

**Düzce Üniversitesi**

**Mühendislik Fakültesi-Bilgisayar Mühendisliği**

**Veritabanı Yönetim Sistemleri Dersi Projesi-NÖ**

**211001057- Çiğdem Kurt**

**211001031- Gamze Özdemir**

**221001431- Ayşe Betül Boydaş**

**211001006-Rana Yılmaz**

İçindekiler Tablosu

[1-GİRİŞ 2](#_Toc184819005)

[2-SQL SERVER 3](#_Toc184819006)

[3-VS FORM 3](#_Toc184819007)

[3.1-Giriş Ekranı: 4](#_Toc184819008)

[3.2.1 Doctor Page (Doktor Sayfası): 5](#_Toc184819009)

[3.2.2 -Patient Page(Hasta sayfası): 6](#_Toc184819010)

[3.2.3 -Diagnosis Page(Tanı sayfası): 7](#_Toc184819011)

[4-TRİGGER 9](#_Toc184819012)

[5-STORED PROCEDURE OLUŞTURMA 10](#_Toc184819013)

[6-RAPORLARI KAYDETME 11](#_Toc184819014)

[7-İMPORT/EXPORT 11](#_Toc184819015)

[8-YEDEKLEME İŞLEMİ 12](#_Toc184819016)

[9-EKLER 13](#_Toc184819017)

[10-KAYNAKÇA 25](#_Toc184819018)

# 

# 

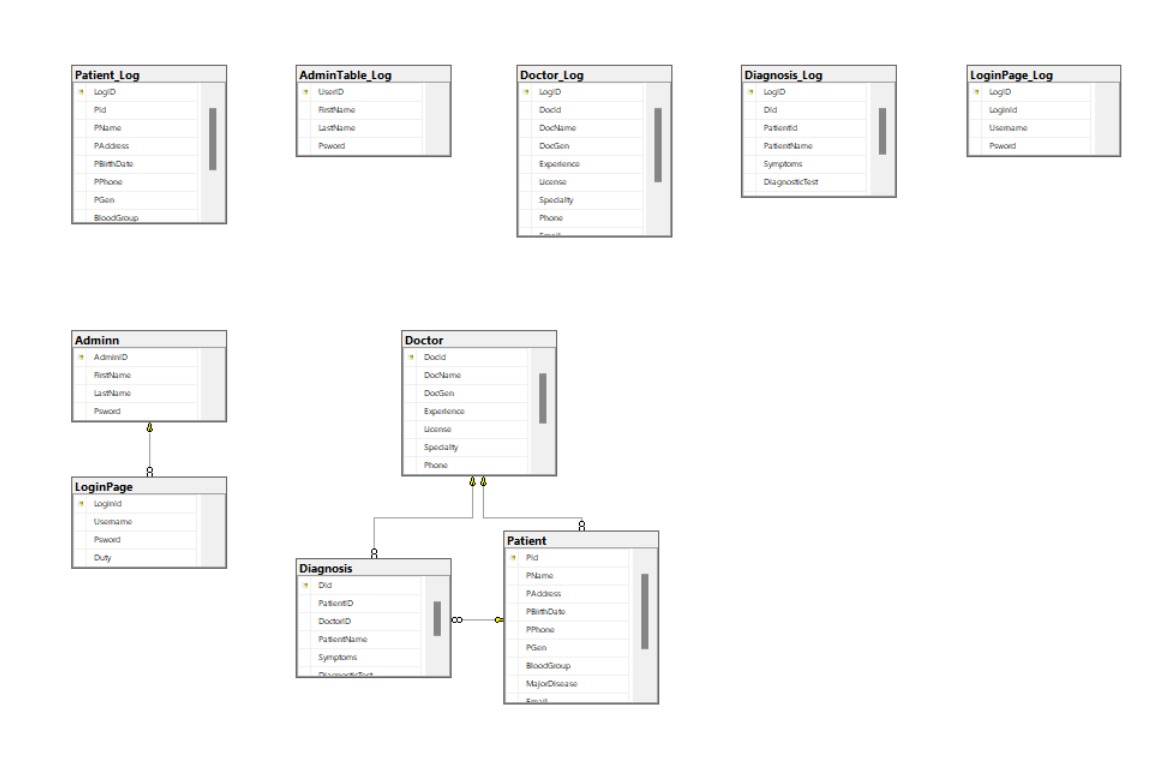
# 1-GİRİŞ

Bu proje, hastanelerdeki hasta kayıtları, tedavi süreçleri ve raporlama işlemlerini yönetmek amacıyla geliştirilmiştir. Visual Studio kullanılarak C# form uygulaması şeklinde tasarlanmış ve SQL Server veritabanından veri çekilerek oluşturulmuştur. Projenin detaylarına raporun ilerleyen bölümlerinde yer verilmiştir. Projede yer alan tablolar ve form üzerindeki görseller, daha kapsamlı ve evrensel bir yapı sunması amacıyla İngilizce olarak hazırlanmıştır.

# 2-SQL SERVER

Projede kullanılan veritabanı, SQL Server olarak seçilmiştir. Bu aşamada tablolar, tetikleyiciler (triggers) ve saklı yordamlar (stored procedures) gibi veritabanı unsurları oluşturulmuş olup, tüm veri çekme ve kaydetme işlemleri bu veritabanı üzerinden gerçekleştirilmiştir. Ayrıca, proje kapsamında oluşturulan ER diyagramları da bu bölümde tasarlanmıştır.

Örnek olarak, oluşturulan bazı tablo isimleri şunlardır: **Admin**, **Doctor**, **Patient**, **Diagnosis** ve silinen kayıtların kaydedileceği **Log** tabloları. Veritabanı tablolarını oluşturma kodları EK1 bölümünde gösterilmiştir.



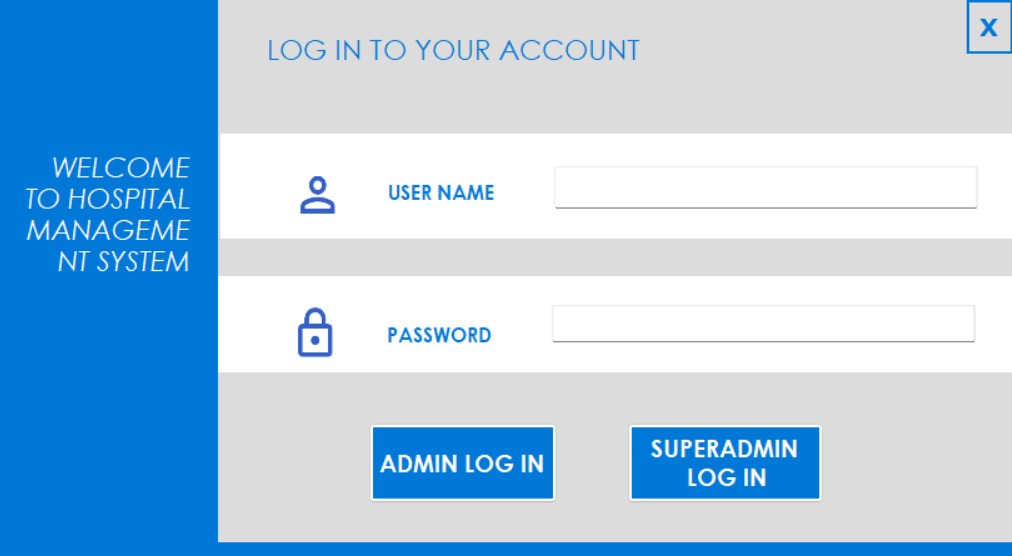
# 3-VS FORM

Bu bölümde, Visual Studio kullanılarak form kısmında gerçekleştirilen proje detayları maddeler halinde sıralanmıştır:

## 3.1-Giriş Ekranı:

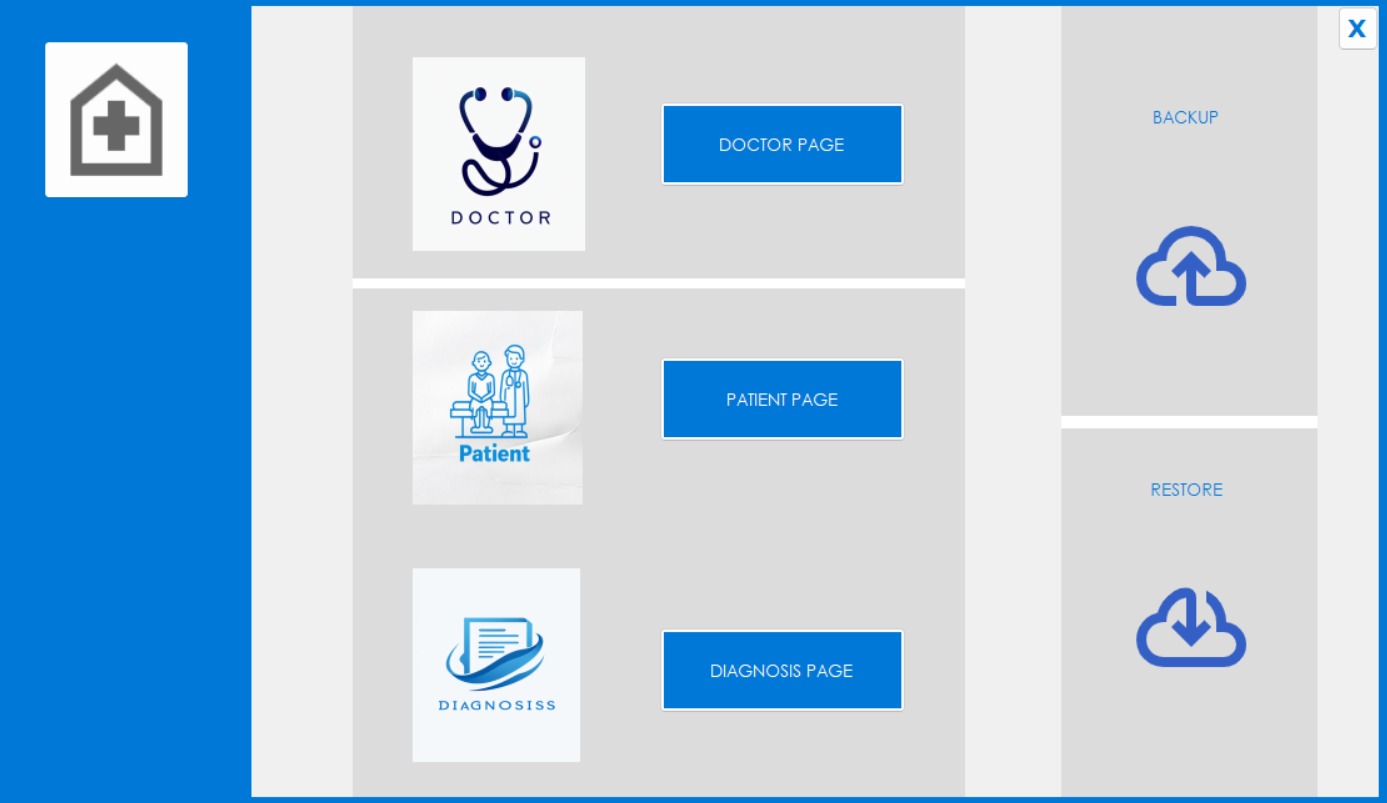
**Hastane Yönetim Sistemine Hoşgeldiniz!**

* Bu ekran, uygulama çalıştırıldığında kullanıcıyı karşılayan ilk ekrandır.
* Kullanıcı adı ve şifre girişini sağlayan **label** ve **textbox** alanları bulunmaktadır.
* Kullanıcının kimliğine göre seçim yapabileceği iki farklı giriş seçeneği sunulmuştur: **Admin Login**, ve **Superadmin Login**.
* Kullanıcı, kendi rolüne uygun butona tıklayarak sisteme giriş yapar.



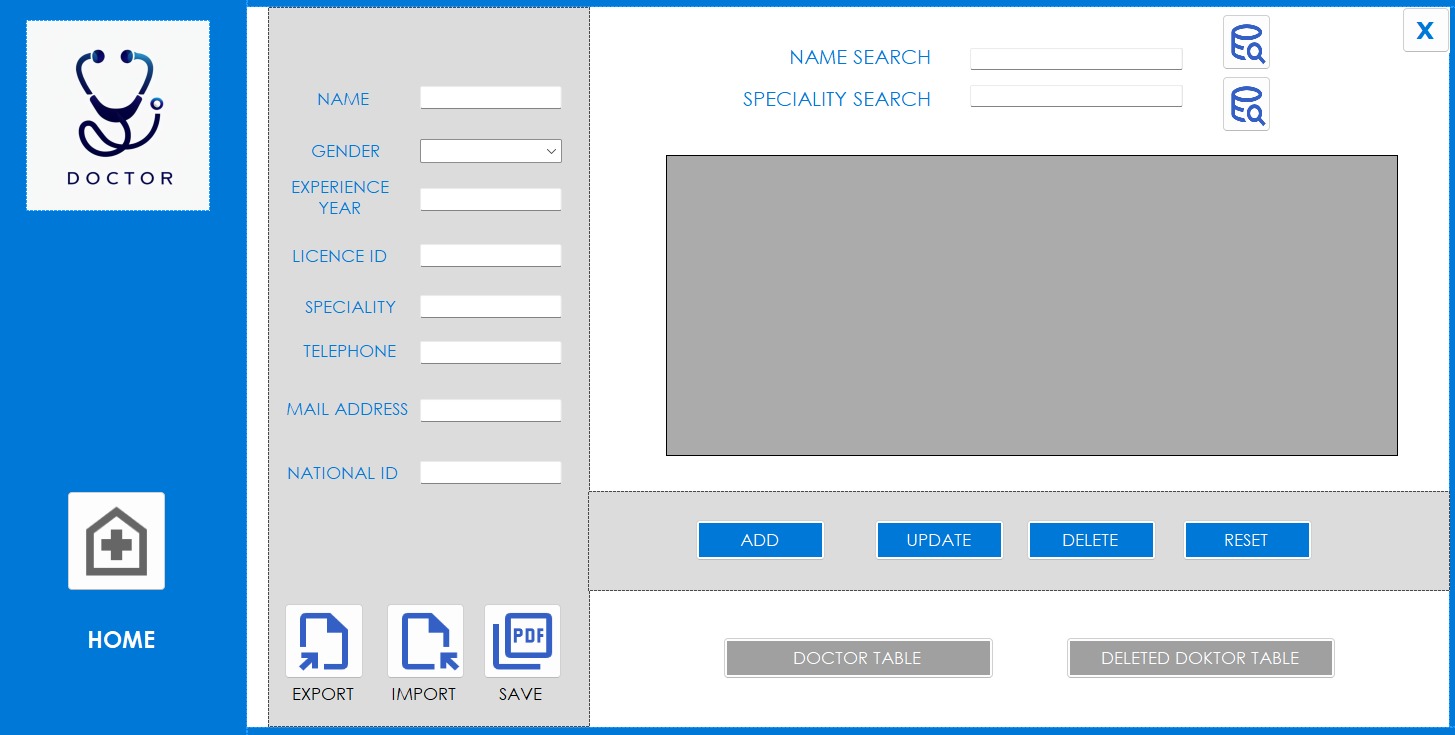
3.2- HOME (Admin Login):

* İlk ekranda bulunan **Admin Login** seçeneği ile giriş yapılması durumunda, kullanıcıyı "ana ekran" dediğimiz bu sayfa karşılar.
* Bu ekranda, **Doctor Page (Doktor Sayfası)**, **Patient Page (Hasta Sayfası)** ve **Diagnosis Page (Teşhis Sayfası)** olmak üzere üç farklı buton yer alır.
* Ayrıca, diğer ekranlardan farklı olarak, burada **Backup (Yedekleme)** ve **Restore (Geri Yükleme)** seçenekleri bulunmaktadır.



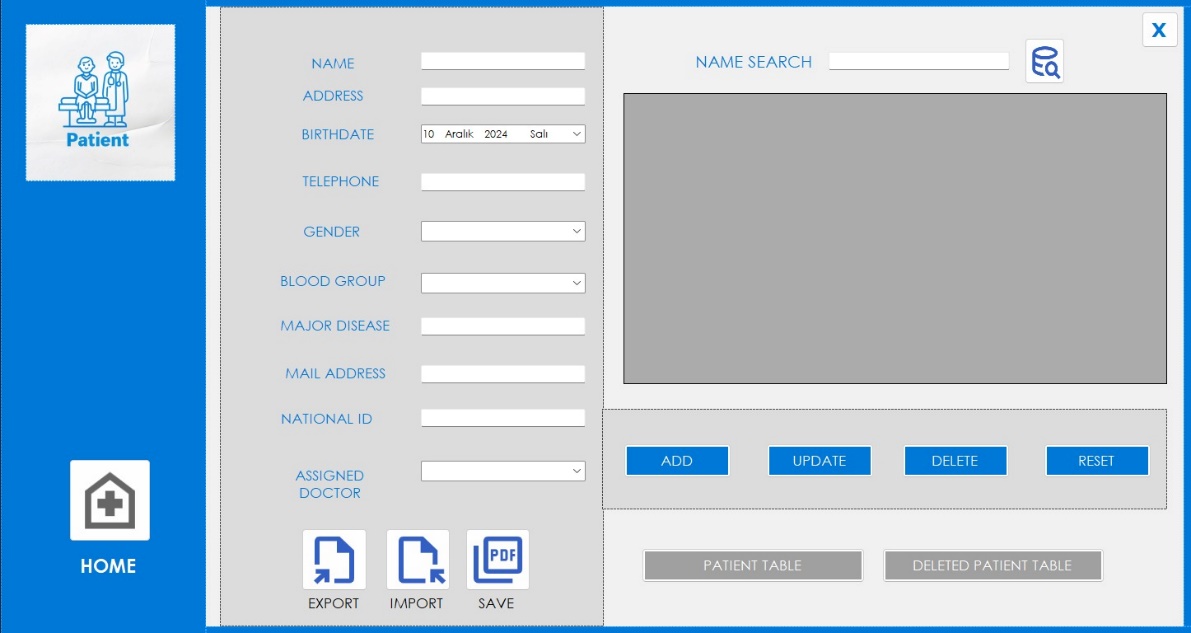
### 3.2.1 Doctor Page (Doktor Sayfası):

* Bu butona tıklandığında, doktor işlemlerinin gerçekleştirildiği ekran açılır.
* Doktor kaydı ekleme, güncelleme, silme ve sıfırlama (resetleme) gibi işlemler bu sayfada yapılır.
* Doktor bilgileri için ilgili **textbox** alanlarına şu bilgiler girilir:
  + **İsim**
  + **Cinsiyet**
  + **Deneyim Yılı**
  + **Lisans ID’si**
  + **Uzmanlık Alanı**
  + **Telefon Numarası**
  + **E-posta Adresi**
  + **Ulusal ID (örn. TC Kimlik Numarası)**
* Girilen bilgilerin ardından, ekleme, güncelleme, silme ve sıfırlama işlemleri kolaylıkla gerçekleştirilebilir.
* Ekranın sağ alt köşesinde bulunan **Doctor Table (Doktor Tablosu)** butonuna tıklanarak mevcut tüm doktor kayıtları görüntülenebilir. Aynı şekilde, **Deleted Doctor Table (Silinmiş Doktor Tablosu)** butonuna tıklanarak silinen doktor kayıtları incelenebilir.
* **Import/Export (Veri Al/Ver)** butonları kullanılarak, Excel,CSV formatındaki dosyalardan veri aktarımı yapılabilir veya mevcut veriler dışa aktarılabilir.
* **Save (Kaydet)** butonuyla, ekranda listelenmekte olan mevcut rapor, PDF formatında bir dosya olarak kaydedilebilir.
* Bu bölümde, iş süreçlerini hızlandırmak adına **stored procedure** kullanılarak geliştirilmiş iki arama butonu bulunmaktadır:
  + **Name Search (İsim Arama):** Doktorun adına göre arama yapılmasını sağlar.
  + **Specialty Search (Uzmanlık Arama):** Doktorun uzmanlık alanına göre arama yapılmasını sağlar.
* Son olarak, **Home (Ana Sayfa)** butonu kullanılarak ana ekrana kolayca dönüş yapılabilir.



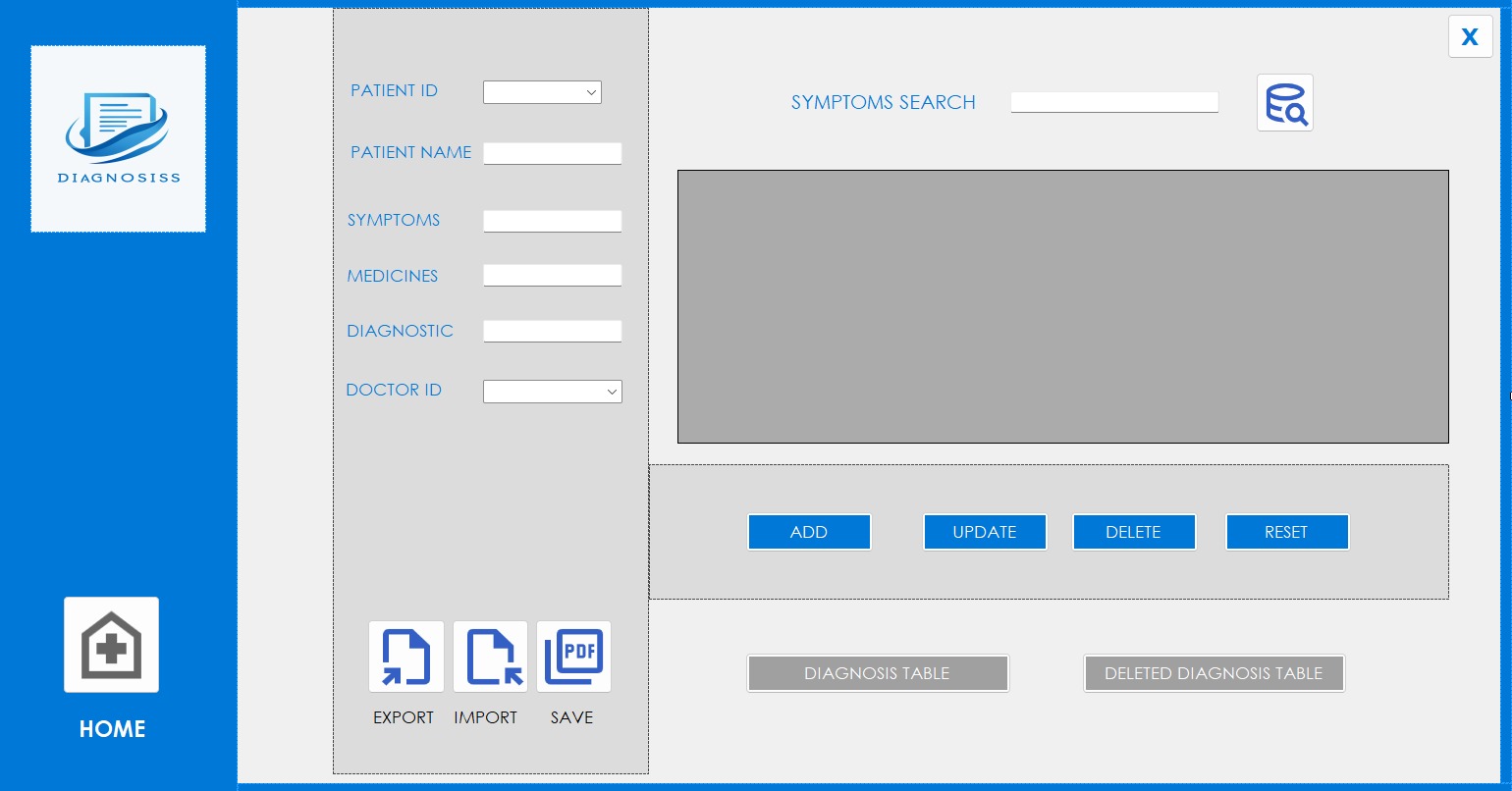
### 3.2.2 -Patient Page(Hasta sayfası):

* Bu butona tıklandığında, hasta işlemlerinin gerçekleştirildiği ekran açılır.
* hasta kaydı ekleme, güncelleme, silme ve sıfırlama (resetleme) gibi işlemler bu sayfada yapılır.
* Hasta bilgileri için ilgili **textbox** alanlarına şu bilgiler girilir:
  + Ad (Name)
  + Adres (Address)
  + Telefon (Phone)
  + Doğum Tarihi (Date of Birth)
  + Kan Grubu (Blood Type)
  + Cinsiyet (Gender)
  + Önemli Hastalık (Major Disease)
  + E-posta Adresi (Email Address)
  + Ulusal ID (National ID)
  + Atanan Doktor (Assigned Doctor)
* Girilen bilgilerin ardından, ekleme, güncelleme, silme ve sıfırlama işlemleri kolaylıkla gerçekleştirilebilir.
* Ekranın sağ alt köşesinde bulunan **Patient** **Table (Hasta Tablosu)** butonuna tıklanarak mevcut tüm hasta kayıtları görüntülenebilir. Aynı şekilde, **Deleted Patient Table (Silinmiş Hasta Tablosu)** butonuna tıklanarak silinen hasta kayıtları incelenebilir.
* **Import/Export (Veri Al/Ver)** butonları kullanılarak, Excel,CSV formatındaki dosyalardan veri aktarımı yapılabilir veya mevcut veriler dışa aktarılabilir.
* **Save (Kaydet)** butonuyla, ekranda listelenmekte olan mevcut rapor, PDF formatında bir dosya olarak kaydedilebilir.
* Bu bölümde, iş süreçlerini hızlandırmak adına **stored procedure** kullanılarak geliştirilmiş bir arama butonu bulunmaktadır:
  + **Name Search (İsim Arama):** Hastanın adına göre arama yapılmasını sağlar.
* Son olarak, **Home (Ana Sayfa)** butonu kullanılarak ana ekrana kolayca dönüş yapılabilir.



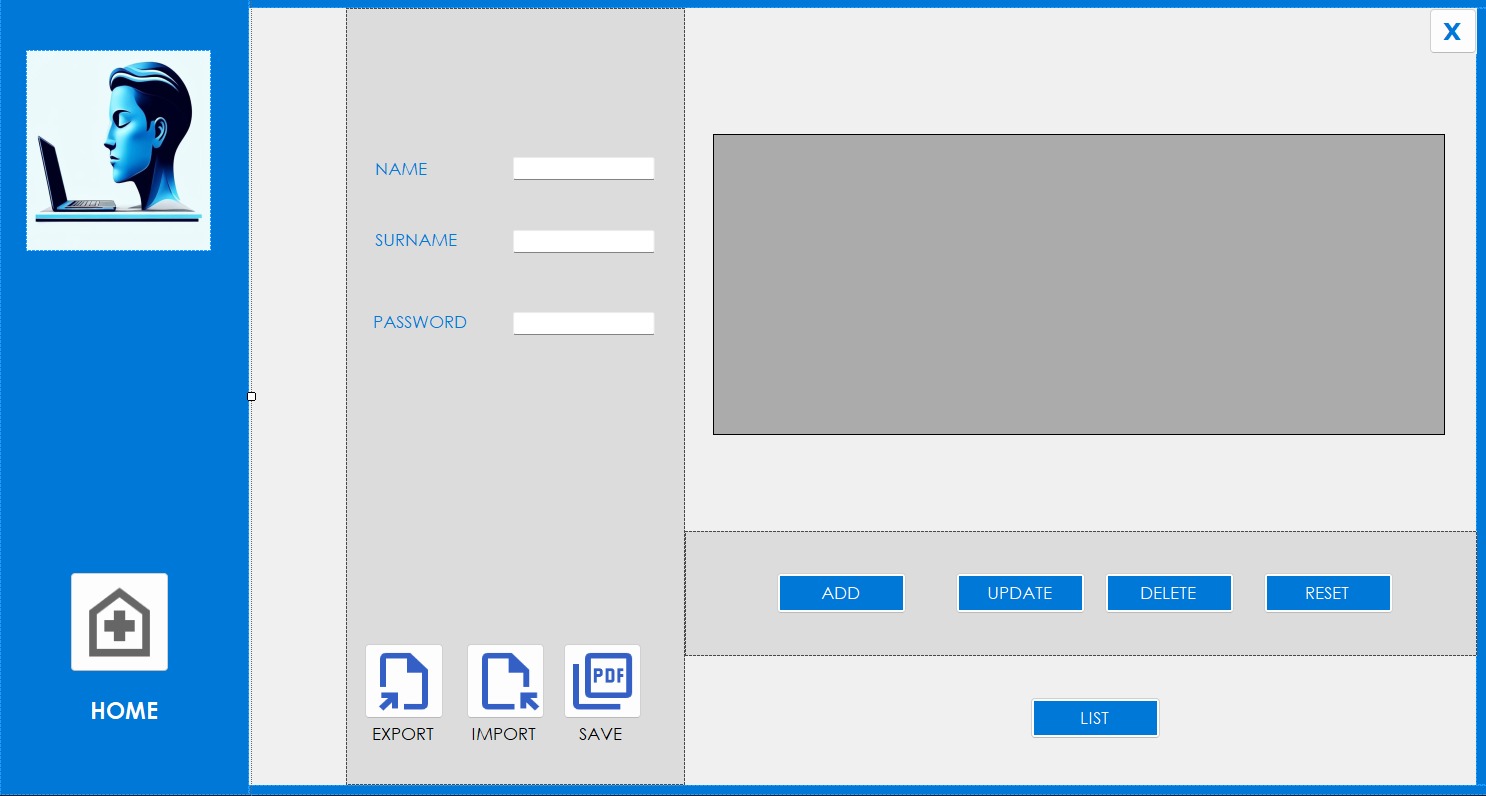
### 3.2.3 -Diagnosis Page(Tanı sayfası):

* Bu butona tıklandığında, Tanı işlemlerinin gerçekleştirildiği ekran açılır.
* Tanı kaydı ekleme, güncelleme, silme ve sıfırlama (resetleme) gibi işlemler bu sayfada yapılır.
* Tanı bilgileri için ilgili **textbox** alanlarına şu bilgiler girilir:
  + Patient ID (Hasta ID’si)
  + Patient Name (Hasta Adı)
  + Symptoms (Belirtiler)
  + Medicines (İlaçlar)
  + Diagnostic (Tanı)
  + Doctor ID (Doktor ID’si)
* Girilen bilgilerin ardından, ekleme, güncelleme, silme ve sıfırlama işlemleri kolaylıkla gerçekleştirilebilir.
* Ekranın sağ alt köşesinde bulunan **Dıagnosis Table (Tanı Tablosu)** butonuna tıklanarak mevcut tüm tanı kayıtları görüntülenebilir. Aynı şekilde, **Deleted Diagnosis Table (Silinmiş Tanı Tablosu)** butonuna tıklanarak silinen tanı kayıtları incelenebilir.
* **Import/Export (Veri Al/Ver)** butonları kullanılarak, Excel,CSV formatındaki dosyalardan veri aktarımı yapılabilir veya mevcut veriler dışa aktarılabilir.
* **Save (Kaydet)** butonuyla, ekranda listelenmekte olan mevcut rapor, PDF formatında bir dosya olarak kaydedilebilir.
* Bu bölümde, iş süreçlerini hızlandırmak adına **stored procedure** kullanılarak geliştirilmiş bir arama butonu bulunmaktadır:
  + **Symptoms Search(Belirti Arama):** Hastanın belirtilerine göre arama yapılmasını sağlar.
* Son olarak, **Home (Ana Sayfa)** butonu kullanılarak ana ekrana kolayca dönüş yapılabilir.



3.3 -SuperAdmin Login:  
Bu ekran, yeni kullanıcı oluşturma ve bu kullanıcılara yetki atama işlemlerinin yapıldığı sayfadır.

* Kullanıcı bilgileri, **Name**, **Surname** ve **Password** gibi bilgilerle ilgili **textbox** alanları kullanılarak tanımlanır.
* **SuperAdmin**, bu ekranda admin kullanıcıları üzerinde ekleme, silme, güncelleme, resetleme,listeleme işlemleri gerçekleştirebilir.
* Ayrıca, **Import/Export (Veri Al/Ver)** butonları ile Excel,CSV formatındaki dosyalardan veri aktarımı yapılabilir ya da mevcut veriler dışa aktarılabilir.
* **Save (Kaydet)** butonuyla, ekranda listelenen mevcut rapor, PDF formatında bir dosya olarak kaydedilebilir.



# 4-TRİGGER

Projede kullanılan **trigger**'lar, veritabanında yapılacak silme işlemleri sırasında önemli verilerin kaybolmasını engellemek için **log** kayıtlarını oluşturur. Bu sayede, sistemde herhangi bir verinin yanlışlıkla silinmesi durumunda, geçmiş verilere ulaşılabilir.

Projede kullanılan beş farklı trigger ve amaçları aşağıda detaylı olarak açıklanmıştır:

#### **1. trg\_Adminn\_Insert Trigger**

Bu trigger, Adminn tablosuna yeni bir kayıt eklendiğinde çalışır. Yeni eklenen admin kaydının bilgilerini, LoginPage tablosuna ekler. Burada, adminin adı ve soyadı birleşerek kullanıcı adı (username) olarak atanır, şifre ise olduğu gibi aktarılır.

* Yeni eklenen Adminn kaydının adı ve soyadı birleştirilerek kullanıcı adı oluşturulur.
* Kullanıcı adı, şifre ve görev bilgileri LoginPage tablosuna aktarılır.

Trigger kodu EK2 bölümünde gösterilmiştir.

#### **2. trg\_AdminTable\_Delete Trigger**

Bu trigger, Adminn tablosunda bir kayıt silindiğinde çalışır. Silinen kaydın bilgileri AdminTable\_Log tablosuna aktarılır ve silme işlemi Adminn tablosunda gerçekleştirilir. Bu, admin kaydının silindiğini izlemek için kullanılır.

* Silinen admin kaydının bilgileri AdminTable\_Log tablosuna aktarılır.
* Ardından, silinen kayıt Adminn tablosundan kaldırılır.

Trigger kodu EK3 bölümünde gösterilmiştir.

#### **3. trg\_Doctor\_Delete Trigger**

Doctor tablosunda bir doktor kaydının silinmesi durumunda çalışır. Silinen doktor kaydının bilgileri Doctor\_Log tablosuna eklenir ve kaydın silinmesi sağlanır.

* Silinen doktor kaydının bilgileri Doctor\_Log tablosuna eklenir.
* Sonrasında, kaydın Doctor tablosundan silinmesi işlemi gerçekleştirilir.

Trigger kodu EK4 bölümünde gösterilmiştir.

#### **4. trg\_Diagnosis\_AfterDelete Trigger**

Diagnosis tablosunda bir teşhis kaydı silindiğinde çalışır. Silinen kaydın bilgileri Diagnosis\_Log tablosuna aktarılır, ardından asıl kayıt Diagnosis tablosundan silinir.

* Silinen teşhis kaydının bilgileri Diagnosis\_Log tablosuna eklenir.
* Sonrasında, ilgili teşhis kaydı Diagnosis tablosundan silinir

Trigger kodu EK5 bölümünde gösterilmiştir.

#### **5. trg\_Patient\_Delete Trigger**

Patient tablosunda bir hasta kaydının silinmesi durumunda, silinen hastanın bilgileri Patient\_Log tablosuna aktarılır ve kaydın Patient tablosundan silinmesi işlemi gerçekleştirilir.

* Silinen hasta kaydının bilgileri Patient\_Log tablosuna aktarılır.
* Sonrasında, kaydın Patient tablosundan silinmesi işlemi yapılır

Trigger kodu EK6 bölümünde gösterilmiştir.

# 5-STORED PROCEDURE OLUŞTURMA

Bu bölümde, altı farklı **stored procedure** kullanılarak altı adet rapor seçeneği oluşturulmuştur.

* **Doktor Kısmı:**
  + **SearchDoctorByName**: Doktorların adlarına göre filtrelenmesini sağlar. Bu özelliği kullanmak için **Name Search** isimli arama motoru eklenmiştir.

SP kodu EK7 bölümünde gösterilmiştir.

* + **SearchDoctorsBySpecialty**: Doktorların uzmanlık alanlarına göre detaylı bir şekilde listelenmesini sağlar. . Bu özelliği kullanmak için **Specialty Search** isimli arama motoru eklenmiştir.

SP kodu EK8 bölümünde gösterilmiştir.

* + **GetDoctorLogs:** Bu Proje,veritabanındaki silme işlemlerini takip etmek ve kayıt altına almak için bir **Doctor\_Log** tablosu kullanmaktadır. **GetDoctorLogs** stored procedure’ü, bu tabloyu sorgulamak ve silinen doktor kayıtlarının detaylarını görüntülemek amacıyla geliştirilmiştir.

SP kodu EK9 bölümünde gösterilmiştir.

* **Hasta Kısmı:**
  + **SearchPatientByName**: Hasta isimlerine göre ilgili kayıtları getirir. **Name Search** isimli arama motoru eklenmiştir.

SP kodu EK10 bölümünde gösterilmiştir.

* + **GetPatientLogs:** Bu proje, hasta kayıtlarının silinme işlemlerini takip etmek ve bu işlemleri kayıt altına almak için **Patient\_Log** tablosunu kullanmaktadır. **GetPatientLogs** stored procedure’ü, bu tabloyu sorgulamak ve silinen hasta kayıtlarının detaylarını görüntülemek amacıyla geliştirilmiştir.

SP kodu EK11 bölümünde gösterilmiştir.

* **Tanı Kısmı:**
  + **SearchDiagnosisBySymptoms**: Kullanıcıların teşhis tablosunda semptomlara göre arama yapmasını sağlar. Bu özelliği kullanmak için **Symptoms Search** isimli arama motoru eklenmiştir.

SP kodu EK12bölümünde gösterilmiştir.

Bu store procedürler sayesinde, ilgili raporlar görüntülenebilir, incelenebilir ve PDF formatında kaydedilebilir.

# 6-RAPORLARI KAYDETME

Ekranda listelenmekte olan mevcut raporun PDF formatında kaydedilmesi, **Save** butonuna eklenmiş bir özellik sayesinde gerçekleştirilmektedir.

Kullanıcı, **Save** butonuna tıkladığında, bir **Save File Dialog** penceresi açılır. Buradan kaydedilecek PDF dosyasının adı ve konumu seçilir. Kod, ekranda bulunan **DataGridView**’deki verileri alır, başlıkları ve içerikleri bir tablo formatında düzenler ve bir **PDF** dosyasına yazar. Bu işlem için **iTextSharp** kütüphanesi kullanılmıştır.

# 7-İMPORT/EXPORT

**Import(veri alma)**

Bu projede, veri yönetim işlemlerini kolaylaştırmak amacıyla **CSV import** özelliği geliştirilmiştir. Bu özellik, kullanıcıların harici bir CSV dosyasındaki verileri programa aktarmasına olanak tanır.

* Kullanıcı, form ekranında bulunan **Import** butonuna tıkladığında bir dosya seçme penceresi (OpenFileDialog) açılır.
* Kullanıcı, bir **.csv** dosyası seçtiğinde, seçilen dosya **CsvHelper** kütüphanesi ile okunur.
* Okunan veriler bir tablo yapısına dönüştürülür ve form ekranında görüntülenen **DataGridView** bileşenine yüklenir.
* İşlem başarıyla tamamlandığında kullanıcıya bir başarı mesajı gösterilir.

**Export**

Export butonu Save butonuyla aynı şekilde işlev görür.(bknz bölüm6)

# 8-YEDEKLEME İŞLEMİ

**Backup komutu**

veritabanı yedekleme işlemini gerçekleştirmek için Backup özelliği geliştirilmiştir. Bu özellik, SQL Server’da tanımlı VThastane veritabanının .bak formatında bir yedeğini oluşturur.

Backup işlemi şu adımlarla gerçekleştirilir:

1. Kullanıcı, form üzerindeki Backup butonuna tıkladığında, sistem mevcut tarih ve saat bilgisine göre dosya adı oluşturur.
2. SQL Server'dan BACKUP DATABASE sorgusu çalıştırılarak belirtilen dizine bir yedek dosyası kaydedilir.
3. Yedekleme işlemi başarıyla tamamlandığında kullanıcıya bilgilendirme mesajı gösterilir.

Bu özellik, verilerin güvenliğini sağlamak ve olası veri kayıplarına karşı önlem almak amacıyla geliştirilmiştir. Hata yakalama mekanizması sayesinde işlem sırasında karşılaşılabilecek hatalar kullanıcıya bildirilmektedir.

**RESTORE Komutu**

veritabanı geri yükleme işlemini gerçekleştiren bir Restore özelliği geliştirilmiştir. Bu özellik, kullanıcıların daha önce alınan bir yedek dosyasını seçerek veritabanını eski haline döndürmesine olanak tanır.

Restore işlemi şu adımlarla gerçekleştirilir:

1. Kullanıcı, form üzerindeki Restore butonuna tıkladığında bir dosya seçme penceresi açılır.
2. Kullanıcı, daha önce alınmış bir .bak uzantılı yedek dosyasını seçer.
3. SQL Server’da tanımlanan VThastane veritabanı, seçilen dosyadan geri yüklenir. Geri yükleme işlemi sırasında veritabanı tek kullanıcı moduna alınır, işlemler tamamlandıktan sonra tekrar çok kullanıcı moduna geçirilir.
4. İşlem başarıyla tamamlandığında kullanıcıya bir bilgilendirme mesajı gösterilir.

Bu özellik, verilerin güvenliğini artırmak ve kayıp durumlarında sistemi hızlı bir şekilde kurtarmak için projeye dahil edilmiştir. Hata yakalama mekanizması, işlem sırasında oluşabilecek sorunları kullanıcıya bildirmek için kullanılmıştır.

# 9-EKLER

**EK1**

Bu bölümde oluşturulan normal tablolar,log tabloları ve veritabnından ınsert ınto metodu ile girilen veriler gösterilmektedir.

CREATE TABLE Adminn (

AdminID INT IDENTITY(1,1) PRIMARY KEY,

[FirstName] VARCHAR(50) NOT NULL, -- Kullanıcı Adı

[LastName] VARCHAR(50) NOT NULL, -- Kullanıcı Soyadı

[Psword] NVARCHAR(6) NOT NULL,

SuperAdmin BIT NOT NULL DEFAULT 0 -- 1: SuperAdmin, 0: Normal Admin

);

CREATE TABLE [dbo].[LoginPage] (

[LoginId] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

[Username] VARCHAR(25) NOT NULL,

[Psword] CHAR(6) NOT NULL,

[Duty] VARCHAR(50) CHECK (Duty IN ('Adminn')) NOT NULL,

aID INT ,

FOREIGN KEY (aID) REFERENCES Adminn(AdminID) ON DELETE CASCADE

);

INSERT INTO [dbo].[Adminn] (FirstName, LastName, Psword, SuperAdmin)

VALUES

('Ali', 'Kaya', '123456', 1), -- Ali Kaya, SuperAdmin

('Ayşe', 'Demir', 'abcdef', 0), -- Ayşe Demir, Normal Admin

('Can', 'Korkmaz', '654321', 0), -- Can Korkmaz, Normal Admin

('Hülya', 'Uzun', 'admin1', 0), -- Hülya Uzun, Normal Admin

('Murat', 'Polat', 'secure', 0); -- Murat Polat, Normal Admin

INSERT INTO [dbo].[LoginPage] (Username, Psword, Duty, aID)

VALUES

('AliKaya', '123456', 'Adminn', 1), -- Ali Kaya, SuperAdmin

('AyseDemir', 'abcdef', 'Adminn', 2), -- Ayşe Demir, Normal Admin

('CanKorkmaz', '654321', 'Adminn', 3), -- Can Korkmaz, Normal Admin

('HulyaUzun', 'admin1', 'Adminn', 4), -- Hülya Uzun, Normal Admin

('MuratPolat', 'secure', 'Adminn', 5); -- Murat Polat, Normal Admin

create table [dbo].[Doctor](

[DocId] INT NOT NULL IDENTITY(1000,1) PRIMARY KEY,

[DocName] VARCHAR(50) NOT NULL,

[DocGen] VARCHAR(6) NOT NULL,

[Experience] VARCHAR(2) NOT NULL,

[License] VARCHAR(5) NOT NULL,

Specialty NVARCHAR(50) NOT NULL, -- Branş

Phone CHAR(11), -- Telefon

Email NVARCHAR(50) CONSTRAINT CK\_Email\_Gmail CHECK (Email LIKE '%@gmail.com'),

NationalID CHAR(11) NOT NULL UNIQUE -- TCKimlikNo

);

CREATE TABLE [dbo].[Patient] (

[PId] INT NOT NULL IDENTITY(1,1) PRIMARY KEY, -- Birincil anahtar

[PName] VARCHAR(50) NOT NULL, -- Hasta adı

[PAddress] VARCHAR(50) NOT NULL, -- Adres

[PBirthDate] date NOT NULL, -- dg

[PPhone] CHAR(11) NOT NULL, -- Telefon

[PGen] VARCHAR(6) NOT NULL, -- Cinsiyet

[BloodGroup] VARCHAR(3) NOT NULL, -- Kan grubu

[MajorDisease] VARCHAR(50) NOT NULL, -- Hastalık

Email NVARCHAR(50) CONSTRAINT CK\_Email2\_Gmail CHECK (Email LIKE '%@gmail.com'),

NationalID CHAR(11) NOT NULL UNIQUE ,-- TCKimlikNo

AssignedDoctorID INT NULL, -- Atanmış Doktor (Opsiyonel)

FOREIGN KEY (AssignedDoctorID) REFERENCES Doctor(DocId) ON DELETE SET NULL

);

CREATE TABLE Diagnosis (

DId INT IDENTITY(1,1) PRIMARY KEY,

PatientID INT NOT NULL, -- Hasta ID

DoctorID INT NOT NULL,

PatientName varchar(50) NULL,

Symptoms VARCHAR(50) NOT NULL, -- Tanı Açıklaması

DiagnosticTest Varchar(50),

Medicines varchar(50),

FOREIGN KEY (PatientID) REFERENCES Patient(PId) ON DELETE CASCADE,

FOREIGN KEY (DoctorID) REFERENCES Doctor(DocId) ON DELETE CASCADE

);

CREATE TABLE AdminTable\_Log(

UserID INT IDENTITY(1,1) PRIMARY KEY,

[FirstName] VARCHAR(50) NOT NULL, -- Kullanıcı Adı

[LastName] VARCHAR(50) NOT NULL, -- Kullanıcı Soyadı

[Psword] NVARCHAR(50) NOT NULL,

SuperAdmin BIT NOT NULL DEFAULT 0

);

CREATE TABLE Doctor\_Log (

LogID INT IDENTITY(1,1) PRIMARY KEY,

[DocId] INT NOT NULL ,

[DocName] VARCHAR(50) NOT NULL,

[DocGen] VARCHAR(6) NOT NULL,

[Experience] VARCHAR(2) NOT NULL,

[License] VARCHAR(5) NOT NULL,

Specialty NVARCHAR(50) NOT NULL, -- Branş

Phone CHAR(11), -- Telefon

Email NVARCHAR(50),

NationalID CHAR(11) NOT NULL , -- TCKimlikNo

DeletedAt DATETIME NOT NULL DEFAULT GETDATE()

);

CREATE TABLE Patient\_Log (

LogID INT IDENTITY(1,1) PRIMARY KEY,

PId INT NOT NULL,

PName VARCHAR(50),

PAddress VARCHAR(50),

PBirthDate Date,

PPhone VARCHAR(11),

PGen VARCHAR(6),

BloodGroup VARCHAR(3),

MajorDisease VARCHAR(50),

DeletedAt DATETIME NOT NULL DEFAULT GETDATE()

);

CREATE TABLE Diagnosis\_Log (

LogID INT IDENTITY(1,1) PRIMARY KEY,

DId INT NOT NULL,

PatientId INT,

PatientName VARCHAR(50),

Symptoms VARCHAR(50),

DiagnosticTest VARCHAR(50),

Medicines VARCHAR(50),

DeletedAt DATETIME NOT NULL DEFAULT GETDATE()

);

CREATE TABLE LoginPage\_Log (

LogID INT IDENTITY(1,1) PRIMARY KEY,

LoginId INT NOT NULL,

Username VARCHAR(25),

Psword CHAR(6), -- Burada doğru sütun adını kullanıyoruz

DeletedAt DATETIME NOT NULL DEFAULT GETDATE()

);

INSERT INTO Adminn (FirstName, LastName, Psword, SuperAdmin)

VALUES

('Ali', 'Kaya', '123456', 1),

('Ayşe', 'Demir', 'abcdef', 0),

('Can', 'Korkmaz', '654321', 0),

('Hülya', 'Uzun', 'admin1', 0),

('Murat', 'Polat', 'secure', 0);

INSERT INTO LoginPage (Username, Psword, Duty, aID)

VALUES

('admin1', '123456', 'Adminn', 1),

('admin2', 'abcdef', 'Adminn', 2),

('admin3', 'qwerty', 'Adminn', 3),

('admin4', 'zxcvbn', 'Adminn', 4),

('admin5', '111111', 'Adminn', 5);

INSERT INTO Doctor (DocName, DocGen, Experience, License, Specialty, Phone, Email, NationalID)

VALUES

('Dr. Ali', 'Male', '15', '12345', 'Cardiology', '05001111111', 'ali@gmail.com', '11111111111'),

('Dr. Ayşe', 'Female', '10', '23456', 'Dermatology', '05002222222', 'ayse@gmail.com', '22222222222'),

('Dr. Fatma', 'Female', '12', '34567', 'Neurology', '05003333333', 'fatma@gmail.com', '33333333333'),

('Dr. Mehmet', 'Male', '20', '45678', 'Pediatrics', '05004444444', 'mehmet@gmail.com', '44444444444'),

('Dr. Ahmet', 'Male', '8', '56789', 'Orthopedics', '05005555555', 'ahmet@gmail.com', '55555555555'),

('Dr. Elif', 'Female', '9', '67890', 'Gynecology', '05006666666', 'elif@gmail.com', '66666666666'),

('Dr. Can', 'Male', '5', '78901', 'Psychiatry', '05007777777', 'can@gmail.com', '77777777777'),

('Dr. Hülya', 'Female', '14', '89012', 'Surgery', '05008888888', 'hulya@gmail.com', '88888888888'),

('Dr. Murat', 'Male', '11', '90123', 'Radiology', '05009999999', 'murat@gmail.com', '99999999999'),

('Dr. Zeynep', 'Female', '7', '01234', 'Endocrinology', '05000000000', 'zeynep@gmail.com', '12345678901');

INSERT INTO Patient (PName, PAddress, PBirthDate, PPhone, PGen, BloodGroup, MajorDisease, Email, NationalID, AssignedDoctorID)

VALUES

('Hasan Yılmaz', 'Ankara', '1980-01-01', '05011111111', 'Male', 'A+', 'Diabetes', 'hasan@gmail.com', '11112222333', 1000),

('Ayşe Kaya', 'İstanbul', '1990-05-15', '05012222222', 'Female', 'B-', 'Hypertension', 'aysek@gmail.com', '22223333444', 1001),

('Mehmet Can', 'İzmir', '1975-03-10', '05013333333', 'Male', 'O+', 'Heart Disease', 'mehmetc@gmail.com', '33334444555', 1002),

('Fatma Arslan', 'Antalya', '1985-12-20', '05014444444', 'Female', 'AB-', 'Asthma', 'fatma@gmail.com', '44445555666', 1003),

('Ali Demir', 'Bursa', '1995-07-25', '05015555555', 'Male', 'A-', 'Cancer', 'alid@gmail.com', '55556666777', 1004),

('Elif Şahin', 'Mersin', '2000-08-30', '05016666666', 'Female', 'B+', 'Allergy', 'elif@gmail.com', '66667777888', 1005),

('Hülya Kurt', 'Trabzon', '1992-04-14', '05017777777', 'Female', 'O-', 'Kidney Disease', 'hulya@gmail.com', '77778888999', 1006),

('Can Polat', 'Samsun', '1988-02-28', '05018888888', 'Male', 'A+', 'Diabetes', 'canp@gmail.com', '88889999000', 1007),

('Zeynep Uzun', 'Eskişehir', '1999-11-11', '05019999999', 'Female', 'AB+', 'Hypertension', 'zeynepu@gmail.com', '99990000111', 1008),

('Murat Baş', 'Konya', '1983-09-09', '05020000000', 'Male', 'O+', 'Heart Disease', 'muratb@gmail.com', '12345678011', 1009);

INSERT INTO Diagnosis (PatientID, DoctorID, PatientName, Symptoms, DiagnosticTest, Medicines)

VALUES

(1, 1000, 'Hasan Yılmaz', 'Cough', 'Blood Test', 'Paracetamol'),

(2, 1001, 'Ayşe Kaya', 'Fever', 'X-Ray', 'Ibuprofen'),

(3, 1002, 'Mehmet Can**'**, 'Headache', 'MRI', 'Aspirin'),

(4, 1003, 'Fatma Arslan', 'Shortness of breath', 'Spirometry', 'Inhaler'),

(5, 1004, 'Ali Demir', 'Chest Pain', 'ECG', 'Nitroglycerin'),

(6, 1005, 'Elif Şahin', 'Rash', 'Allergy Test', 'Antihistamine'),

(7, 1006, 'Hülya Kurt', 'Nausea', 'Ultrasound', 'Omeprazole'),

(8, 1007, 'Can Polat', 'Back Pain', 'CT Scan', 'Muscle Relaxants'),

(9, 1008, 'Zeynep Uzun', 'Fatigue', 'Blood Test', 'Vitamin Supplements'),

(10, 1009, 'Murat Baş', 'Weight Loss', 'Thyroid Test', 'Levothyroxine');

**EK2**

CREATE TRIGGER trg\_Adminn\_Insert

ON Adminn

AFTER INSERT

AS

BEGIN

-- LoginPage tablosuna kayıt ekleme

INSERT INTO LoginPage (Username, Psword, Duty, aID)

SELECT

CONCAT(FirstName, LastName) AS Username, -- Kullanıcı adı olarak Ad ve Soyad birleştirilir

Psword, -- Şifre aynı kalır

'Adminn', -- Görev olarak 'Adminn' atanır

AdminID -- Admin tablosunun ID'si eklenir

FROM INSERTED; -- INSERTED sanal tablosundan yeni eklenen değerler alınır

END;

**EK3**

CREATE TRIGGER trg\_AdminTable\_Delete

ON Adminn

INSTEAD OF DELETE

AS

BEGIN

-- Silinen kayıtları AdminTable\_Log tablosuna ekle

INSERT INTO AdminTable\_Log (UserId, FirstName, LastName, Psword, SuperAdmin)

SELECT AdminID, FirstName, LastName, Psword, SuperAdmin

FROM DELETED;

-- Adminn tablosundan silme işlemini gerçekleştirme

DELETE FROM Adminn

WHERE AdminID IN (SELECT AdminID FROM DELETED);

-- Kayıtların AdminTable\_Log tablosuna taşındığını belirten mesaj

PRINT 'Record moved to AdminTable\_Log and deleted from Adminn';

END;

**EK4**

create TRIGGER trg\_Doctor\_Delete

ON Doctor

INSTEAD OF DELETE

AS

BEGIN

-- Silinen kaydı Doctor\_Log tablosuna ekle

INSERT INTO Doctor\_Log (DocId, DocName, DocGen, Experience, License,Specialty,Phone,Email,NationalID, DeletedAt)

SELECT DocId, DocName, DocGen, Experience, License,Specialty,Phone,Email,NationalID, GETDATE()

FROM DELETED;

-- Asıl Doctor tablosundan kaydı sil

DELETE FROM Doctor

WHERE DocId IN (SELECT DocId FROM DELETED);

PRINT 'Record moved to Doctor\_Log and deleted from Doctor';

END;

**EK5**

CREATE TRIGGER trg\_Diagnosis\_AfterDelete

ON Diagnosis

AFTER DELETE

AS

BEGIN

-- Silinen kaydın bilgilerini Diagnosis\_Log tablosuna ekliyoruz

INSERT INTO Diagnosis\_Log (DId, PatientID, PatientName, Symptoms, DiagnosticTest, Medicines)

SELECT DId, PatientID, PatientName, Symptoms, DiagnosticTest, Medicines FROM DELETED;

-- Silinen kaydı Diagnosis tablosundan siliyoruz

DELETE FROM Diagnosis WHERE DId IN (SELECT DId FROM DELETED);

PRINT 'Record moved to Diagnosis\_Log and deleted from Diagnosis table';

END;

**EK6**

CREATE TRIGGER trg\_Patient\_Delete

ON Patient

INSTEAD OF DELETE

AS

BEGIN

-- Silinen kaydı Patient\_Log tablosuna ekle

INSERT INTO Patient\_Log (PId, PName, PAddress, PBirthDate, PPhone, PGen, BloodGroup, MajorDisease)

SELECT PId, PName, PAddress, PBirthDate, PPhone, PGen, BloodGroup, MajorDisease

FROM DELETED;

-- Asıl Patient tablosundan kaydı sil

DELETE FROM Patient

WHERE PId IN (SELECT PId FROM DELETED);

PRINT 'Record moved to Patient\_Log and deleted from Patient';

END;

EK7

CREATE PROCEDURE [dbo].[SearchDoctorByName]

@DocName NVARCHAR(50)

AS

BEGIN

SELECT \*

FROM Doctor

WHERE DocName LIKE '%' + @DocName + '%';

END;

**EK8**

CREATE PROCEDURE [dbo].[SearchDoctorsBySpecialty]

@Specialty NVARCHAR(50) -- Uzmanlık alanını parametre olarak alır

AS

BEGIN

SELECT

DocId AS DoctorID,

DocName AS DoctorName,

Specialty AS Expertise,

Phone,

Email,

Experience AS YearsOfExperience

FROM

Doctor

WHERE

Specialty = @Specialty; -- Uzmanlık alanına göre filtreleme

END;

**EK9**

CREATE PROCEDURE [dbo].[GetDoctorLogs]

AS

BEGIN

SELECT \* FROM Doctor\_Log

ORDER BY DeletedAt DESC; -- En son silinen kayıtları göstermek için

END;

**EK10**

CREATE PROCEDURE [dbo].[SearchPatientByName]

@PName VARCHAR(50)

AS

BEGIN

SELECT \*

FROM Patient

WHERE PName LIKE '%' + @PName + '%'; -- Adın kısmi eşleşmesi

END;

**EK11**

CREATE PROCEDURE [dbo].[GetPatientLogs]

AS

BEGIN

SELECT \* FROM Patient\_Log

ORDER BY DeletedAt DESC; -- En son silinen kayıtları göstermek için

END;

**EK12**

CREATE PROCEDURE [dbo].[SearchDiagnosisBySymptoms]

@Symptoms NVARCHAR(50) -- Semptom parametresi

AS

BEGIN

SELECT \*

FROM Diagnosis -- Teşhis tablosu adı

WHERE Symptoms LIKE '%' + @Symptoms + '%';

END;

# **10-KAYNAKÇA**

-BTK Akademi-c# programlama eğitimi

-Sql Server Sorgularla Adım Adım-Murat Yücedağ

-Turkcell geleceği yazanlar-C# Form ile Görsel ve Nesne Tabanlı Programlama eğitimi