, clinic\_are\_code, clinic\_phone\_number);

Product(**product\_id**,product\_name, product\_QOH, Product\_min, product\_reorder,***clinic\_id***);

Dentist(**Dentist\_id**, dentist\_role);

Employee(, **employee\_id**, employee\_f\_name, employee\_l\_name, employee\_DOB, employee\_state, employee\_city, employee\_street, employee\_area\_code, employee\_phone\_number);

Appointment(**Appointment\_ID**, appointment\_time,***Dentist\_id***, ***Patient\_personal\_number***);

Patient\_Treatment(**Patient\_personal\_number**, **Treatement\_id**);

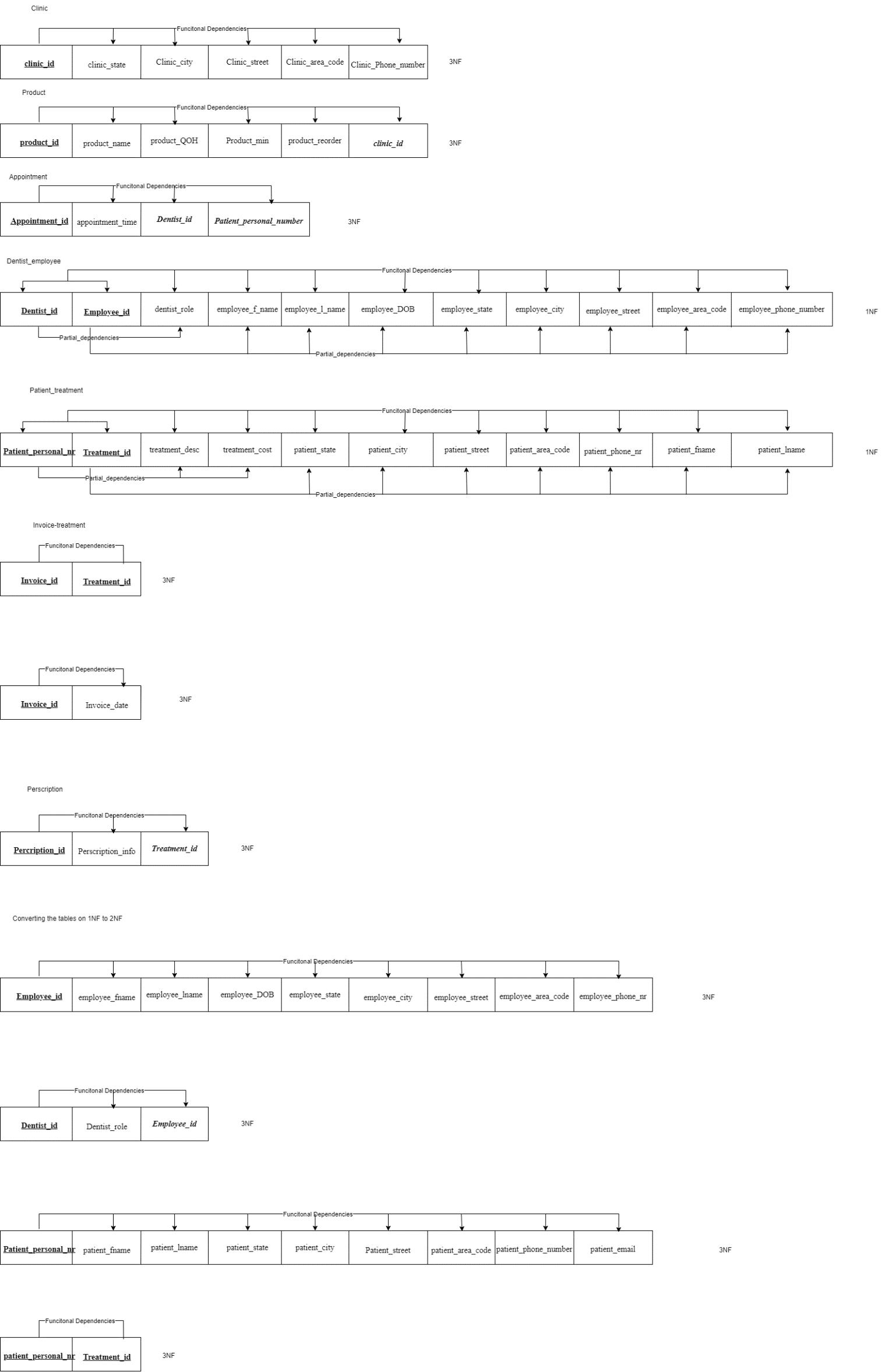
Patient(**Patient\_personal\_number**, Patient\_fname, Patient\_lname, Patient\_state, Patient\_city, Patient\_street, Patient\_area\_code, Patient\_phone\_number, Patient\_email,)

Treatment(**Treatement\_id**, Treatment\_description, Treatment\_cost)

Invoice\_treatment(**Invoice\_id, Treatment\_id**)

Invoice(**Invoice\_id**, invoice\_date);

Perscription(**Perscription\_id**, ***Treatment\_id***, prescription\_info);



***5. Data Dictionary:***



***It is also found as an excel spreadsheet on the folder of the project***

***6. Implement your tables in SQL SERVER Management Studio or any RDBMS you***

***are familiar to.***

CREATE DATABASE DENTAL\_CLINIC;

CREATE TABLE CLINIC(

CLINIC\_ID INTEGER NOT NULL,

CLINIC\_NAME VARCHAR(50) NOT NULL,

CLINIC\_STATE VARCHAR(20) NOT NULL,

CLINIC\_CITY VARCHAR(20) NOT NULL,

CLINIC\_STREET VARCHAR(20) NOT NULL,

CLINIC\_AREA\_CODE INTEGER NOT NULL,

CLINIC\_PHONE\_NUMBER INTEGER NOT NULL,

PRIMARY KEY(CLINIC\_ID)

); --End of table clinic

CREATE TABLE PRODUCT(

PRODUCT\_ID INTEGER NOT NULL,

PRODUCT\_NAME VARCHAR(50) NOT NULL,

PRODUCT\_QOH INTEGER NOT NULL,

PRODUCT\_MIN INTEGER NOT NULL,

PRODUCT\_REORDER INTEGER NOT NULL,

CLINIC\_ID INTEGER NOT NULL,

PRIMARY KEY(PRODUCT\_ID),

FOREIGN KEY (CLINIC\_ID) REFERENCES CLINIC

);

CREATE TABLE EMPLOYEE(

EMPLOYEE\_ID INTEGER IDENTITY(1,1),

EMPLOYEE\_F\_NAME VARCHAR(50) NOT NULL,

EMPLOYEE\_L\_NAME VARCHAR (50) NOT NULL,

EMPLOYEE\_DOB DATE NOT NULL,

EMPLOYEE\_STATE VARCHAR(20) NOT NULL,

EMPOYEE\_CITY VARCHAR(20) NOT NULL,

EMPLOYEE\_STREET VARCHAR(20) NOT NULL,

EMPLOYEE\_AREA\_CODE INTEGER NOT NULL,

EMPLOYEE\_PHONE\_NUMBER INTEGER NOT NULL,

CLINIC\_ID INTEGER NOT NULL,

PRIMARY KEY(EMPLOYEE\_ID),

FOREIGN KEY(CLINIC\_ID) REFERENCES CLINIC

); --End of table employee

CREATE TABLE LOGIN\_EMPLOYEE(

EMPLOYEE\_USERNAME VARCHAR(50) NOT NULL,

EMPLOYEE\_ID INTEGER NOT NULL,

EMPLOYEE\_PASSWORD VARCHAR(30) NOT NULL,

EMPLOYEE\_ROLE VARCHAR(20) NOT NULL,

PRIMARY KEY(EMPLOYEE\_USERNAME, EMPLOYEE\_ID),

FOREIGN KEY(EMPLOYEE\_ID) REFERENCES EMPLOYEE

ON DELETE CASCADE

); --End of table login

CREATE TABLE DENTIST(

DENTIST\_ID INTEGER IDENTITY(1,1),

DENTIST\_ROLE VARCHAR NOT NULL,

EMPLOYEE\_ID INTEGER ,

PRIMARY KEY(DENTIST\_ID),

FOREIGN KEY(EMPLOYEE\_ID) REFERENCES EMPLOYEE

);--End of table dentist

CREATE TABLE PATIENT(

PATIENT\_PERSONAL\_NUMBER INTEGER NOT NULL,

PATIENT\_FIRST\_NAME VARCHAR(50),

PATIENT\_LAST\_NAME VARCHAR(50),

PATIENT\_STATE VARCHAR(20) NOT NULL,

PATIENT\_CITY VARCHAR(20) NOT NULL,

PATIENT\_STREET VARCHAR(20) NOT NULL,

PATIENT\_AREA\_CODE INTEGER NOT NULL,

PATIENT\_PHONE\_NUMBER INTEGER NOT NULL,

PATIENT\_EMAIL VARCHAR(50),

PRIMARY KEY (PATIENT\_PERSONAL\_NUMBER)

);--End of table patient

CREATE TABLE APPOINTMENT(

APPOINTMENT\_ID INTEGER IDENTITY(1,1),

APPOINTMENT\_TIME DATE NOT NULL,

DENTIST\_ID INTEGER NOT NULL,

PATIENT\_PERSONAL\_NUMBER INTEGER NOT NULL,

PRIMARY KEY(APPOINTMENT\_ID),

FOREIGN KEY(DENTIST\_ID) REFERENCES DENTIST,

FOREIGN KEY(PATIENT\_PERSONAL\_NUMBER) REFERENCES PATIENT

);--End of table appointment

CREATE TABLE TREATMENT(

TREATMENT\_ID INTEGER IDENTITY(1,1),

TREATMENT\_DESCRIPTION VARCHAR(100) NOT NULL,

TREATMENT\_COST INTEGER NOT NULL,

PRIMARY KEY(TREATMENT\_ID)

); --End of table treatment

CREATE TABLE PATIENT\_TREATMENT(

PATIENT\_PERSONAL\_NUMBER INTEGER NOT NULL,

TREATMENT\_ID INTEGER NOT NULL,

PRIMARY KEY(PATIENT\_PERSONAL\_NUMBER, TREATMENT\_ID),

FOREIGN KEY (PATIENT\_PERSONAL\_NUMBER) REFERENCES PATIENT,

FOREIGN KEY(TREATMENT\_ID) REFERENCES TREATMENT

ON DELETE CASCADE

); --End of table patient\_treatment

CREATE TABLE PRESCRIPTION(

PRESCRIPTION\_ID INTEGER IDENTITY(1,1),

TREATMENT\_ID INTEGER,

PRESCRIPTION\_INFO VARCHAR(200),

PRIMARY KEY(PRESCRIPTION\_ID, TREATMENT\_ID),

FOREIGN KEY (TREATMENT\_ID) REFERENCES TREATMENT

ON DELETE CASCADE

);

CREATE TABLE INVOICE(

INVOICE\_ID INTEGER IDENTITY(1,1),

INVOICE\_DATE DATE NOT NULL,

PRIMARY KEY(INVOICE\_ID)

);

CREATE TABLE INVOICE\_TREATMENT(

INVOICE\_ID INTEGER NOT NULL,

TREATMENT\_ID INTEGER NOT NULL,

PRIMARY KEY(INVOICE\_ID, TREATMENT\_ID),

FOREIGN KEY(INVOICE\_ID) REFERENCES INVOICE,

FOREIGN KEY(TREATMENT\_ID) REFERENCES TREATMENT

ON DELETE CASCADE

);

***7. Build a set of queries that must include:***

**CLINIC:**

string query = @"SELECT \* FROM CLINIC";

string query = @"INSERT INTO CLINIC VALUES(

'" + clinic.ClinicName + @"'

,'" + clinic.ClinicState + @"'

,'" + clinic.ClinicCity + @"'

,'" + clinic.ClinicStreet + @"'

,'" + clinic.ClinicAreaCode + @"'

,'" + clinic.ClinicPhoneNumber + @"')";

string query = @"UPDATE CLINIC SET

CLINIC\_NAME='"+clinic.ClinicName+@"',

CLINIC\_STATE='"+clinic.ClinicState+ @"',

CLINIC\_CITY='"+clinic.ClinicCity+ @"',

CLINIC\_STREET='" + clinic.ClinicStreet + @"',

CLINIC\_AREA\_CODE='" + clinic.ClinicAreaCode + @"',

CLINIC\_PHONE\_NUMBER='" + clinic.ClinicPhoneNumber + @"'

WHERE CLINIC\_ID="+clinic.ClinicId+@"";

string query = @"DELETE FROM CLINIC WHERE CLINIC\_ID="+id+@"";

**EMPLOYEE:**

string query = @"SELECT \* FROM EMPLOYEE FULL OUTER JOIN LOGIN\_EMPLOYEE ON EMPLOYEE.EMPLOYEE\_ID = LOGIN\_EMPLOYEE.EMPLOYEE\_ID";

string query = @"INSERT INTO [dbo].[EMPLOYEE]

([EMPLOYEE\_F\_NAME]

,[EMPLOYEE\_L\_NAME]

,[EMPLOYEE\_DOB]

,[EMPLOYEE\_STATE]

,[EMPOYEE\_CITY]

,[EMPLOYEE\_STREET]

,[EMPLOYEE\_AREA\_CODE]

,[EMPLOYEE\_PHONE\_NUMBER]

,[CLINIC\_ID]) VALUES(

'" + employee.EmployeeFName + @"'

,'" + employee.EmployeeLName + @"'

,'" + employee.EmployeeDOB + @"'

,'" + employee.EmployeeState + @"'

,'" + employee.EmployeeCity + @"'

,'" + employee.EmployeeStreet + @"'

,'" + employee.EmployeeAreaCode+@"'

,'" + employee.EmployeePhoneNumber+@"'

,'" + employee.ClinicId + @"')";

string query = @"UPDATE EMPLOYEE SET

EMPLOYEE\_F\_NAME='" + employee.EmployeeFName + @"',

EMPLOYEE\_L\_NAME='" + employee.EmployeeLName + @"',

EMPLOYEE\_DOB='" + employee.EmployeeDOB + @"',

EMPLOYEE\_STATE='" + employee.EmployeeState+ @"',

EMPOYEE\_CITY='" + employee.EmployeeCity + @"',

EMPLOYEE\_STREET='" + employee.EmployeeStreet + @"',

EMPLOYEE\_AREA\_CODE='" + employee.EmployeeAreaCode + @"',

EMPLOYEE\_PHONE\_NUMBER='" + employee.EmployeePhoneNumber + @"',

CLINIC\_ID='" + employee.ClinicId + @"'

WHERE EMPLOYEE\_ID=" + employee.EmployeeId + @"";

string query = @"DELETE FROM EMPLOYEE WHERE EMPLOYEE\_ID=" + id + @"";

**EMPLOYEE\_LOGIN:**

string query = @"INSERT INTO LOGIN\_EMPLOYEE VALUES(

'" + login.username + @"'

,'" + login.EmployeeId + @"'

,'" + login.EmployeePassword + @"'

,'" + login.EmployeeRole + @"')";

string query = @"UPDATE LOGIN\_EMPLOYEE SET

EMPLOYEE\_USERNAME='" + login.username + @"',

EMPLOYEE\_PASSWORD='" + login.EmployeePassword + @"',

EMPLOYEE\_ROLE='" + login.EmployeeRole + @"'

WHERE EMPLOYEE\_ID=" + login.EmployeeId + @"";

**PRODUCT:**

string query = @"SELECT \* FROM PRODUCT";

string query = @"INSERT INTO PRODUCT VALUES(

'" + product.ProductName+ @"'

,'" + product.ProductQOH + @"'

,'" + product.ProductMin + @"'

,'" + product.ClinicId + @"'

,'" + product.ProductReorder + @"')";

string query = @"UPDATE PRODUCT SET

PRODUCT\_NAME='" + product.ProductName + @"',

PRODUCT\_QOH='" + product.ProductQOH + @"',

PRODUCT\_MIN='" + product.ProductMin + @"',

CLINIC\_ID='" + product.ClinicId + @"',

PRODUCT\_REORDER='" + product.ProductReorder + @"'

WHERE PRODUCT\_ID='" + product.ProductId + @"";

string query = @"DELETE FROM PRODUCT WHERE PRODUCT\_ID=" + id + @"";

**DENTIST:**

string query = @"SELECT EMPLOYEE\_F\_NAME, EMPLOYEE\_L\_NAME, EMPLOYEE\_DOB,EMPLOYEE\_STATE, EMPOYEE\_CITY,EMPLOYEE\_STREET,EMPLOYEE\_AREA\_CODE,EMPLOYEE\_PHONE\_NUMBER,

DENTIST\_ROLE, CLINIC\_NAME FROM DENTIST JOIN EMPLOYEE ON DENTIST.EMPLOYEE\_ID=EMPLOYEE.EMPLOYEE\_ID

JOIN CLINIC ON EMPLOYEE.CLINIC\_ID=CLINIC.CLINIC\_ID";

string query = @"INSERT INTO DENTIST VALUES(

'" + dentist.DentistId + @"'

,'" + dentist.EmployeeId + @"'

,'" + dentist.DentistRole + @"'')";

string query = @"UPDATE DENTIST SET

DENTIST\_ID='" + dentist.DentistId + @"',

DENTIST\_ROLE='" + dentist.DentistRole + @"',

WHERE EMPLOYEE\_ID=" + dentist.EmployeeId + @"";

string query = @"DELETE FROM DENTIST WHERE EMPLOYEE\_ID=" + id + @"";

**APPOINTMENT:**

string query = @"SELECT [PATIENT\_FIRST\_NAME] ,[PATIENT\_LAST\_NAME] ,[PATIENT\_STATE] ,[PATIENT\_CITY] ,[PATIENT\_STREET] ,[PATIENT\_AREA\_CODE]

[PATIENT\_PHONE\_NUMBER],[PATIENT\_EMAIL],[APPOINTMENT\_TIME], EMPLOYEE\_F\_NAME, EMPLOYEE\_L\_NAME FROM EMPLOYEE JOIN DENTIST ON EMPLOYEE.EMPLOYEE\_ID=DENTIST.EMPLOYEE\_ID

JOIN APPOINTMENT ON DENTIST.DENTIST\_ID=APPOINTMENT.DENTIST\_ID JOIN PATIENT ON APPOINTMENT.PATIENT\_PERSONAL\_NUMBER=PATIENT.PATIENT\_PERSONAL\_NUMBER";

string query = @"INSERT INTO APPOINTMENT VALUES(

'" + app.AppointmentTime + @"'

,'" + app.DentistId + @"'

,'" + app.PatientPersonalNumber + @"')";

string query = @"DELETE FROM APPOINTMENT WHERE APPOINTMENT\_ID=" + id + @"";

**TREATMENT:**

string query = @"SELECT TREATMENT\_DESCRIPTION, TREATMENT\_COST FROM TREATMENT";

string query = @"INSERT INTO TREATMENT VALUES(

'" + tt.TreatmentDescription + @"'

,'" + tt.TreatmentCost+ @"')";

string query = @"DELETE FROM TREATMENT WHERE TREATMENT\_ID=" + id + @"";

**INVOICE:**

string query = @"SELECT INVOICE\_ID,INVOICE DATE, SUM(TREATMENT\_COST) FROM TREATMENT JOIN INVOICE\_TREATMENT ON TREATMENT.TREATMENT\_ID=INVOICE\_TREATMENT.TREATMENT\_ID

JOIN INVOICE ON INVOICE\_TREATMENT.INVOICE\_ID=INOVICE.INOVICE\_ID";

string query = @"INSERT INTO INVOICE VALUES(

'" + inv.InvoiceDate+ @"')";

string query = @"DELETE FROM INVOICE WHERE INVOICE\_ID=" + id + @"";

CREATE TRIGGER TRG\_PROD\_REORDER

ON PRODUCT

AFTER UPDATE

AS

UPDATE PRODUCT

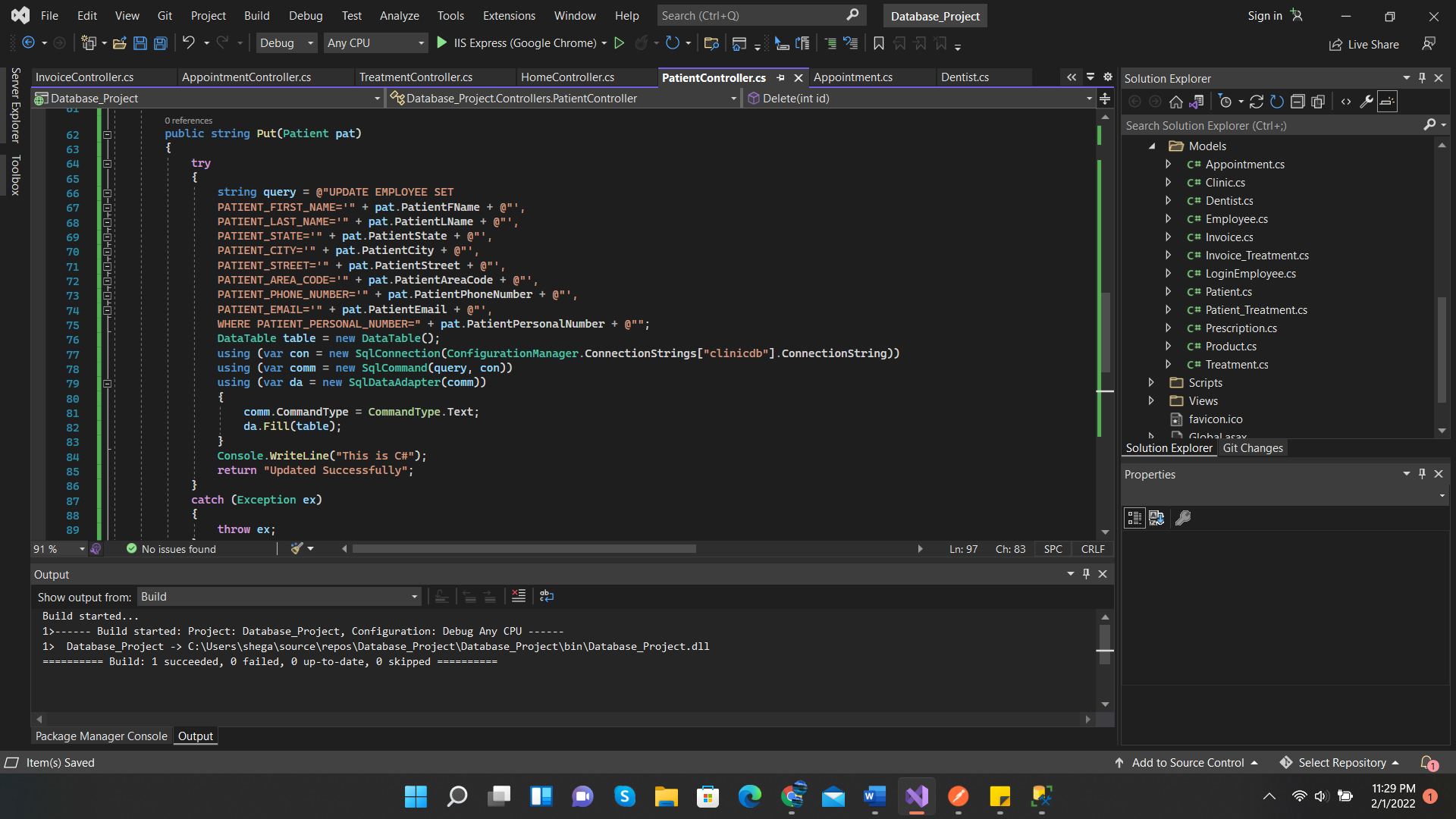
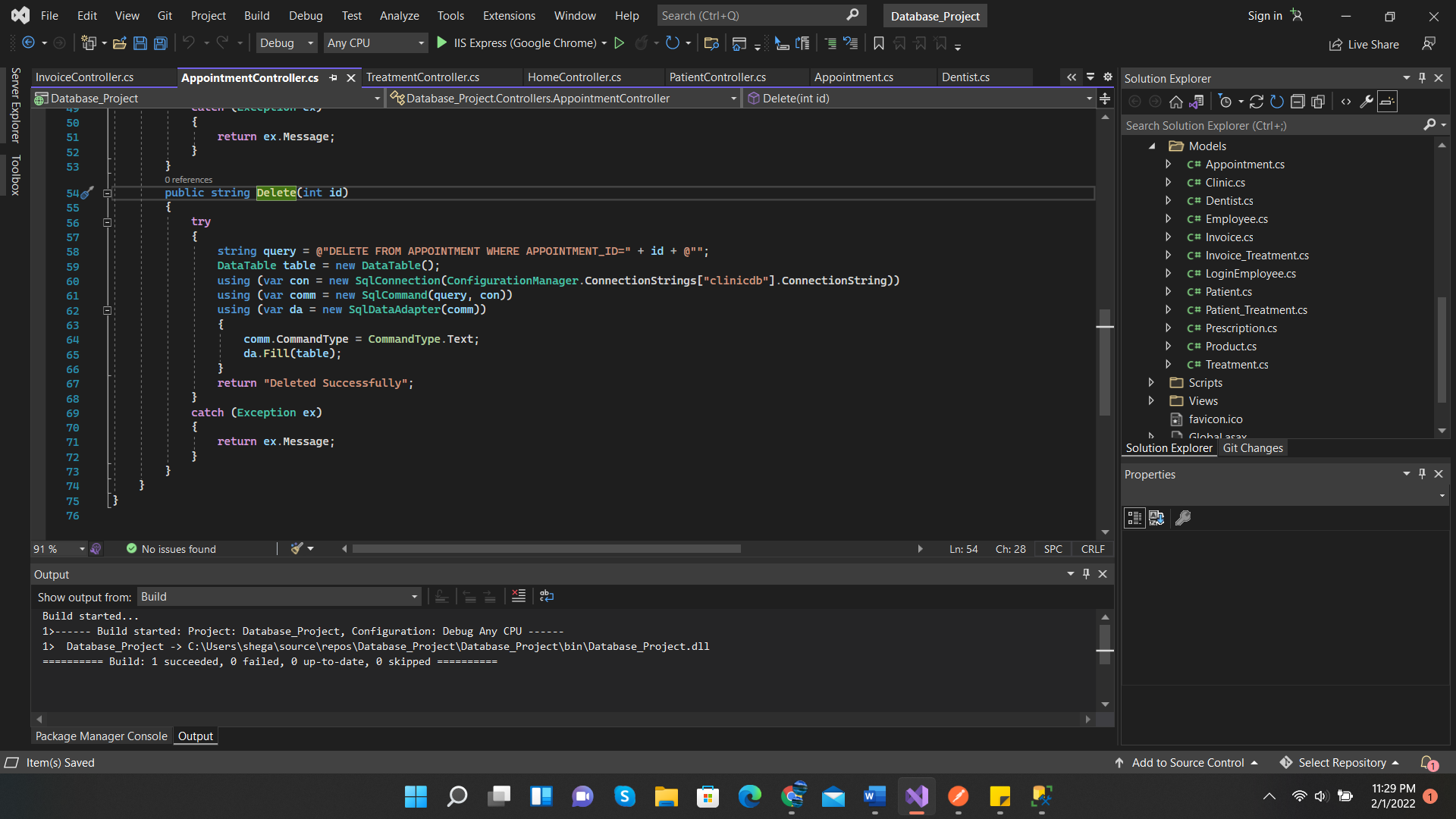
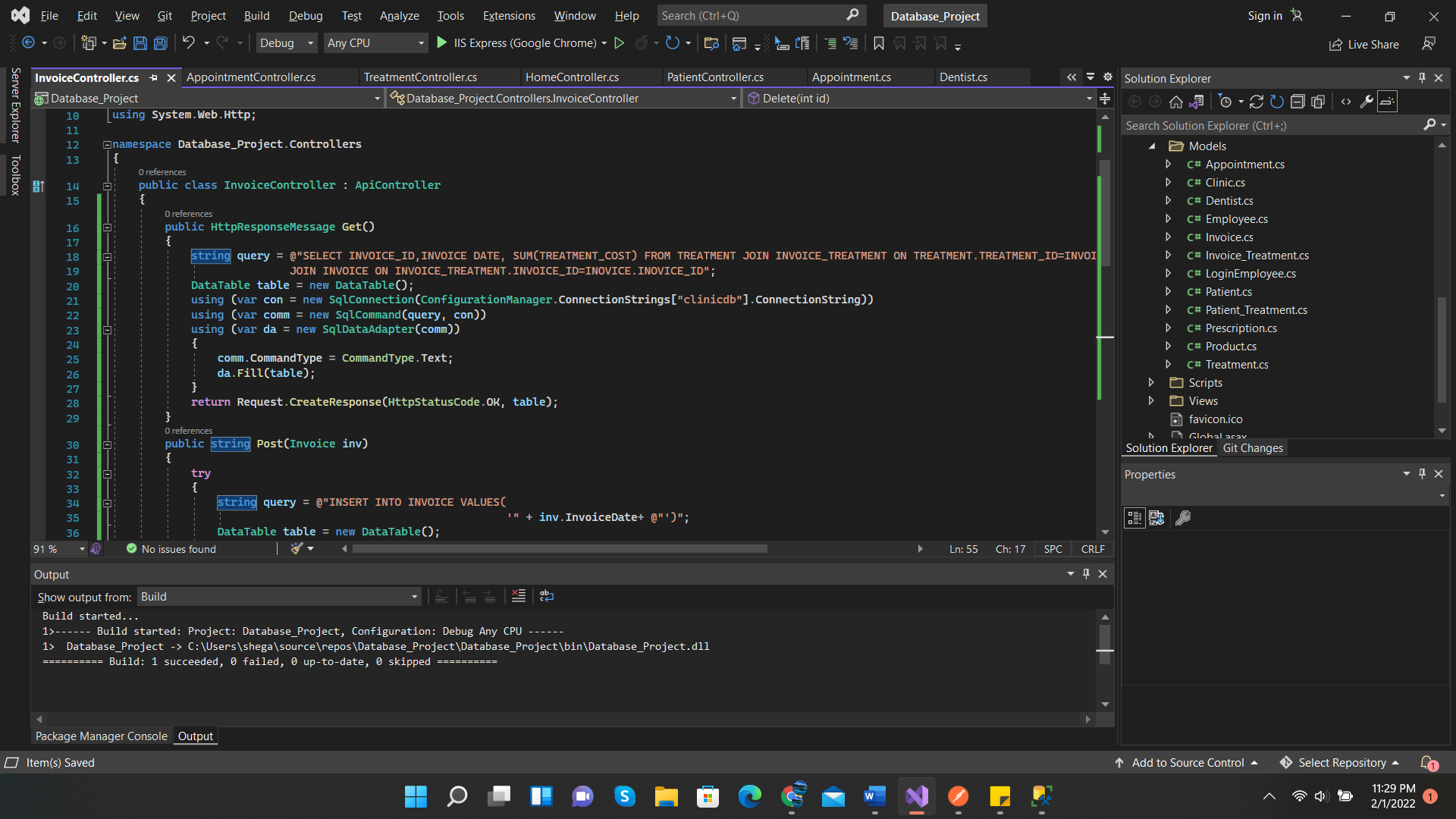
SET PRODUCT\_REORDER = 1

WHERE PRODUCT\_QOH <= PRODUCT\_MIN

GO

***8. Create a user interface to build a web-based or desktop application to interact with***

***your database. The programming language is your free choice.***

******