Kitap Dükkanları Zinciri

Veritabanı Yönetimi Dönem Projesi

Bilal Latif Ozdemir

91200001179

Bilallozdemir@outlook.com

Giris

Veritabanı Yönetimi dersi dönem projesi kapsamında sunulan opsiyonlardan “Bir Veri-tabanı Uygulaması” seçeneğini seçilmiştir. Proje SQL Server üzerinde oluşturulan bir ilişkisel veri tabanını ve bu veritabanına bağlı çalışan Python flask/dash kütüphanelerinden yararlanarak oluşturulan detaylı bir web uygulamasından oluşturulmaktadır.

Dönem boyunca dersten edilen kazanımlar uygulamaya dökülerek oluşturulan ilişkisel veri tabanına konuya uygun tablolar oluşturulmuştur. Tablolarda yer alan kolonlara uygun veri tipleri seçilmiş ve kolonların alacağı değerlere dair kısıtlamalar tanımlanmıştır. Ve yine tablolar arasındaki ilişkiler tanımlanmıştır.

Veritabanına bağlı çalışan web uygulaması senaryoya uygun bir şekilde kullanıcı için önem düzeyi yüksek verilerin bir arayüz üzerinden okunması, yazılması, ve düzenlenmesine izin verecek şekilde tasarlanmıştır.

Projenin Senaryosu

Proje Önerisi aşamasında da belirtildiği gibi projenin dayandığı senaryo şu şekildedir:

Türkiyenin farklı şehirlerinde onlarca şubesi bulunan bir Kitapçılar Zinciri'ne sahibiz. Bu sahibi olduğumuz kitapçı dükkanlarında satışı gerçekleştirilen kitapların ve alışveriş yapan müşterilerin bilgilerini kayıt altına almak, bunları şubelere göre görüntüleyebiliyor olmak istiyoruz.

Bununla birlikte tüm bu şubelerimizde çalışanların kullanabileceği, gerekli kitap/ müşteri / alışverişler ile ilişkili bilgileri kayıt altına almakta ve müşteri ihtiyaçlarına daha hızlı karşılık vermekte kullanılabilecek bir arayüz/ web sayfasına ihtiyaç duyuyoruz.

Web sayfasını tüm şubelerimizdeki çalışanlar kullanabilmesini ve müşterilerin "Stephen King'in Doktor Uyku kitabı elinizde var mı?" gibisinden sorularını kolay bir şekilde sorgulayabiliyor olmasını istiyoruz. Satış işlemlerini kayıt altına almak ve biriken verilerle tüm şubelerin karlılığa etkisi ve genel durumlarını kolay bir şekilde gözlemleyebiliyor olmak istiyoruz.

Bu Projenin Seçiliyor Olma Sebebi

Genel olarak veritabanı oluşturma ve tablo/ ilişki detaylarına yeteri kadar hakim olmadığının farkındalığı ile böyle bir proje seçilmiştir. Baştan sona veri tabanının tasarlanması, gerekli kısıtların tanımlanması, bunun bir arayüz ile son kullanıcıya sunulmasının böyle bir projeyi taçlandıracağı düşünülmüştür.

Günlük hayatta kullanılan telefon uygulamaları, bilgisayar oyunları, sosyal ağ platformları her alanda bir veritabanını kullanırken baştan sona son kullanıcıya sunulacak bir uygulamada tüm sürece hakim olma ve zorluklar ile başaçıkarak öğrenme amacıyla böyle bir proje tercih edilmiştir.

Ve onca zorluğun sonunda çok da faydası görüldü tarafımca.

BÖLÜM 1: VERİTABANI TASARIMI

Tabloların Belirlenmesi

Senaryoya uygun veritabanı olarak bir No SQL uygun değildir. Çalışmanın ihtiyaçlarını gidereceği ve ücretsiz versiyonun etkinliği sebebi ile SQL Server üzerinde bir veritabanı tasarlanması uygun görülmüştür.

İlk haftalarda kitapçılar zinciri senaryosunun ihtiyaçları üzerine çalışılmıştır. Gerekli olan tablolar, bu tablolarda oluşturulan kolonlar/ değişkenler belirlenmiştir. Bu aşamada yer yer online kitap satış siteleri gözlemlenmiş ve oradaki hesaplar incelenmiştir.

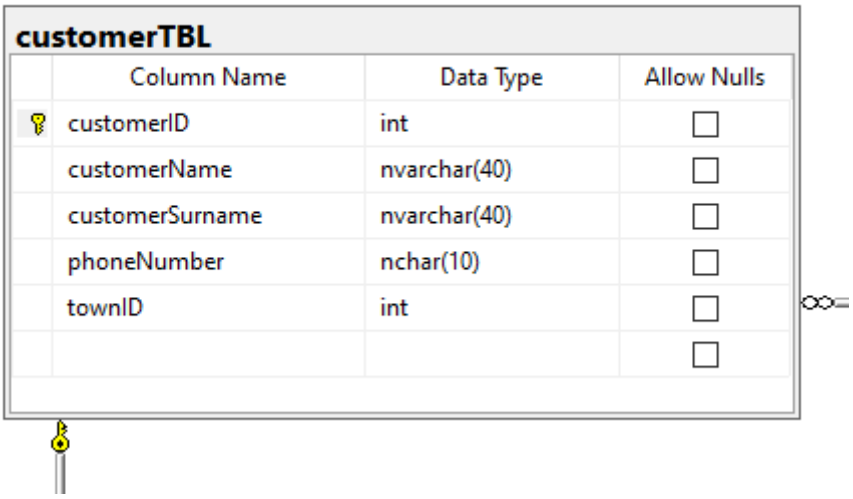
Son durumda karar verilen tablolar şu şekildedir:

- Kitap Tablosu,
- Tür Tablosu (kitap türü),
- Yazar Tablosu,
- Yayın Evi,
- Şehir Tablosu,
- İlçe Tablosu,
- Şube Tablosu,

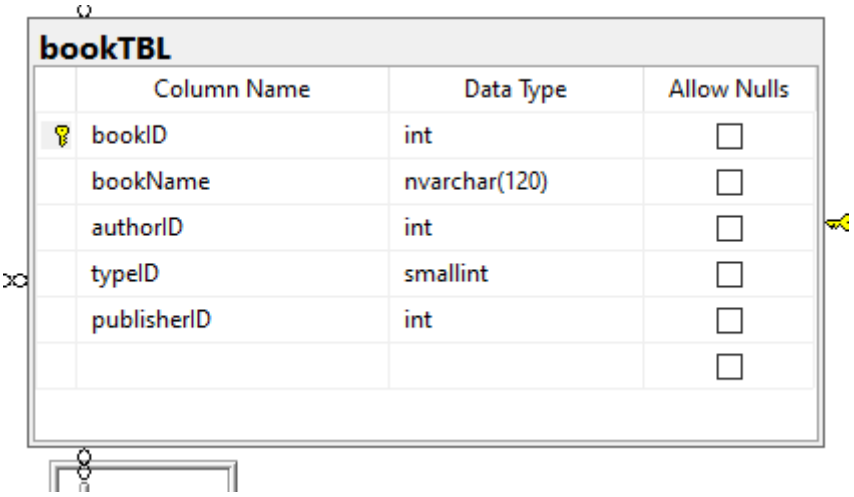
- Satış İşlemleri Tablosu,
- Müşteri Tablosu.

Kolonların/ değişkenlerin veri tiplerinin nasıl olması gerektiğine karar aşamasında öncelikli olarak bir şablon çıkartılmıştır. Fakat web scrabing yöntemi ile elde edilen kitap bilgileri ışığında “kitap isimleri en fazla kaç karakter olabilir? Yazar isimleri kac karakter olmalı?” gibi sorular ışığında tekrar düzenlenmiştir.

Toplam kolon sayısı 30 civarında olduğunu için her bir kolon ayrıntılı olarak incelenmeyecektir. Örnek olaması açısından Customer (müşteri) Tablosu aşağıdaki gibidir. Senaryoya göre müşteri tablosunda customerName, customerSurname, phoneNumber, ve müşterilerin bulundukları ilçelerin tutulduğu ilçe tablosuyla ilişkide olan townID kolonları uygun görülmüştür. Yapılan inceleme sonrasında veri tiplerine karar aşamasında daha sonra bahsedilecek olan ilişkileri normal formlara uydurmak için tekrarlı kolonlar(bu durumda townID) integer olarak farklı tablolarla ilişkilendirilmiştir. Ve primary key olarak tanımlayıcı ve sorgularda indekslemesiyle hız kazandıracak olan customerID kolonu oluşturulmuştur.



bookTBL (kitaplar Tablosu)’nda ise yine tekrarlı olan kolonların bulunduğu diğer tablolarla bağlantılı typeID, authorID,publisherID kolonları int olarak tanımlanmıştır ve yapılan gözlemlerde kitap isimlerinin 110 karaktere kadar olabildiğinden dolayı kitap isimleri max 120 olacak şekilde nvarchar olarak tanımlanmıştır. Ve yine primary key olarak atanan ve otomatik artan olarak tanımlanan bookID kolonu eklenmiştir.



Genel olarak veri tipleri ve kolonların seçimi bu şekilde olup her bir tablo tek tek incelenmeyecektir. Bahsedilenler ışığında sonraki aşamalarda gösterilecek ilişki diyagramında tüm tabloların kolon isimleri ve veri tipleri bilgileri görüntülenebilmektedir.

Ek olarak tüm Primary Key kolonlar otomatik artan, 0'dan başlayan ve 1'er artımlı olarak ayarlanmıştır.

İlişkilerin Tanımlanması

İhtiyaçlar ve genel veri tipleri belirlendikten sonra tablolar arasındaki ilişkiler tanımlanmıştır. İlişkileri bulunan kolonların tiplerinin uyumu aşamasına dikkat edilmiştir. Normal formlara uyumun dikkate alındığı bir şekilde tekrarlardan olabildiğince kaçınılmıştır. İlişki diyagramı bir sonraki sayfada tam boy olarak yer almaktadır.

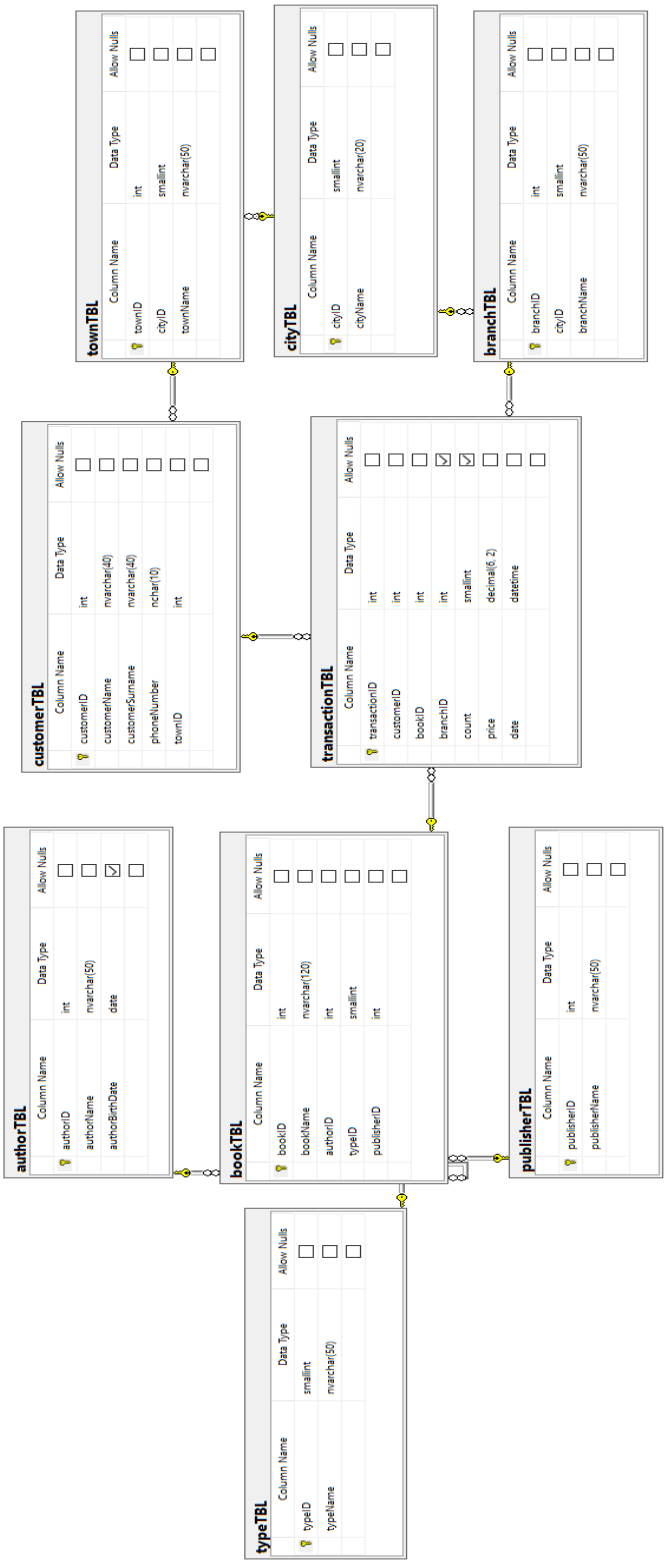
Buradaki ilişkilere tek tek değinilmesi uygundur.

En altlardan başlayarak anlatacak olursak Kitap tablosu (bookTBL) kitaplar hakkında bilgi tutmaktadır. Bu bilgiler kitap ismi (bookName), yazar ismi (authorName), kitap türü (bookType), yayınevi adı (publisherName) şeklindedir. Bu bilgiler her kitap için mevcuttur. Fakat sorun şu ki belki aslında 100000'lerce binlerce kitap aslında aynı 200 yayınevinden çıkar. 2000 yazar tarafından yazılmıştır ve belki sadece 40 tane farklı kitap türü vardır. Yani bu demek olur ki 100000 kitap hakkında bilgi tutarken aynı 40 tür bilgisini belki 3000'er kez tekrar tekrar yazıyor olunacak. Veya bu durum yazar ismi ve yayın evi ismi için de tekrar ediliyor olunacak. Bu gibi tekrar durumlarından kaçınmak için bilgilerin tekrar edeceği durumlarda tutulacak veri boyutunu azaltmak için ilişkiler kurarak asıl tablolarda diğer tablolardaki bilgileri ilişkiyi ifade etmek için primary key'ler ile ilişkilendirilmiştir. Bu şekilde tekrardan kaçınılmıştır ve normal formlara uygunluk sağlanmaktadır.

bookTBL ve kendi bilgilerini tuttuğu ilişkili olan tablolar (publisherTBL, authorTBL, typeTBL) bu şekildeydi. transactionTBL (Satış bilgileri) tablosunun komponentleri aslında markette bile benzer formattadir. Satış bilgileri çok büyük ölçüde başka alan bilgilerinden oluşur. Bizim senaryomuzda da benzer bir durum olduğundan satış bilgileri tablosu içerisinde bulunan müşteri, şube, satılan kitap bilgileri aslında her satışta tekrar etmesi olası değişkenlerdir. Bu kolonlardaki tekrardan kaçınabilmek ve düzeni sağlamak için bu bilgilerin tutulduğu farklı tablolarla primary key – foreign key ilişkileri kurulmuştur.

Yine her şube (branch)'nin bulunduğu şehrin bilgisinin tutulması, her müşterinin de bulunduğu ilçenin bilgisinin tutulması ihtiyacı ve tekrardan kaçma amacıyla gerekli ilişkiler tanımlanmıştır. Müşterilerin

sadece bulunduğu ilçe bilgisinin tutulması, il bilgisinin tutulmaması da aslında gereksiz yere 1 kolon tutulmasını engelleyen çok yerinde bir uygulama olmuştur. Çünkü aslında primary key'i belli olan bir ilçe bilgisi elimizde varken şehirri ayrıyeten tutmaya gerek yoktur. Bu durumda mesela Manisa Alaşehir ve Konya Alaşehir ilçeleri birbirine karışmayacaktır, çünkü her 2 ilçe farklı primary keyleri ile hangi illere ait oldukları bilgisini de taşımaktadırlar aslında.



Gerekli Kısıtlar

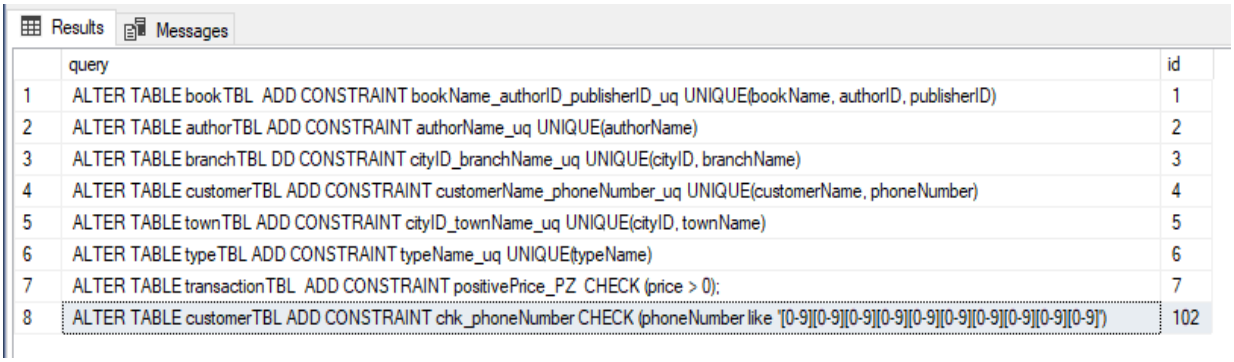
Tablolar oluşturulup, gerekli kabul edilebilir veri tipleri girilmesine rağmen tüm işlemler henüz bitmemiştir. Veri tipleri her ne kadar belirlenmiş de olsa bazı hatalara açık kapılar mevcuttur. En başta eksik değer hataları ile karşılaşılma ihtimali olsa da bunlara izin verilmediği ilişki tablosunda da sunulmaktadır. Bazı hatalara sistem yine de açıktır ve bunların engellemesi gereklidir. Birkaç tanesine örnek verecek olursak mesela bir telefon numarasının 0'dan sonra 10 karakter olacağı herkes tarafından bilinmektedir. Veri tipi olarak da 10 adet sayıyı girmeye uygun bir veri tipini seçmiş de olsak hala kullanıcılar eksik olarak girebilirler numaralarını. Veya kullanıcılar tam olarak da girseler bu boşluğu harf girmeye çalışabilirler. Bu durum telefon numrasının her bir 10 karakteri için 0-9 aralığında rakamların girilebilir olduğunu belirten bir kısıt koyularak yanlışlık/ hata ihtimalini engellenebilir.

Diğer bir uygulanması gereken kısıt ise alışverişlerde girilebilir ürün fiyatının negatif olma durumu olmalıdır. Cunku bildiğiniz gibi bir ürün fiyatı negatif olamaz. Her ne kadar veri değişken tipi olarak decimal tanımlamış olsak da negatif değerlerin girilmesini de engellememiz gerekir.

Diğer bir dikkate alınması gereken kısım ise aynı müşterilerin ve aynı kitapların veritabanına tekrar tekrar kaydedilmesinin önüne geçmektir. Bunu ise bir müşteri tablosuna aynı değelerden oluşa / tekrarlı bir şekilde müşteri ve telefon numarası 2lisinin girilmemesi gerektiğidir. Böyle bir kısıt koyma sebebim ise: Bir aileden baba ve çocukları farklı farklı kendi isimlerimlerine kayıt oluşturtabilir, ve aynı telefon numaralarını kullanabilirler kayıt için (çocuğun telefonu olmayabilir). Böyle bir durumda isimler farklı olacaktır, fakat telefon numarası aynı olacaktır. Bu sorun değildir. Fakat kişi adı ile telefon numarası aynı ise aynı kişi için 2. Tekrarlı hesap oluşturuluyor demektir. Bunun önüne geçilmelidir. Burada müşteri soyismini kısıta dahil edilmemiştir. Çünkü mantıksal olarak gerek yoktur.

Diğer bir kısıt ise kitap isimleri ile ilgilidir. Bir yazarın aynı kitabı farklı yayın evleri tarafından satışa çıkarılabilir. Yani bir kitabın adı ve yazar adı aynı olabilir, fakat yayın evi bilgileri değişecektir. Eğer ki bu 3'lünün aynı olduğu bir durum var ise kitaplar tekrarlı olarak kaydediliyor demektir. Bu hatanın önüne kısıt koyarak geçilmiştir.

Diğer kısıtların üzerinde teker teker durulmayacaktır. Aşağıdaki tabloda çalıştırılan kısıtlar kayıt sorguları ile birlikte kayıt altına alınmıştır, oradan incelenebilir.


Results Messages

	query	id
1	ALTER TABLE bookTBL ADD CONSTRAINT bookName_authorID_publisherID_uq UNIQUE(bookName, authorID, publisherID)	1
2	ALTER TABLE authorTBL ADD CONSTRAINT authorName_uq UNIQUE(authorName)	2
3	ALTER TABLE branchTBL DD CONSTRAINT cityID_branchName_uq UNIQUE(cityID, branchName)	3
4	ALTER TABLE customerTBL ADD CONSTRAINT customerName_phoneNumber_uq UNIQUE(customerName, phoneNumber)	4
5	ALTER TABLE townTBL ADD CONSTRAINT cityID_townName_uq UNIQUE(cityID, townName)	5
6	ALTER TABLE typeTBL ADD CONSTRAINT typeName_uq UNIQUE(typeName)	6
7	ALTER TABLE transactionTBL ADD CONSTRAINT positivePrice_PZ CHECK (price > 0);	7
8	ALTER TABLE customerTBL ADD CONSTRAINT chk_phoneNumber CHECK (phoneNumber like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')	102

Veri Ekleme ve Silme Sırası

Simdiye kadar anlatılan basamaklarda uygun formatta tablolar oluşturulmuştur. Gerekli ilişkiler kurulmuş ve gerekli kısıtlar verilmiştir, boş tabloları doldurma sırası gelmiştir. Bu tabloları doldurma kısmında direkt olarak bir kitap ekleyeyim demek mümkün değildir. Tablolar arası ilişki durumları ve

eksik değerlere izin verilmediği durumları da gözönünde bulundurularak veri yazma ve silme işlemleri bir mantık sırasını takip etmelidir.

Örneklendirmek gerekir ise mesela direkt olarak kitap tablosunu doldurmak mümkün değildir. Çünkü kaydedilmek istenilen kitabın yazar adı, türü ve yayın evi bilgileri sistemde kayıtlı olmadığından ilk olarak onlar eklenmelidir. Benzer bir sıra müşteri ve satış işlemleri tablosu için de geçerlidir. Mesela İlçeler tablosu doldurulmadan müşteriler tablosu doldurulamaz. Veya bir şehir belirtilmeden o şehrin bir ilçesi belirtilemez. Bu sebeple tabloların doldurulma sırasında bir mantık izlenmelidir.

Benzer şekilde ilişkilerden dolayı silme işlemlerinde de rastgele bir sıra izlenemez. Bu sefer tam tersi bir sıra ile işlemlerin gerçekleştirilir. İller tablosunu direkt olarak silinemez, çünkü bu tablo verilerini barındıran ilçeler tablosu buna izin vermez. Yine kitap türleri tablosu direkt olarak silinemez, çünkü bu türleri kullanan kitaplar vardır ve bu tablo buna izin vermez.

Bu konu çok önemli olup mantıksal sıra birkaç farklı şekilde gerçekleştirilebilir. Sıfırdan bu veritabanını doldururken ilk başta itap türü veya yazar tablosu verileri girilebilir. Çünkü bu tabloların altında yer alan başka tablolar mevcut değildir. Ve tabloları silerken de ilk başta satış işlemleri tablosu silinir fakat 2. Sırada müşteri tablosu veya kitap tablosu silinebilir, bu 2'si arasında bir öncelik de yoktur.

Verilerin Veritabanına Yazılması

Veritabanına verilerin yazılması aşamasında şuanki çalışma bir gerçek hayatı yansıtması için ve bir proje olduğundan dolayı türetilmiş veriler kullanılarak doldurulmuştur. Proje uygulama dosyalarının içerisinde writeIT.ipynb isimli notebook içerisinde türetilmiş olan verilerin insert edilme adımları görüntülenebilir.

Projeye veri elde etme kısımlarında müşteri isimleri internetten bulunan isimler ve soyisimler listeleri bulunarak bu isimler rastgele olarak birbirleri ile eşleştirilmesi ile elde edilmiştir. Kişi isimleri gerçek kişileri yansıtmamaktadır. Aynı şekilde kişilerin telefon numaraları da türetilmiştir. Telefon numaraların 10 hanesi ilk 3 hanenin 545, 548 gibi kullanılan operatör kodları ile birlikte ve bir mantık çerçevesinde türetilmiştir. Kişilere ilçe atamaları da yine random olarak atanmıştır.

Kitap tablosu için gerçeği yansıtması önemli bir kriter olarak görülmüştür bunun için random atama yapmak doğru değildir. Kitap isimleri ve yazar isimleri bilgileri python ile web scrabing yapılarak çekilmiştir. Kitap isim, yazar bilgileri yayın evleri ve kitap türleri ile ise rastgele eşleştirilmiştir.

Şehir ve ilçe bilgileri de yine internetten bulunup gerçeği yansıtacak şekilde kaydedilmişir.

Kitapevleri, şube isimleri, yazar doğum günleri ve satış işlemleri bilgileri random türetilen bilgiler ile doldurulmuştur. Satış işlemleri tablosu ilişkiler uygun olarak şekilde doldurulmuştur.

Aslında bildiğiniz gibi ilişkiler izinler dışında veri kaydetmeyi engellemektedir. Bu ilişkilerin çerçevesine uygun olarak herhangi bir hata olmaksızın her insert işlemi üzerine yoğun bir şekilde çalışılarak bir mantık çerçevesinde kaydedilmiştir.

Örnek olarak kullanılan insert komutlarından birkaçını örnek olarak altta görülebilir. Yazılımın backend kısmında gerekli yerlerde çağırılan bu 3 fonksiyon dinamik olarak içerisine aldığı değerleri veritabanına kaydetmektedir. Aslında dinamikleştirilen kısımları ile birlikte bir bütün olarak kod satırlarının birebir SQL Server'a uygun olduğu görülmektedir. Aslında direkt olarak Python – SQL bağlantısı yapılarak bu işlemi sadece python üzerinde gerçekleştirilmiştir.

```
54
55  def addAuthor(authorName_, authorBirthDate_):
56      conn = conn_()
57      cursor = conn.cursor()
58      insertQuery = f"INSERT INTO authorTBL (authorName, authorBirthDate) VALUES ('{authorName_}', '{authorBirthDate_}')"
59      cursor.execute(insertQuery)
60      conn.commit()
61      conn.close()
62
63
64  def addCustomer(customerName_, customerSurname_, phoneNumber_, townID_):
65      conn = conn_()
66      cursor = conn.cursor()
67      insertQuery = f"INSERT INTO customerTBL (customerName, customerSurname, phoneNumber,
68                      townID) VALUES ('{customerName_}', '{customerSurname_}', {phoneNumber_}, {townID_})"
69      cursor.execute(insertQuery)
70      conn.commit()
71      conn.close()
72
73
74  def addTransaction(customerID_, bookID_, branchID_, count_, price_):
75      conn = conn_()
76      cursor = conn.cursor()
77      insertQuery = f"""INSERT INTO transactionTBL (customerID, bookID, branchID, count, price)
78                          VALUES ({customerID_}, {bookID_}, {branchID_}, {count_}, {price_})"""
79      cursor.execute(insertQuery)
80      conn.commit()
81      conn.close()
82
```

Bir bütün olarak veritabanını içerisi dolu ve tam fonksiyonel bir şekilde görüntülenebilmesi amacıyle bu veri yazma işlemleri tabştan yapılmıştır. Amaç, daha sonraki veri kayıtlarının ve sorguların web arayüz üzerinden yapılmasıdır.

BÖLÜM 2: UYGULAMA

Uygulamanın Çerçevesi

Veritabanı tasarımı bu aşamaya kadar tam olarak yapılmıştır. Buradan sonrasında veritabanında kayıtlı verilerin ulaşılacağı ve kayıt işlemlerinin gerçekleştirilebileceği diğer bir deyişle senaryomuza uygun bir şekilde ihtiyaçların giderilebileceği bir uygulama ihtiyacı bulunmaktadır. Bu uygulama gerçekten de bu tarz bir şirkette “al kullan” denilebilecek seviyede ayrıntılar ile ve ihtiyaçları giderir nitelikte olmalıdır, oldu da.

Web uygulamasının genel amacı teknik olamayan/ sql kullanmaya gerek kalmadan kişilerin hata yapmadan gerekli kayıtları, sorguları, düzenlemeleri yapabileceği kullanımı kolay bir yapı sunmaktır. Bu amaç doğrultusunda yetkinliklerim dahilinde Python programlama dilinden faydalanmış bulunmaktayım.

Uygulama 5 web sayfasından oluşmakta olup, satış işlemlerinin gerçekleştirildiği (ekleme, silme, sorgulama) ekranı, kitap ve müşteri (ekleme, silme, sorgulama) ekranı, diğer alt tablolara ait ekleme ve silme işlemlerinin yapıldığı bir ekran ve genel durum hakkında bilgi veren bir dashboard/izleme ekranından oluşmaktadır.

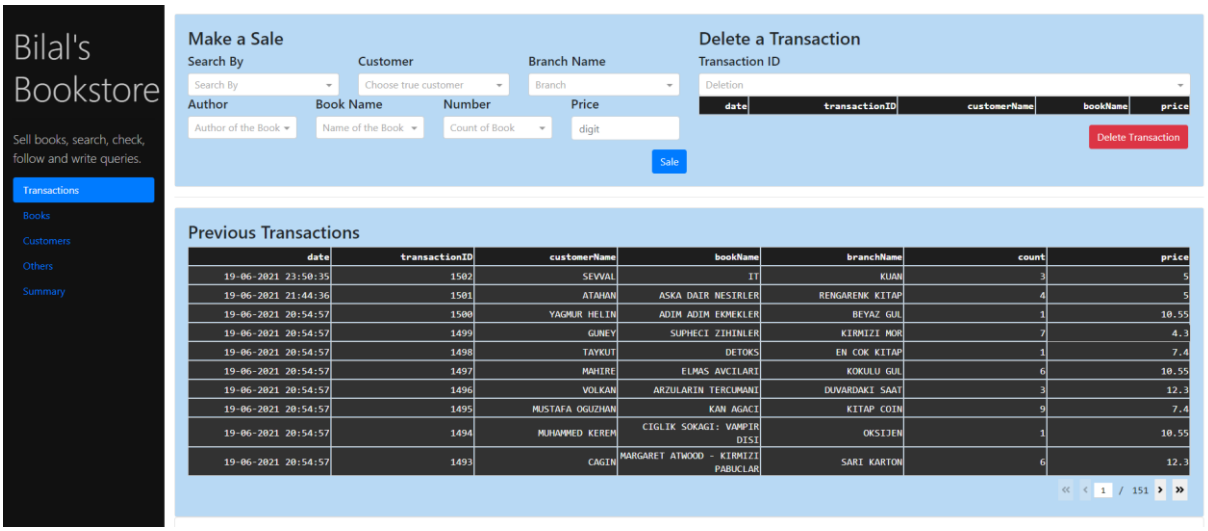
Uygulama tarafından yapılan her ekleme/ silme / güncelleme ve sorgu işlemi SQL sorguları vasıtasıyla gerçekleştirilmektedir.

Web Sayfa İşlevleri

Bazı web sayfalarının görüntüleri hemen altta yer almaktadır. Burada insert aşamasında kullanılan kutulardaki bilgiler direkt olarak SQL'den select yapılan komutlar vasıtasıyla çağırılıyor olup. İlişkiler dışında bir adım atmayı engellemek için ve kullanıcının bunları kontrol etmesine gerek olmasını gerektirmeksizin kolay bir şekilde yapılması amacı ile oluşturulmuştur.

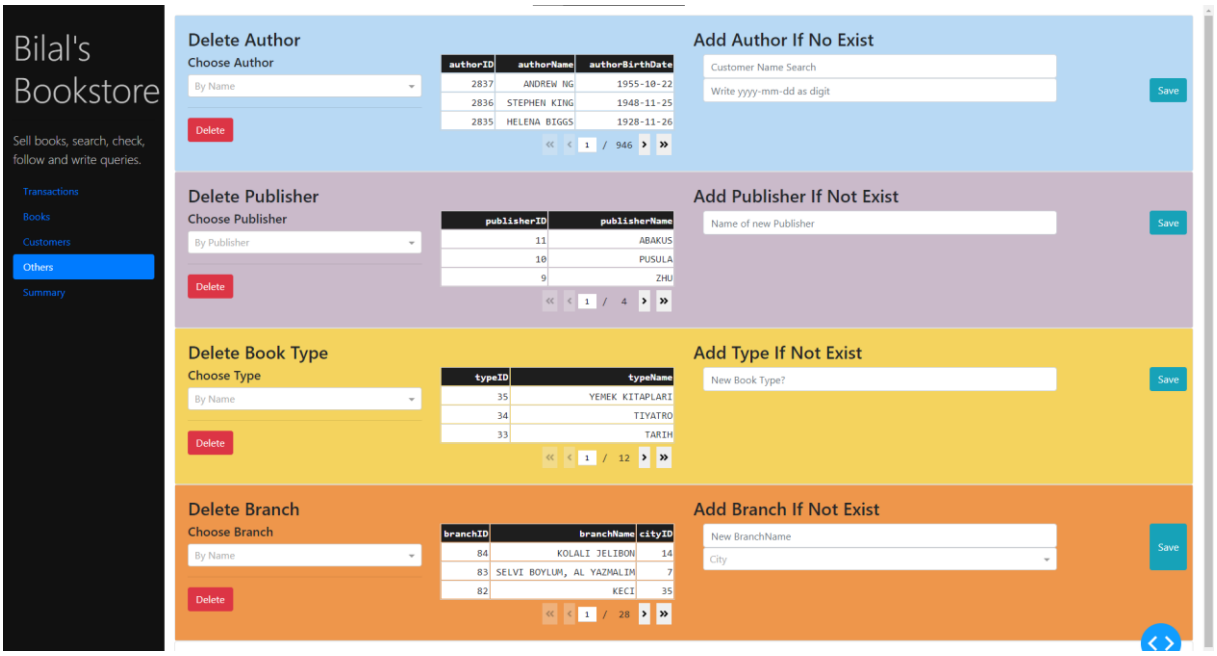
Previous Translations tablosu da yine kullanıcının direkt olarak diğer tablo ID'lerini görmesi ve anlamsız bir süreç işlenmesini engellemek için sql tarafından INNER JOIN içeren son eklenen translation'ın en üstte olduğu şekilde bir sorgu çalıştırmaktadır.

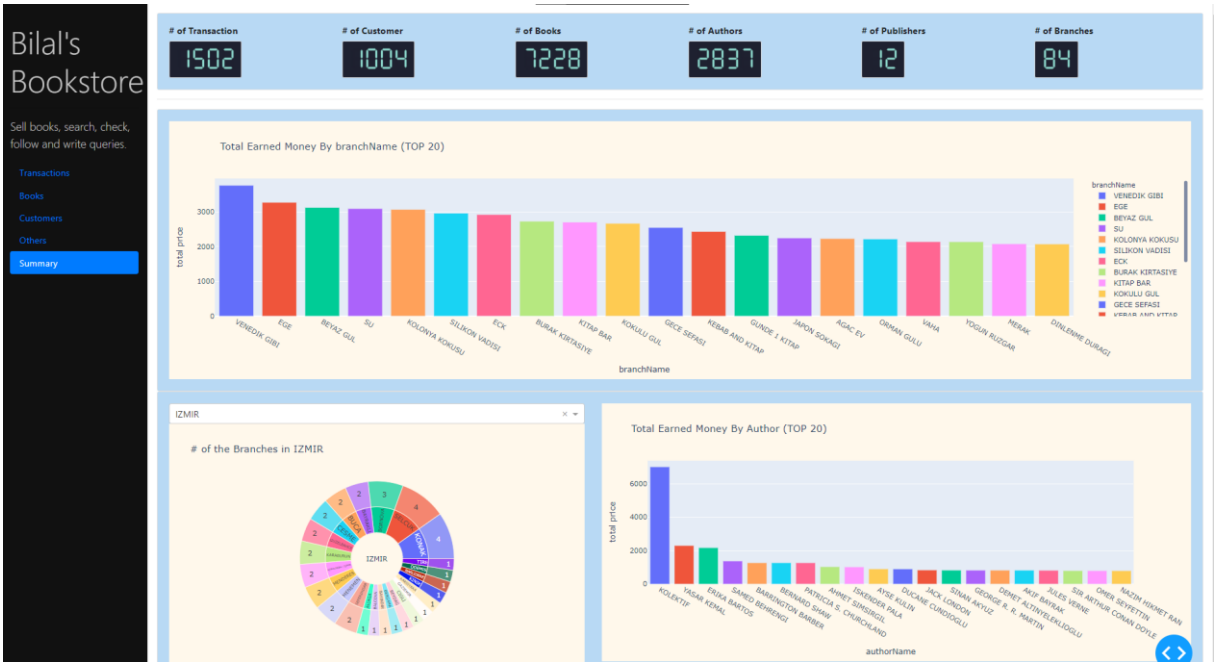
```
def transactionList1():
    conn = conn_()
    sqlQuery = """SELECT transactionTBL.transactionID, transactionTBL.date AS [date],
        CUSTOMERTBL.customerName, BOOKTBL.bookName,BRANCHTBL.branchName,
        transactionTBL.count, transactionTBL.price from transactionTBL
            INNER JOIN CUSTOMERTBL ON transactionTBL.customerID=CUSTOMERTBL.customerID
            INNER JOIN BOOKTBL ON transactionTBL.bookID=BOOKTBL.bookID
            INNER JOIN BRANCHTBL ON transactionTBL.branchID=BRANCHTBL.branchID
        ORDER BY transactionTBL.date DESC"""
    df = pd.read_sql(sqlQuery, conn)
    conn.close()
    if (df.shape[0] > 0):
        df['date'] = [i.strftime("%d-%m-%Y %H:%M:%S") for i in df['date']]
    return df
```



Diğer hemen hemen tüm sorgular benzer bir mantıklar hazırlanmış fonksiyonlar vasıtası ile çağırılıyor olup içlerinde birebir SQL sorguları çalıştırmaktadırlar.

Uygulama kayıt işlemleri ve silme işlemleri gerçekleştiğinde bildirim vererek işlemin gerçekleştiği konusunda kullanıcıyı bilgilendirmektedir.

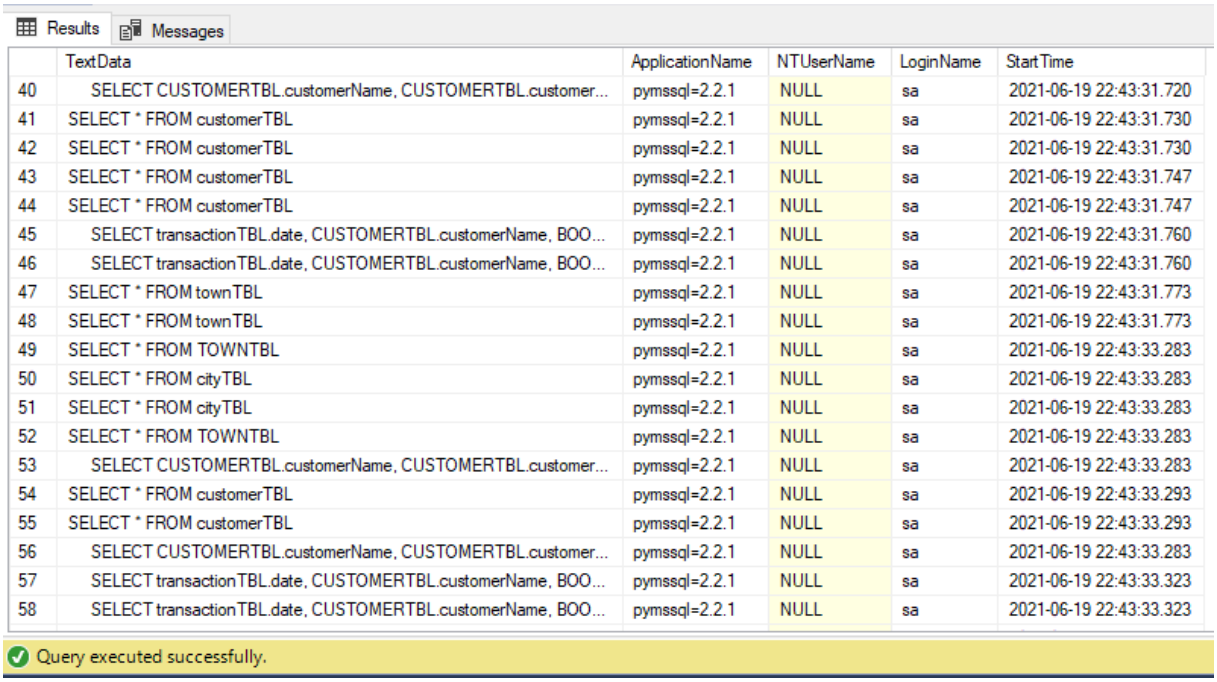




BÖLÜM 3: SQL PROFILER

Uygulamamızda kullanıcıya faydalı olacak bir SQL Server ek aracı olan SQL Profiler ile de ek bir yetenek kazandırıp gerçekleşen sorguları izleme ve yapılan sorgular ile bu sorguları kimlerin hangi saatlerde gerçekleştirdiğini gözlemlemek mümkündür. Kişilerin yapması gerekenin dışında değişiklikler yapıp yapmadığını ve hangi saaatlerde nasıl bir youpunluk olduğunu gözlemlemek için

SQL Profiler'ı kullanmak faydalı olabilir.



Results | Messages

	TextData	ApplicationName	NTUserName	LoginName	StartTime
40	SELECT CUSTOMERTBL.customerName, CUSTOMERTBL.customer...	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.720
41	SELECT * FROM customerTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.730
42	SELECT * FROM customerTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.730
43	SELECT * FROM customerTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.747
44	SELECT * FROM customerTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.747
45	SELECT transactionTBL.date, CUSTOMERTBL.customerName, BOO...	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.760
46	SELECT transactionTBL.date, CUSTOMERTBL.customerName, BOO...	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.760
47	SELECT * FROM townTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.773
48	SELECT * FROM townTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:31.773
49	SELECT * FROM TOWNTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.283
50	SELECT * FROM cityTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.283
51	SELECT * FROM cityTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.283
52	SELECT * FROM TOWNTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.283
53	SELECT CUSTOMERTBL.customerName, CUSTOMERTBL.customer...	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.283
54	SELECT * FROM customerTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.293
55	SELECT * FROM customerTBL	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.293
56	SELECT CUSTOMERTBL.customerName, CUSTOMERTBL.customer...	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.283
57	SELECT transactionTBL.date, CUSTOMERTBL.customerName, BOO...	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.323
58	SELECT transactionTBL.date, CUSTOMERTBL.customerName, BOO...	pymssql=2.2.1	NULL	sa	2021-06-19 22:43:33.323

Query executed successfully.

Ustteki görselde SQL Profiler'ın uygulamamızın isteklerini dahilinde gerçekleşen işlemleri nasıl gerçekleştirdiği görüntlenebilmektedir. Pymssql Python içerisinde sql bağlantısı için kullandığımız kütüphane olup "sa" kullanıcı adı da bu pymssql uygulamasında bağlantı için kullandığımız kullanıcı adı bilgisidir.

SQL Profiler'ı Web Uygulamasına eklemedik çünkü SQL Profiler her SQL kapattığımızda veya bilgisayarı kapatıp açtığımızda kapanan bir servis. Ve tekrar tekrar çalıştırılması gerekiyor olduğundan ek bir aşama çıkartıyor karşımıza.

Bir diğer sorun ise SQL Profiler sadece sorgular değil sisteme dair de kayıtları tutuyor olduğundan kısa süre içerisinde kendi kendine çok büyük boyutta veri üretmeye başlıyor. Ve bu projedeki gibi veri boyutu sayısı küçük olan bir veritabanında birkaç saat içerisinde SQL Profiler'ın tabloya yazdığı veri miktarının çalışmamızdaki tablo verilerinin üstüne çıkması olası. Bunun için uygulamaya eklemeyip sadece bir paragraf olarak düşmeyi uygun gördüm.

SONUC

Proje Önerisi aşamasında da belirttiğim üzre haftada 5 gün zaten işim dolayısı ile (Veri Bilimci olarak çalışıyorum) SQL ve NoSQL yapılarını aktif kullanıyorum. Fakat kullanımların genellikle select ve insert methodlarını kullanmakla kısıtlı çoğunlukla. Geriye kalan kısımları team lead veya IT tarafından yapılan işlemler. Tamamiyle benim kontrolümde olan veritabanları ise genellikle test çalışmalarında kullanıyor olduklarım. Hal böyle olunca detayların üzerinde duracak çok şansım / zoraki durum olmuyor.

Ben bu projeyi yaparken baştan sona veritabanının yaratılmasından uygulama ile entegrasyonu ve uygulama ile haberleşmesi kısımlarını sadece benim kontrolüm ve düzenlemem ile gerçekleştiriyor oldum. Bu aynı zamanda tasarıma, diğer taraftan senaryonun ihtiyaçlarına bir bütün olarak yaklaşmamı ve güzel bir denge sağlamak için pürdikkat çalışmamı gerektirdi.

Projeyi yaparken çok çok keyif aldım. Mevcut bilgilerimi sonuna kadar kullandım, gerektiğinde araştırdım/ öğrendim, takılı kaldığım birkaç sorun ile düşüne düşüne birkaç gün geçirdim ve sonuçlarına ulaştım.

Çalışmam çok kapsamlı olmasından kaynaklı yazılım tarafından çok küçük görsel hatalar mevcut, fakat büyük ihtimalle herhangi bir kullanıcı farketmeyecektir. Odağım çoğunlukla database tarafının sorunsuz olması üzerineydi. Bir bütün olarak şuan ortaya çıkardığım projeden son derece memnunum.