T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROJE RAPORU

ÖĞRENCİNİN ADI SOYADI: ALEYNA KILIÇ, SÜMEYYE ÇETİNKAYA

ÖĞRENCİNİN NUMARASI: G231210071, G231210065

ÖĞRENCİNİN GRUBU: 2B

ÖĞRETİM GÖREVLİSİ: Dr.Öğr.Üyesi İsmail ÖZTEL

ÖDEVİN KONUSU: KUAFÖR RANDEVU SİSTEMİ

SAKARYA

ARALIK, 2024

VERİTABANI YÖNETİM SİSTEMİ

Ödevimiz bir kuaför sisteminin randevu ve müşteri yönetimi sistemi hazırladık. Veritabanımız aşağıdaki işlemleri gerçekleştirmekte:

Müşteri Yönetimi: Müşteri bilgileri (ad, soyad, iletişim bilgileri), randevu düzenlemeleri (randevu saati. randevu tarihi, müşterinin hangi işlemi istediği), verilen hizmetler(saç kesimi, bakım) ve müşterilerin yorumları (yorum, verilen puanlar) gibi işlemler veritabanımızda yönetilir.

Çalışan Yönetimi: Çalışanların görev tanımlarını ve kişisel bilgilerini içinde tutup gerektiğinde işlem yapma veritabanımızda mevcut.

Şube Yönetimi: Fatura (toplam tutar, ödeme türü), tedarikçi (tedarikçinin adı, iletişim bilgileri), sipariş (toplam fiyat) gibi işlemler veritabanımızda yönetilir.

İŞ KURALLARI:

Müşteri Randevuları:

- Bir müşteri, aynı anda birden fazla randevuya sahip olabilir.
- Randevular, müşteri adına tarih ve saat bazında benzersiz olmalıdır.
- Randevu en erken 1 gün önceden oluşturulabilir ve iptali 24 saat önceye kadar yapılabilir.
- Her randevunun bir kuaför çalışanına atanması zorunludur.

Çalışan Atamaları:

- Bir çalışan aynı zaman diliminde birden fazla müşteriye hizmet veremez.
- Çalışanların uzmanlık alanları (ör. saç kesimi, renklendirme, bakım) kayıt altına alınmalıdır.
- Çalışanların çalışma saatleri tanımlanmalı ve yalnızca bu saatlerde randevu oluşturulabilmelidir.

Hizmetler:

- Her hizmet bir kategoriye (ör. Saç Kesimi, Boyama, Bakım) ait olmalıdır.
- Her müşteriye sunulan hizmet, randevu bazında kayıt edilmelidir.
- Hizmetlerin sabit bir fiyatı olabilir veya saç uzunluğu/tipine göre fiyat değişebilir.
- Fiyat politikası, sezonluk indirimlere veya kampanyalara uygun şekilde güncellenebilir.

Ödeme ve Faturalandırma:

- Randevu sonrasında hizmetlerin toplam bedeli otomatik olarak hesaplanmalıdır.
- Ödemeler nakit, kredi kartı veya dijital cüzdan aracılığıyla yapılabilir.

- Her ödeme bir faturaya bağlanmalı ve müşterinin geçmiş faturalarına erişimi sağlanmalıdır.

Ürün Satışı:

- Kuaför salonu ürün satışı yapıyorsa, ürünler stok bilgisi ile yönetilmelidir.
- Ürünler müşteri siparişleriyle ilişkilendirilmeli ve satış sonrası ödeme sürecine dahil edilmelidir.

Bağlılık ve Kampanyalar:

- Düzenli müşteriler için bağlılık puanı sistemi uygulanabilir.
- Belirli bir puan eşiğine ulaşan müşterilere indirimler sağlanabilir.
- Kampanyalar belirli tarihlerde veya hizmet türlerine göre geçerli olabilir.

İşletme Yönetimi:

- Çalışan performansı, müşterilerden alınan geri bildirimlerle değerlendirilebilir.
- Günlük, haftalık ve aylık gelir/iş yükü raporları oluşturulmalıdır.
- Çalışanların izin günleri ve molaları dikkate alınarak sistemde randevu düzenlemeleri yapılmalıdır.

Randevu Durumları:

- Randevuların durumları ("Planlandı", "Tamamlandı", "İptal Edildi", "Geç Kaçırıldı") olarak izlenmelidir.
- "Geç Kaçırıldı" durumundaki randevular, müşteri sadakat puanlarını etkileyebilir.
- İptal edilen randevular, çalışanlara bilgilendirme olarak iletilmelidir.

Çalışan Eğitimleri:

- Çalışanların belirli hizmetleri sunabilmeleri için eğitim almış olmaları gerekir.
- Her çalışanın sertifikaları ve uzmanlık seviyeleri sistemde takip edilmelidir.
- Yeni hizmet türleri eklenmeden önce çalışanlara eğitim programları tanımlanabilir.

Stok ve Malzeme Yönetimi:

- Her hizmet, belirli ürün veya malzemeleri gerektirebilir (ör. saç boyası, şampuan, tarak).
- Malzemeler stoktan otomatik olarak düşürülmeli ve kritik stok seviyelerine ulaşıldığında uyarılar tetiklenmelidir.
- Tedarikçiler ve sipariş kayıtları izlenmelidir.

Müşteri Geri Bildirimleri:

- Hizmet sonrası, müşterilerden geri bildirim toplanmalı ve hizmet kalitesi izlenmelidir.

- Geri bildirimler çalışan performansına etkileyebilir ve raporlanmalıdır.
- Negatif geri bildirimlere otomatik olarak çözüm süreci başlatılabilir (ör. indirim teklifleri).

Kuaför Salonunun Fiziksel Kaynakları:

- Her salonun sandalye, ekipman (ör. saç kurutma makinesi, koltuk), oda gibi fiziksel kaynakları takip edilmelidir.
- Randevular fiziksel kaynaklara göre planlanmalıdır; kaynak çakışmaları engellenmelidir.

Çoklu Şube Yönetimi:

- Birden fazla şubesi olan işletmelerde, her şubenin çalışanları ve hizmet türleri ayrı olarak yönetilmelidir.
- Müşteriler, farklı şubelerden hizmet alabilir ve şube bilgileri randevu kayıtlarında görünmelidir.
- Şube bazında gelir ve performans raporları oluşturulmalıdır.

Özel Randevular:

- VIP müşteriler için özel randevu saatleri tanımlanabilir.
- Özel hizmetler daha yüksek fiyatlandırılabilir ve çalışanların uygunluğu bu tür randevular için önceliklendirilebilir.

Çalışanların İş Yükü Dengesi:

- Çalışanların günlük/haftalık iş yükü dengeli bir şekilde dağıtılmalıdır.
- Fazla iş yükü olan çalışanlara randevu ataması yapılmamalıdır.
- Çalışanların izin veya mola sürelerine uyum sağlanmalıdır.

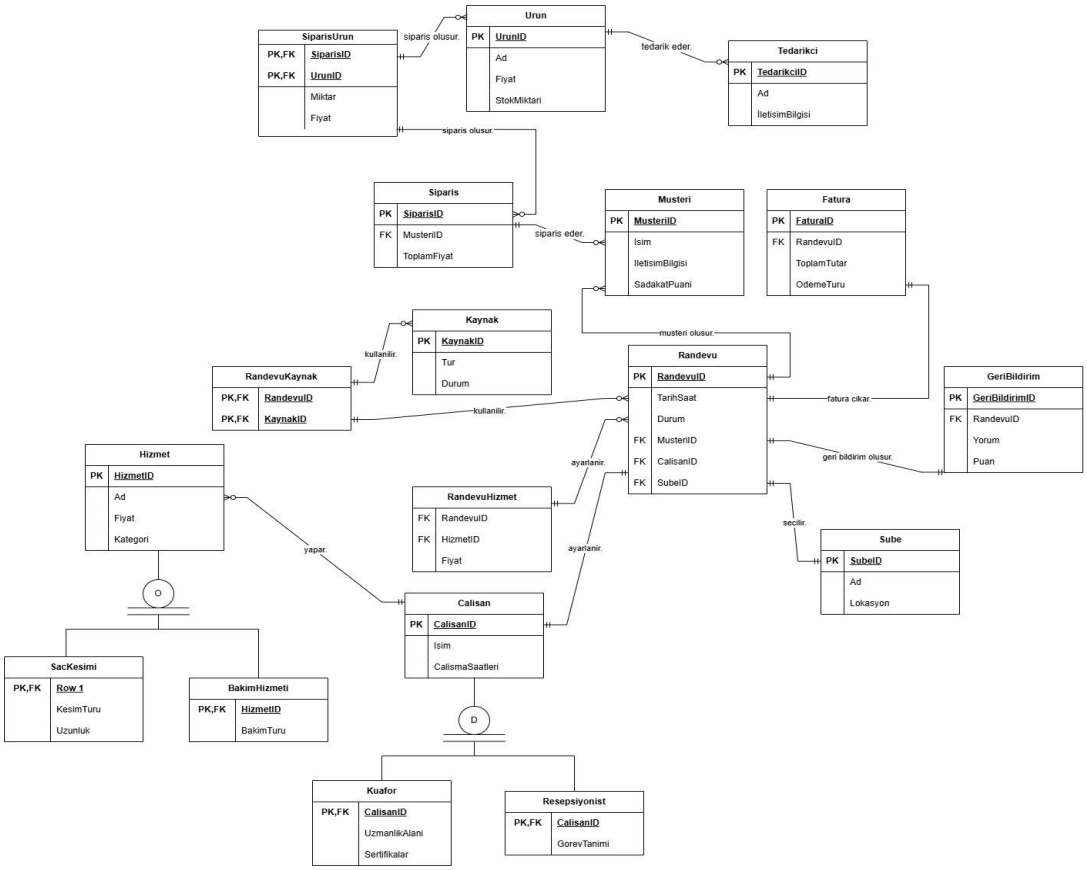
Acil Randevular:

- Acil durumlar için belirli saat aralıklarında "acil randevu" slotları ayrılabilir.
- Acil randevular için ek ücret uygulanabilir ve sadece uygun çalışanlar atanabilir.

İlişkisel Şema:

1. Urun (UrunID: integer, ad: character varying, fiyat: numeric, stokMiktari: integer)
2. Tedarikci (tedarikciID: integer, ad: character varying, iletisimBilgisi: character varying)
3. Siparis (siparisID: integer, musteriID: integer, toplamFiyat: numeric)
4. SiparisUrun(siparisID: integer, urunID: integer, miktar: integer, fiyat: numeric)
5. Musteri (musteriID: integer, isim: character varying, iletisimBilgisi: character varying, sadakatPuani: integer)

6. Fatura (faturaID: integer, randevuID: integer, toplamTutar: numeric, odemeTuru: character varying)
7. Randevu (randevuID: integer, tarihSaat: timestamp, durum: character varying, musteriID: integer, calisanID: integer, subeID: integer)
8. Sube (subeID: integer , ad: character varying, lokasyon: character varying)
9. Calisan (calisanID: integer, isim: character varying, calismaSaatleri: character varying)
10. Hizmet (hizmetID: integer, ad: character varying, fiyat: numeric, kategori: character varying
11. RandevuHizmet (randevuID: integer, hizmetID: integer, fiyat: numeric)
12. Kaynak (kaynakID: integer (PK), tur: character varying, durum: character varying)
13. RandevuKaynak (randevuID: integer (PK, FK),kaynakID: integer)
14. SacKesimi(rowno: integer, kesimTuru: character varying, uzunluk: character varying)
15. BakimHizmeti(hizmetID: integer, bakimTuru: character varying)
16. Resepsiyonist(calisanID: integer, gorevTanimi: character varying)
17. Kuafor(calisanID: integer (PK, FK), uzmanlikAlani: character varying, sertifikalar: character varying)
18. Bildirim (bildirimID: integer, randevuID: integer, yorum: character varying, puan: integer)



TÜM SQL KOMUTLARI

---TABLOLAR CREATE TABLE Urun (UrunID SERIAL PRIMARY KEY, Ad VARCHAR(100) NOT NULL, Fiyat NUMERIC(10, 2) NOT NULL, StokMiktari INT NOT NULL);

CREATE TABLE Tedarikci (TedarikciID SERIAL PRIMARY KEY, Ad VARCHAR(100) NOT NULL, IletisimBilgisi VARCHAR(255));

CREATE TABLE Siparis (SiparisID SERIAL PRIMARY KEY, MusteriID INT NOT NULL, ToplamFiyat NUMERIC(10, 2) NOT NULL, FOREIGN KEY (MusteriID) REFERENCES Musteri(MusteriID));

CREATE TABLE Musteri (MusteriID SERIAL PRIMARY KEY, MusteriAd VARCHAR(100) NOT NULL, IletisimBilgisi VARCHAR(255), SadakatPuani INT);

CREATE TABLE Fatura (FaturaID SERIAL PRIMARY KEY, RandevuID INT NOT NULL, ToplamTutar NUMERIC(10, 2) NOT NULL, OdemeTuru VARCHAR(50), FOREIGN KEY (RandevuID) REFERENCES Randevu(RandevuID));

CREATE TABLE Randevu (RandevuID SERIAL PRIMARY KEY, TarihSaat TIMESTAMP NOT NULL, Durum VARCHAR(50), MusteriAd VARCHAR(100) NOT NULL, CalisanAd VARCHAR(100) NOT NULL, HizmetAd VARCHAR(100) NOT NULL, MusteriID INT NOT NULL, CalisanID INT NOT NULL, SubeID INT NOT NULL, HizmetID INT NOT NULL, FOREIGN KEY (HizmetID) REFERENCES Hizmet(HizmetID), FOREIGN KEY (MusteriID) REFERENCES Musteri(MusteriID), FOREIGN KEY (CalisanID) REFERENCES Calisan(CalisanID), FOREIGN KEY (SubeID) REFERENCES Sube(SubeID));

CREATE TABLE Sube (SubeID SERIAL PRIMARY KEY, SubeAd VARCHAR(100) NOT NULL, Lokasyon VARCHAR(255));

CREATE TABLE Calisan (CalisanID SERIAL PRIMARY KEY, CalisanAd VARCHAR(100) NOT NULL, CalismaSaatleri VARCHAR(255));

CREATE TABLE Hizmet (HizmetID SERIAL PRIMARY KEY, HizmetAd VARCHAR(100) NOT NULL, Fiyat NUMERIC(10, 2) NOT NULL, Kategori VARCHAR(50));

CREATE TABLE RandevuHizmet (RandevuID INT NOT NULL, HizmetID INT NOT NULL, Fiyat NUMERIC(10, 2), PRIMARY KEY (RandevuID, HizmetID), FOREIGN KEY (RandevuID) REFERENCES Randevu(RandevuID), FOREIGN KEY (HizmetID) REFERENCES Hizmet(HizmetID));

CREATE TABLE SacKesimi (Rowno SERIAL PRIMARY KEY, KesimTuru VARCHAR(50) NOT NULL, Uzunluk VARCHAR(50) NOT NULL);

CREATE TABLE BakimHizmeti (HizmetID INT PRIMARY KEY, BakimTuru VARCHAR(50), FOREIGN KEY (HizmetID) REFERENCES Hizmet(HizmetID));

CREATE TABLE Resepsiyonist (CalisanID INT PRIMARY KEY, GorevTanimi VARCHAR(255), FOREIGN KEY (CalisanID) REFERENCES Calisan(CalisanID));

CREATE TABLE Kuafor (CalisanID INT PRIMARY KEY, UzmanlikAlani VARCHAR(255), Sertifikalar TEXT, FOREIGN KEY (CalisanID) REFERENCES Calisan(CalisanID));

CREATE TABLE Bildirim (BildirimID SERIAL PRIMARY KEY, RandevuID INT NOT NULL, Yorum TEXT, Puan INT, FOREIGN KEY (RandevuID) REFERENCES Randevu(RandevuID));

----FONKSİYONLAR CREATE OR REPLACE FUNCTION RandevuUcretiHesapla(RandevuID INT) RETURNS DECIMAL(10, 2) AS $$ DECLARE toplam DECIMAL(10, 2); BEGIN SELECT SUM(Fiyat) INTO toplam FROM RandevuHizmet WHERE RandevuID = RandevuID; RETURN toplam; END; $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION SadakatPuaniGuncelle(MusteriID INT, Puan INT) RETURNS VOID AS $$ BEGIN UPDATE Musteri SET SadakatPuani = SadakatPuani + Puan WHERE MusteriID = MusteriID; END; $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION KritikStokKontrolu(Seviye INT) RETURNS TABLE(UrunID INT, Ad VARCHAR, StokMiktari INT) AS $$ BEGIN RETURN QUERY SELECT UrunID, Ad, StokMiktari FROM Urun WHERE StokMiktari < Seviye; END; $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION CalisanIsYuku(Tarih DATE) RETURNS TABLE(CalisanID INT, RandevuSayisi INT) AS $$ BEGIN RETURN QUERY SELECT CalisanID, COUNT(*) FROM Randevu WHERE DATE(TarihSaat) = Tarih GROUP BY CalisanID; END; $$ LANGUAGE plpgsql;

---TETİKLEYİCİLER CREATE OR REPLACE FUNCTION StokUyarisi() RETURNS TRIGGER AS $$ BEGIN IF NEW.StokMiktari < 10 THEN RAISE NOTICE 'Ürün % stok kritik seviyeye ulaştı!', NEW."Ad"; END IF; RETURN NEW; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER StokGuncellemeUyarisi AFTER UPDATE ON Urun FOR EACH ROW EXECUTE FUNCTION StokUyarisi();

CREATE OR REPLACE FUNCTION FaturaOlustur() RETURNS TRIGGER AS $$ BEGIN IF NEW.Durum = 'Tamamlandı' THEN INSERT INTO Fatura (RandevuID, ToplamTutar, OdemeTuru) VALUES (NEW.RandevuID, RandevuUcretiHesapla(NEW.RandevuID), 'Belirsiz'); END IF; RETURN NEW; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER RandevuTamamlandigindaFatura AFTER UPDATE ON Randevu FOR EACH ROW EXECUTE FUNCTION FaturaOlustur();

CREATE OR REPLACE FUNCTION SadakatPuanEkle() RETURNS TRIGGER AS $$ BEGIN PERFORM SadakatPuaniGuncelle(NEW.MusteriID, NEW.Puan); RETURN NEW; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER GeriBildirimSadakat AFTER INSERT ON GeriBildirim FOR EACH ROW EXECUTE FUNCTION SadakatPuanEkle();

```
CREATE OR REPLACE FUNCTION RandevuCakismaKontrolu() RETURNS TRIGGER AS $$ BEGIN IF EXISTS ( SELECT 1 FROM Randevu WHERE CalisanID = NEW.CalisanID AND TarihSaat = NEW.TarihSaat ) THEN RAISE EXCEPTION 'Aynı zaman diliminde birden fazla randevu atanamaz.'; END IF; RETURN NEW; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER RandevuCakisma BEFORE INSERT ON Randevu FOR EACH ROW EXECUTE FUNCTION RandevuCakismaKontrolu();
```

FONKSİYONLAR & TETİKLEYİCİLER

Fonksiyonlar:

RandevuUcretiHesapla(RandevuID INT): Bu fonksiyonun amacı, bir randevu sırasında alınan hizmetlerin toplam maliyetini hesaplamaktır. Örneğin, bir müşteri birden fazla hizmet almışsa (saç kesimi, bakım hizmeti gibi), bu hizmetlerin fiyatları toplanarak toplam tutar elde edilir. Fonksiyon, belirli bir randevunun kimliğini (RandevuID) parametre olarak alır ve bu kimlik üzerinden hizmet bilgilerini sorgular.

SadakatPuaniGuncelle(MusteriID INT, Puan INT): Bu fonksiyonun amacı, belirli bir müşterinin sadakat puanını güncellemektir. Örneğin, bir müşteri hizmet aldığında veya olumlu bir geri bildirim bıraktığında, sistem müşterinin sadakat puanını artırmak için bu fonksiyonu kullanabilir.

KritikStokKontrolu(Seviye INT): Bu fonksiyonun amacı, stok miktarı belirli bir seviyenin altına düşen ürünleri tespit etmek ve bir liste olarak döndürmektir. Stok yönetimi açısından önemli bir işlevdir, çünkü işletmelerin kritik seviyeye ulaşan ürünlerini hızlıca belirleyip yenilemelerine olanak tanır.

CalisanIsYuku(Tarih DATE): Bu fonksiyonun amacı, bir tarihe göre çalışanların iş yükünü ölçmek ve her çalışanın o tarihte kaç randevusu olduğunu bir tablo halinde döndürmektir. Çalışanların iş yoğunluğunu analiz etmek ve çalışma planlarını düzenlemek için faydalı bir işlevdir.

Tetikleyiciler:

StokUyarısı(): Bir ürünün stok miktarı güncellendiğinde çalışır. Yeni stok miktarı kritik seviyenin (örneğin, 10 birimin) altına düşerse bir uyarı mesajı üretir. Amacı, stok seviyelerinin kritik seviyeye ulaşmadan önce tespit edilmesi ve gerekli aksiyonların alınmasını sağlamaktır.

FaturaOlustur(): Bir randevunun durumu "Tamamlandı" olarak değiştirildiğinde otomatik olarak çalışır. İlgili randevu için bir fatura kaydı oluşturur ve fatura tutarını hesaplar. Bu işlem, manuel fatura oluşturma gerekliliğini ortadan kaldırır.

SadakatPuanEkle(): Yeni bir geri bildirim eklendiğinde çalışır. Geri bildirimde verilen puanı müşterinin sadakat puanlarına ekler. Bu mekanizma, müşterilerin olumlu geri bildirim bırakmalarını teşvik etmeye yardımcı olur.

RandevuCakismaKontrolu(): Yeni bir randevu eklenmeden önce çalışır. Aynı çalışana aynı tarih ve saat için birden fazla randevu atanmasını engeller. Bu, çalışanların iş yükünü düzenlemeye ve çakışmaları önlemeye yardımcı olur.

UYGULAMA İŞLEMLERİ

Ekleme İşlemi

```
1 reference
private void BtnEkle_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        var cmd = new NpgsqlCommand(@"
    INSERT INTO Randevu (TarihSaat, Durum, MusteriAd, CalisanAd, HizmetAd, MusteriID, CalisanID, SubeID, HizmetID)
    VALUES (@tarih, 'Beklemede', @musteriAd, @calisanAd, @hizmetAd, @musteriId, @calisanId, @subeId, @hizmetId)", conn);

        cmd.Parameters.AddWithValue("@tarih", dtpTarihSaat.Value);
        cmd.Parameters.AddWithValue("@musteriAd", NpgsqlTypes.NpgsqlDbType.Varchar).Value = cmbMusteriAd.Text;
        cmd.Parameters.AddWithValue("@calisanAd", NpgsqlTypes.NpgsqlDbType.Varchar).Value = cmbCalisanAd.Text;
        cmd.Parameters.AddWithValue("@hizmetAd", NpgsqlTypes.NpgsqlDbType.Varchar).Value = cmbIslem.Text;
        cmd.Parameters.AddWithValue("@musteriId", NpgsqlTypes.NpgsqlDbType.Integer).Value = cmbMusteriAd.SelectedValue;
        cmd.Parameters.AddWithValue("@calisanId", NpgsqlTypes.NpgsqlDbType.Integer).Value = cmbCalisanAd.SelectedValue;
        cmd.Parameters.AddWithValue("@subeId", NpgsqlTypes.NpgsqlDbType.Integer).Value = cmbSubeAd.SelectedValue;
        cmd.Parameters.AddWithValue("@hizmetId", NpgsqlTypes.NpgsqlDbType.Integer).Value = cmbIslem.SelectedValue;

        cmd.ExecuteNonQuery();
        MessageBox.Show("Randevu başarıyla eklendi.");
        GuncelleDataGridView();
    }
}
```

Arama İşlemi

```
1 reference
private void BtnAra_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        var cmd = new NpgsqlCommand("SELECT * FROM Randevu WHERE MusteriAd = @musteri", conn);
        cmd.Parameters.AddWithValue("@musteri", NpgsqlTypes.NpgsqlDbType.Varchar).Value = cmbMusteriAd.Text;
        var adapter = new NpgsqlDataAdapter(cmd);
        var table = new DataTable();
        adapter.Fill(table);

        dgvRandevular.DataSource = table;
        MessageBox.Show("Arama işlemi tamamlandı.");
    }
}
```

Güncelleme İşlemi

```
3 references
private void GuncelleDataGridView()
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        var cmd = new NpgsqlCommand(@"
    SELECT R.TarihSaat, R.Durum, M.MusteriAd, C.CalisanAd, R.SubeID, H.HizmetAd
    FROM Randevu R
    JOIN Musteri M ON R.MusteriID = M.MusteriID
    JOIN Calisan C ON R.CalisanID = C.CalisanID
    JOIN Hizmet H ON R.HizmetID = H.HizmetID", conn);

        var adapter = new NpgsqlDataAdapter(cmd);
        var table = new DataTable();
        adapter.Fill(table);

        dgvRandevular.DataSource = table;
    }
}
```

Silme İşlemi

```
1 reference
private void BtnSil_Click(object sender, EventArgs e)
{
    using (var conn = new NpgsqlConnection(connectionString))
    {
        conn.Open();
        var cmd = new NpgsqlCommand("DELETE FROM Randevu WHERE MusteriAd = @musteri", conn);
        cmd.Parameters.AddWithValue("@musteri", NpgsqlTypes.NpgsqlDbType.Varchar).Value = cmbMusteriAd.Text;

        cmd.ExecuteNonQuery();
        MessageBox.Show("Seçilen randevu başarıyla silindi.");
        GuncelleDataGridView();
    }
}

3 references
```