

## OMÜ VETERİNER OTOMASYONU KAYNAK KODLARI VE DETAYLAR

**RIDVAN YAĞLIDERE 01-04-2025**

arka plana koyulan resmi düzenleme

Arka planda yer alan resmi düzenlemek için, **Form1** üzerinde gerekli adımları takip ederek resmi doğru şekilde konumlandırabiliriz. Aşağıda açıklamalarla birlikte adım adım nasıl yapılacağına dair bilgiler bulabilirsiniz:

### 1. Form'u Seç (Form1)

İlk olarak **Form1**'i seçin.

### 2. Properties (Özellikler) Penceresini Açın

Sağ tarafta yer alan **Properties** (Özellikler) penceresini açın.

### 3. BackgroundImageLayout Özelliğini Düzenleyin

**BackgroundImageLayout** özelliği, arka plan resminin nasıl yerleştirileceğini belirler.

**Tile:** Resim, formun arka planını döşeyecek şekilde tekrarlanır.

**Stretch:** Resim, formun boyutuna göre genişletilir. (Bu seçenek tavsiye edilir çünkü formun boyutuna uyum sağlar.)

**Center:** Resim, formun ortasına yerleştirilir ve tekrarlanmaz.

**Zoom:** Resim, en-boy oranını koruyarak formun içine sığdırılır.

### 4. İlgili Özelliği Seçin

**BackgroundImageLayout** özelliğini aşağıdaki seçeneklerden biriyle değiştirebilirsiniz:

**Stretch:** Resmi formun boyutuna göre genişletir. (Tavsiye edilir.)

**Center:** Resmi ortalar ve tekrar döşemez.

**Zoom:** Resmi en-boy oranını koruyarak ekrana sığdırır.

Aşağıdaki tablo, seçeneklerin açıklamalarını içermektedir:

Seçenek	Açıklama
<b>Stretch</b>	Resmi formun boyutuna göre genişletir. (Tavsiye edilir)
<b>Center</b>	Resmi ortalar ve tekrar döşemez.
<b>Zoom</b>	Resmi en-boy oranını koruyarak ekrana sığdırır.
<b>Tile</b>	Resmi formun arka planını döşeyecek şekilde tekrarlatır.

size :yana doğru genişlik

font :yazı boyutu

Application.Exit();//uygulamayı kapatır

text align: text içinde içindeki yazıyı sağ-sol-yukarı-aşağı hizalamaya yarar

Form2 form2 = new Form2(); form2.ShowDialog(); 2.sayfaya yönlendirir

---

Severity Code Description Project File Line Suppression State

Error (active) CS1069 The type name 'SqlConnection' could not be found in the namespace 'System.Data.SqlClient'. This type has been forwarded to assembly 'System.Data.SqlClient, Version=0.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a' Consider adding a reference to that assembly. WinFormsApp8  
C:\Users\ridva\source\repos\WinFormsApp8\WinFormsApp8\Form2.cs 21

---

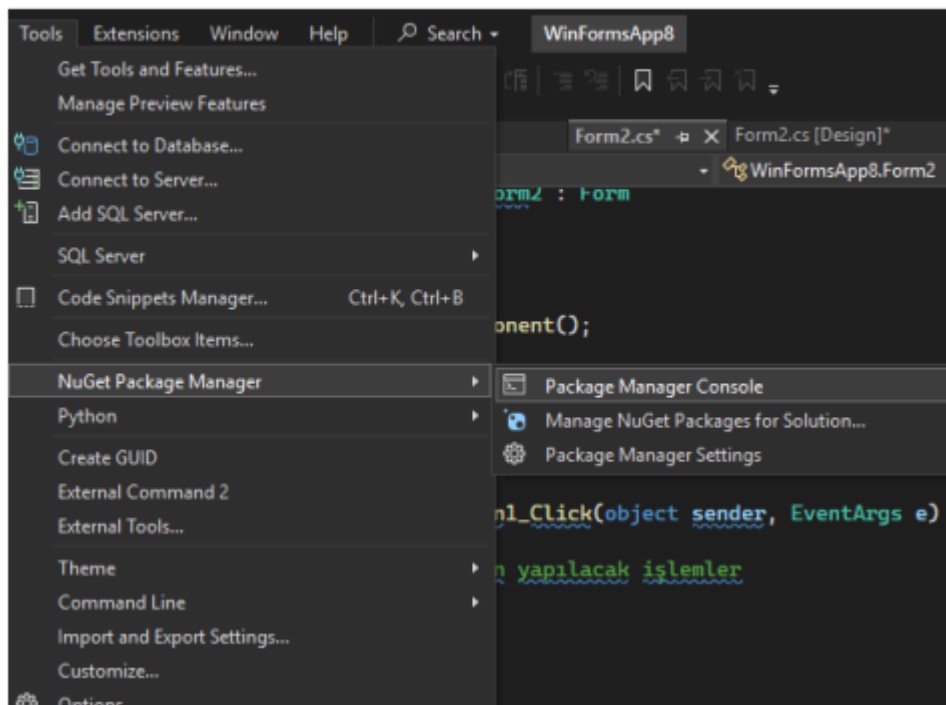
Severity Code Description Project File Line Suppression State

Error (active) CS1069 The type name 'SqlConnection' could not be found in the namespace 'System.Data.SqlClient'. This type has been forwarded to assembly 'System.Data.SqlClient, Version=0.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a' Consider adding a reference to that assembly. WinFormsApp8  
C:\Users\ridva\source\repos\WinFormsApp8\WinFormsApp8\Form2.cs 21

çözüm: using System.Data.SqlClient;

çözümün devamı:

.

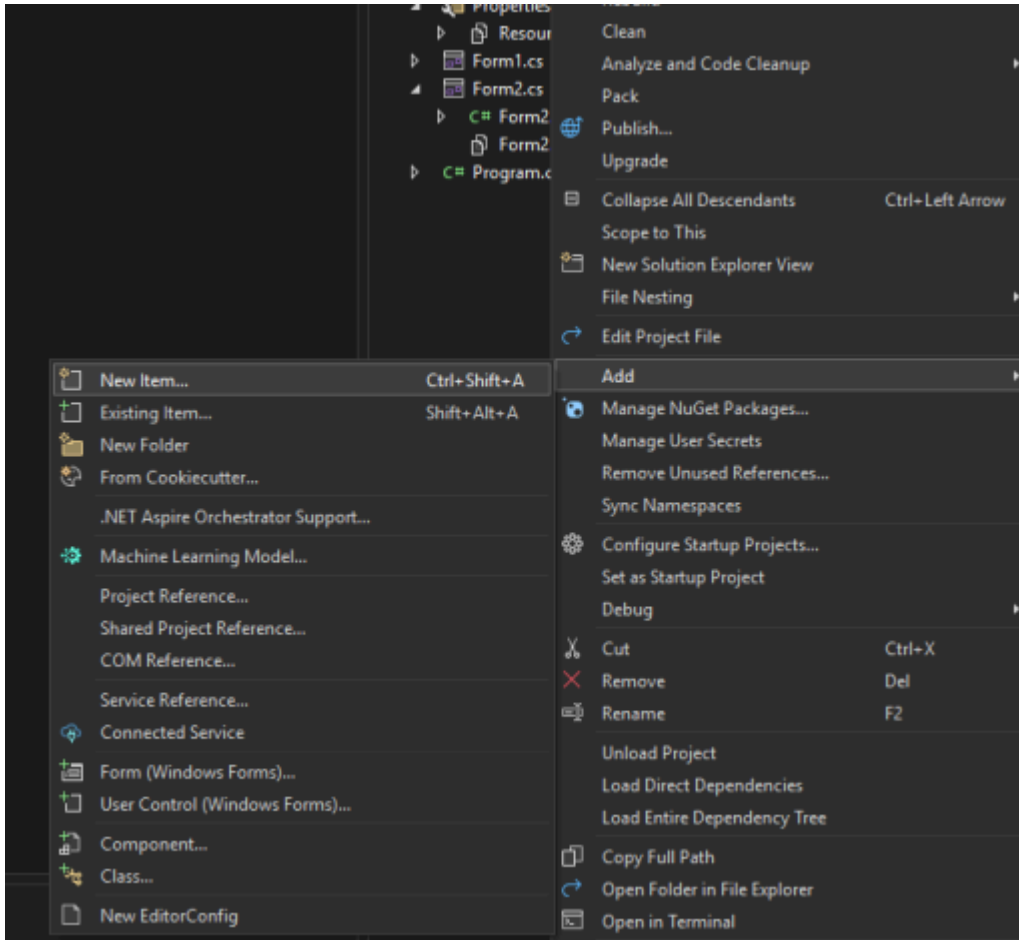


açılan konsola :Install-Package System.Data.SqlClient

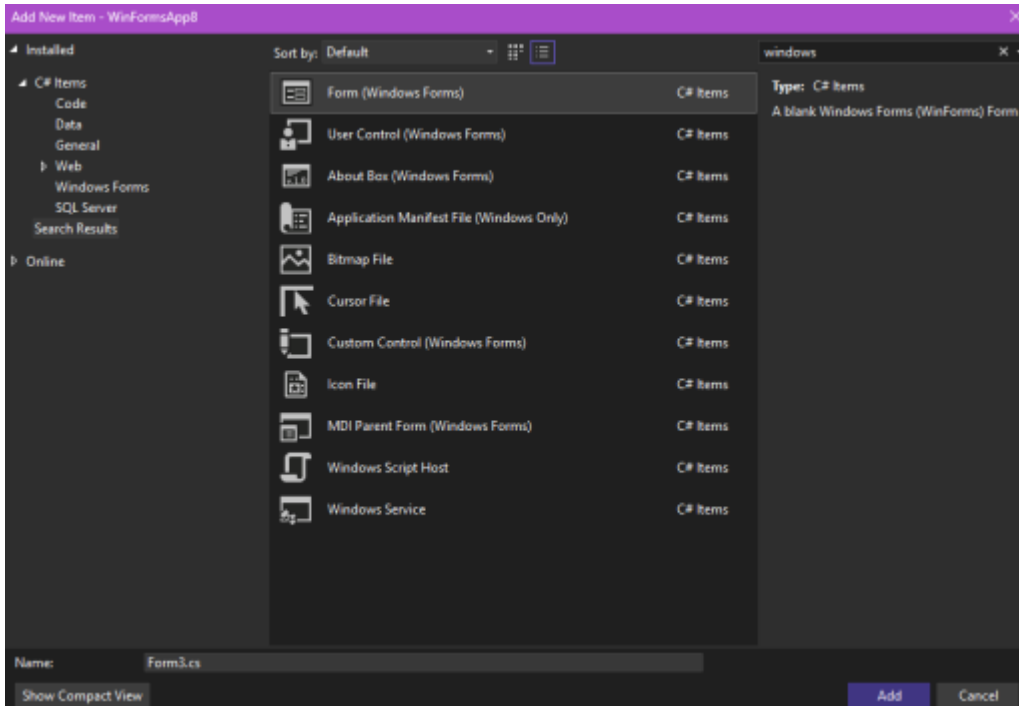
---

proje yeni form sayfası eklemek için adımlar:

1.adım



2.adım



uygulama boyutunu sabitleme:

FormBorderStyle FixedToolWindow

küpe no unicorn yapıldı çünkü benzersiz olsun diye

PlaceholderText varsa ürün no:

işlevi:

varsa ürün no:

## FORMLAR

form1 anasayfa:



c# kodları:

```
using Org.BouncyCastle.Pqc.Crypto.Lms;  
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;
```

```

using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using WinFormsApp8.Classes;

namespace WinFormsApp8
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form6 form6 = Application.OpenForms[nameof(Form6)] as Form6;
            if (form6 != null)
            {
                form6.Show();
            }
            else
            {
                new Form6().Show();
            }
        }

        private void pictureBox5_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form15 form15 = Application.OpenForms[nameof(Form15)] as Form15;
            if (form15 != null)
            {
                form15.Show(); // Eğer Form13 zaten açıksa, onu göster
            }
            else
            {
                new Form15().Show(); // Eğer Form13 yoksa, yeni bir Form13 oluştur ve göster
            }
        }

        private void pictureBox6_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form16 form16 = Application.OpenForms[nameof(Form16)] as Form16;
            if (form16 != null) form16.Show();
            else new Form16().Show();
        }
    }
}

```

```
}
```

```
private void pictureBox7_Click(object sender, EventArgs e)
{
    this.Hide();
    Form5 form5 = Application.OpenForms[nameof(Form5)] as Form5;
    if (form5 != null) form5.Show();
    else new Form5().Show();
}
```

```
private void pictureBox2_Click(object sender, EventArgs e)
{
    this.Hide();
    Form8 form8 = Application.OpenForms[nameof(Form8)] as Form8;
    if (form8 != null) form8.Show();
    else new Form8().Show();
}
```

```
private void pictureBox8_Click(object sender, EventArgs e)
{
    this.Hide();
    Form4 form4 = Application.OpenForms[nameof(Form4)] as Form4;
    if (form4 != null) form4.Show();
    else new Form4().Show();
}
```

```
private void button11_Click(object sender, EventArgs e)
{
}
}
```

```
private void pictureBox3_Click(object sender, EventArgs e)
{
    this.Hide();
    Form7 form7 = Application.OpenForms[nameof(Form7)] as Form7;
    if (form7 != null) form7.Show();
    else new Form7().Show();
}
```

```
private void pictureBox4_Click(object sender, EventArgs e)
{
    Form9 form = new Form9();
    form.Show();
    this.Hide();
}
```

```
private void pictureBox9_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form2 = Application.OpenForms[nameof(Form2)] as Form2;
```

```

        if (form2 != null) form2.Show();
        else new Form2().Show();
    }

    private void pictureBox10_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form3 form3 = Application.OpenForms[nameof(Form3)] as Form3;
        if (form3 != null) form3.Show();
        else new Form3().Show();
    }
}
}

```

---

FORM2 KAYIT ET EKRANI:

kodları :

```

using System;
using System.Data.SqlClient;
using System.Linq;
using System.Windows.Forms;

namespace WinFormsApp8
{
    public partial class Form2 : Form
    {
        // SQL Bağlantı Dizesi
        static string constring =
"Server=RIDVAN;Database=veteriner;Trusted_Connection=True;";
        SqlConnection baglanti = new SqlConnection(constring);

        public Form2()
        {
            InitializeComponent();
        }
    }
}

```

```

// Kayıt Ekleme Butonu
private void button1_Click(object sender, EventArgs e)
{
    string telefon = textBox2.Text;
    string cinsi = textBox4.Text;
    string adres = richTextBox1.Text;
    string gebelikDurumu = "Bilinmiyor"; // Gebelik durumu sabit bir değer
    olabilir ya da bir comboBox üzerinden alınabilir.

    // Veri doğrulama
    if (telefon.Length != 11)
    {
        MessageBox.Show("Telefon numarası 11 karakterden olmalıdır.");
        return;
    }

    if (cinsi.Length > 25)
    {
        MessageBox.Show("Cinsi 25 karakterden fazla olamaz.");
        return;
    }

    if (adres.Length > 255)
    {
        MessageBox.Show("Adres 255 karakterden fazla olamaz.");
        return;
    }

    // SQL Kayıt ekleme
    AddHayvanKayit(textBox1.Text, telefon, textBox3.Text, cinsi, adres,
    gebelikDurumu);
}

// SQL Kayıt Ekleme İşlemi
private void AddHayvanKayit(string hayvanSahibi, string telefon, string
kupeNo, string cinsi, string adres, string gebelikDurumu)
{
    string query = "INSERT INTO HayvanKayit (HayvanSahibi, Telefon, KupeNo,
Cinsi, Adres, GebelikDurumu) " +
"VALUES (@HayvanSahibi, @Telefon, @KupeNo, @Cinsi, @Adres,
@GebelikDurumu)";

    using (SqlConnection connection = new SqlConnection(constring))
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        command.Parameters.AddWithValue("@HayvanSahibi", hayvanSahibi);
        command.Parameters.AddWithValue("@Telefon", telefon);
        command.Parameters.AddWithValue("@KupeNo",
string.IsNullOrEmpty(kupeNo) ? (object)DBNull.Value : kupeNo);
        command.Parameters.AddWithValue("@Cinsi", cinsi);
        command.Parameters.AddWithValue("@Adres", adres);
        command.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);

        try
        {
            connection.Open();
            int result = command.ExecuteNonQuery();
            if (result > 0)
            {
                MessageBox.Show("Kayıt başarıyla eklendi!", "Bilgi",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                // TextBox ve RichTextBox temizleme
                Temizle();
            }
        }
        catch { }
    }
}

```



```

        // Form3'ü güncelle
        Form3 form3 = Application.OpenForms["Form3"] as Form3;
        if (form3 != null)
        {
            form3.VerıYenile(); // Kayıt sonrası veriyi yenile
        }
    }
    else
    {
        MessageBox.Show("Kayıt eklenemedi. Lütfen tekrar deneyin.",
            "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message, "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

// TextBox ve RichTextBox Temizleme Butonu
private void button3_Click(object sender, EventArgs e)
{
    Temizle();
    MessageBox.Show("Temizlendi", "Bilgi", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}

// TextBoxları temizleyen yardımcı metod
private void Temizle()
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
    richTextBox1.Clear();
}

// Sadece rakam girilmesine izin ver (Telefon)
private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}

// TextBox MaxLength ayarlamaları
private void textBox1_TextChanged(object sender, EventArgs e)
{
    textBox1.MaxLength = 100;
}

private void textBox2_TextChanged(object sender, EventArgs e)
{
    textBox2.MaxLength = 11;

    if (!System.Text.RegularExpressions.Regex.IsMatch(textBox2.Text,
        "[0-9]*$"))
    {
        MessageBox.Show("Sadece rakam girebilirsiniz!", "Uyarı",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

```

```

        textBox2.Text = new
string(textBox2.Text.Where(char.IsDigit).ToArray());
        textBox2.SelectionStart = textBox2.Text.Length;
    }
}

private void textBox3_TextChanged(object sender, EventArgs e)
{
    textBox3.MaxLength = 20;
}

private void textBox4_TextChanged(object sender, EventArgs e)
{
    textBox4.MaxLength = 25;
}

private void richTextBox1_TextChanged(object sender, EventArgs e)
{
    richTextBox1.MaxLength = 255;
}

// Formlar Arası Geçiş Butonu
private void button4_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form1"] as Form1;

    if (form9 != null)
    {
        form9.Show();
    }
    else
    {
        form9 = new Form1();
        form9.Show();
    }
}

private void pictureBox2_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
    richTextBox1.Clear();
}

private void pictureBox3_Click(object sender, EventArgs e)
{
    string telefon = textBox2.Text;
    string cinsi = textBox4.Text;
    string adres = richTextBox1.Text;
    string gebelikDurumu = "Bilinmiyor"; // Gebelik durumu sabit bir değer
    olabilir ya da bir comboBox üzerinden alınabilir.

    // Veri doğrulama
    if (telefon.Length != 11)
    {
        MessageBox.Show("Telefon numarası 11 karakterden olmalıdır.");
        return;
    }

    if (cinsi.Length > 25)
    {
        MessageBox.Show("Cinsi 25 karakterden fazla olamaz.");
    }
}

```

```

        return;
    }

    if (adres.Length > 255)
    {
        MessageBox.Show("Adres 255 karakterden fazla olamaz.");
        return;
    }

    // SQL Kayıt ekleme
    AddHayvanKayit(textBox1.Text, telefon, textBox3.Text, cinsi, adres,
gebelikDurumu);
}




private void pictureBox4_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form1"] as Form1;
    if (form9 != null)
    {
        form9.Show();
    }
    else
    {
        form9 = new Form1();
        form9.Show();
    }
}
}
}

```

sql:

	Table Name	Column Name	Column ID	Data Type	Max Length	Is Nullable	Index Name	Index Type	Foreign Key Name	Referenced Table	Referenced Column
1	HayvanKayit	id	1	int	4	0	PK__HayvanKa__3213E83F2DAB05C6	CLUSTERED	NULL	NULL	NULL
2	HayvanKayit	KupeNo	2	nvarchar	100	0	UQ__KupeNo	NONCLUSTERED	NULL	NULL	NULL
3	HayvanKayit	Telefon	3	nvarchar	40	0	NULL	NULL	NULL	NULL	NULL
4	HayvanKayit	HayvanSahibi	4	nvarchar	200	0	NULL	NULL	NULL	NULL	NULL
5	HayvanKayit	Cinsi	5	nvarchar	100	0	NULL	NULL	NULL	NULL	NULL
6	HayvanKayit	Adres	6	nvarchar	400	0	NULL	NULL	NULL	NULL	NULL
7	HayvanKayit	GebelikDurumu	7	nvarchar	40	0	NULL	NULL	NULL	NULL	NULL
8	HayvanKayit	Kayit Tarihi	8	datetime	8	0	NULL	NULL	NULL	NULL	NULL
		6	Adres	nvarchar	200				NO		
		7	GebelikDurumu	nvarchar	20				NO		
		8	Kayit Tarihi	datetime	NULL				NO		

FORM3 KAYITLARIM

Kayıtlarım




güncelle
çıkış
AnaSayfa

	HayvanID	HayvanSahibi	Telefon	KupeNo	Cinsi	Adres	GebelikDurumu
»							

#### C# KODLARI:

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace WinFormsApp8
{
    public partial class Form3 : Form
    {
        private DataTable tablo = new DataTable();
        private readonly string connectionString =
"Server=RIDVAN;Database=veteriner;Trusted_Connection=True;";

        public Form3()
        {
            InitializeComponent();
            VeriYenile();
        }

        // Veritabanından verileri çekip DataGridView1'e yükler
        public void VeriYenile()
        {
            try
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string query = "SELECT HayvanID, HayvanSahibi, Telefon, KupeNo,
Cinsi, Adres, GebelikDurumu FROM HayvanKayit";
                    using (SqlDataAdapter adapter = new SqlDataAdapter(query,
connection))
                    {
                        tablo.Clear(); // Önceki verileri temizler
                        adapter.Fill(tablo); // Yeni verilerle tabloyu doldurur
                        dataGridView1.DataSource = tablo; // DataGridView1'e tabloyu
bağlar
                    }
                }
            }
        }
    }
}

```

```

    }
    catch (Exception ex)
    {
        MessageBox.Show($"Veri yenileme hatası: {ex.Message}", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

// DataGridView1'de satır silme işlemi (Delete tuşu ile)
private void DataGridView1_UserDeletingRow(object sender,
DataGridViewRowCancelEventArgs e)
{
    try
    {
        if (e.Row.DataBoundItem == null) return;

        DialogResult result = MessageBox.Show("Bu satırı silmek istediğinize
emin misiniz?", "Onay", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (result == DialogResult.No)
        {
            e.Cancel = true; // Silme işlemini iptal eder
        }
        else
        {
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                connection.Open();
                DataRow row = ((DataRowView)e.Row.DataBoundItem).Row;
                string query = "DELETE FROM HayvanKayıt WHERE HayvanID =
@HayvanID";

                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@HayvanID",
row["HayvanID"]);

                    command.ExecuteNonQuery(); // Veritabanından satırı siler
                }

                // Yerel tabloda da satırı sil
                DataRowView drv = e.Row.DataBoundItem as DataRowView;
                drv.Row.Delete();
                tablo.AcceptChanges(); // Silme işlemini yerel tabloda onaylar
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Silme hatası: {ex.Message}", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

// PictureBox1 tıklandığında güncelleme işlemini gerçekleştirir ve tabloyu
kaydeder
private void pictureBox1_Click(object sender, EventArgs e)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            DataTable changes = tablo.GetChanges(); // Değişiklikleri kontrol
            eder

            if (changes == null)
            {

```

```

        MessageBox.Show("Herhangi bir değişiklik yapılmadı!", "Bilgi",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }

    // Güncelleme işlemi
    using (SqlDataAdapter adapter = new SqlDataAdapter("SELECT
HayvanID, HayvanSahibi, Telefon, KupeNo, Cinsi, Adres, GebelikDurumu FROM
HayvanKayit", connection))
    {
        SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
        adapter.UpdateCommand = builder.GetUpdateCommand(); //
Otomatik UPDATE komutu oluşturur
        adapter.Update(tablo); // Değişiklikleri veritabanına kaydeder
    }

    tablo.AcceptChanges(); // Yerel tabloyu günceller
}

    MessageBox.Show("Değişiklikler başarıyla kaydedildi!", "Başarılı",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    dataGridView1.DataSource = tablo; // Güncel tabloyu DataGridView1'e
bağlar
}
catch (Exception ex)
{
    MessageBox.Show($"Güncelleme hatası: {ex.Message}", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

// Uygulamayı kapatır
private void pictureBox2_Click(object sender, EventArgs e)
{
    Application.Exit();
}

// Birinci forma geçiş yapar
private void pictureBox4_Click(object sender, EventArgs e)
{
    try
    {
        this.Hide();
        Form1 form1 = Application.OpenForms["Form1"] as Form1 ?? new Form1();
        form1.Show();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Form geçiş hatası: {ex.Message}", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
}
//form2 deki bilgileri gördüğümüz ekran

```

---

#### FORM4 STOK BİLGİLERİ

C# KODLARI:

```

using System;
using System.Data;
using System.Data.SqlClient;

```

```

using System.Windows.Forms;

namespace WinFormsApp8
{
    public partial class Form4 : Form
    {
        private string connectionString =
"Server=localhost;Database=veteriner;Integrated Security=True;";
        private DataTable originalDataTable; // Orjinal veriyi saklamak için

        public Form4()
        {
            InitializeComponent();
            StokListele();
            this.dataGridView1.KeyDown += new
KeyEventHandler(this.dataGridView1_KeyDown);
            this.dataGridView1.CellEndEdit += new
DataGridViewCellEventHandler(this.dataGridView1_CellEndEdit);
            this.textBox6.TextChanged += new EventHandler(this.textBox6_TextChanged);
// TextBox6 için TextChanged olayı
            this.textBox5.TextChanged += new EventHandler(this.textBox5_TextChanged);
// TextBox5 için TextChanged olayı
        }

        // Stokları DataGridView'de göster
        private void StokListele()
        {
            try
            {
                // DataGridView'in düzenleme modundan çıkmasını sağla
                if (dataGridView1.IsCurrentCellInEditMode)
                {
                    dataGridView1.EndEdit();
                }

                // Geçerli hücreyi temizle
                dataGridView1.ClearSelection();
                dataGridView1.CurrentCell = null;

                using (SqlConnection con = new SqlConnection(connectionString))
                {
                    con.Open();
                    string query = "SELECT id, urun_adi, urun_no, adet, alis_fiyati,
satis_fiyati, kazanc FROM stok";
                    SqlDataAdapter da = new SqlDataAdapter(query, con);
                    originalDataTable = new DataTable(); // Orjinal veriyi sakla
                    da.Fill(originalDataTable);

                    // Veri kaynağını güncelle
                    dataGridView1.DataSource = originalDataTable;
                    dataGridView1.Columns["kazanc"].DefaultCellStyle.Format = "C2";
                    dataGridView1.Columns["alis_fiyati"].DefaultCellStyle.Format =
"C2";
                    dataGridView1.Columns["satis_fiyati"].DefaultCellStyle.Format =
"C2";
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Hata: Stoklar listelenirken bir sorun oluştu.\n" +
ex.Message);
            }
        }

        // TextBox6'daki metne göre filtreleme (urun_adi veya urun_no)

```

```

private void textBox6_TextChanged(object sender, EventArgs e)
{
    Filtrele();
}

// TextBox5'daki metne göre filtreleme (örneğin sadece satis_fiyati'na göre)
private void textBox5_TextChanged(object sender, EventArgs e)
{
    Filtrele();
}

// Filtreleme işlemini birleştiren metot
private void Filtrele()
{
    try
    {
        // DataGridView'in düzenleme modundan çıkmasını sağla
        if (dataGridView1.IsCurrentCellInEditMode)
        {
            dataGridView1.EndEdit();
        }

        // Geçerli hücreyi temizle
        dataGridView1.ClearSelection();
        dataGridView1.CurrentCell = null;

        string filterText6 = textBox6.Text.Trim();
        string filterText5 = textBox5.Text.Trim();

        DataView dv = originalDataTable.DefaultView;
        string filterExpression = "";

        // TextBox6 için filtre (urun_adi veya urun_no)
        if (!string.IsNullOrEmpty(filterText6))
        {
            filterExpression = $"urun_adi LIKE '%{filterText6}%' OR urun_no
LIKE '%{filterText6}%'";
        }

        // TextBox5 için filtre (örneğin satis_fiyati'na göre)
        if (!string.IsNullOrEmpty(filterText5))
        {
            if (decimal.TryParse(filterText5, out decimal satisFiyati))
            {
                string satisFiyatiFilter = $"satis_fiyati = {satisFiyati}";
                filterExpression = string.IsNullOrEmpty(filterExpression)
                    ? satisFiyatiFilter
                    : $"{filterExpression} AND {satisFiyatiFilter}";
            }
            else
            {
                MessageBox.Show("Lütfen satış fiyatı için geçerli bir sayı
girin (Örnek: 75.00).");
                return;
            }
        }

        // Filtreyi uygula
        if (string.IsNullOrEmpty(filterExpression))
        {
            dv.RowFilter = string.Empty; // Filtre yoksa orijinal veriyi
göster
            dataGridView1.DataSource = originalDataTable;
        }
        else
    }
}

```



```

        {
            dv.RowFilter = filterExpression;
            dataGridView1.DataSource = dv;
        }

        // Eğer filtrelenmiş veri yoksa kullanıcıyı bilgilendir
        if (dv.Count == 0)
        {
            MessageBox.Show("Arama kriterlerine uygun veri bulunamadı.");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Hata: Filtreleme sırasında bir sorun oluştu.\n" +
ex.Message);
    }
}

// Textbox'ları temizle
private void TextboxlariTemizle()
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
    textBox5.Clear();
    textBox6.Clear(); // TextBox6'yı da temizle
}

// Delete tuşuyla seçili satırı silme
private void dataGridView1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Delete)
    {
        try
        {
            if (dataGridView1.SelectedRows.Count > 0)
            {
                int seciliSatirId =
Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["id"].Value);
                DialogResult result = MessageBox.Show("Bu ürünü silmek
istediğinize emin misiniz?", "Silme İşlemi", MessageBoxButtons.YesNo);
                if (result == DialogResult.Yes)
                {
                    using (SqlConnection con = new
SqlConnection(connectionString))
                    {
                        con.Open();
                        string deleteQuery = "DELETE FROM stok WHERE id =
@id";
                        using (SqlCommand deleteCmd = new
SqlCommand(deleteQuery, con))
                        {
                            deleteCmd.Parameters.AddWithValue("@id",
seciliSatirId);
                            deleteCmd.ExecuteNonQuery();
                        }
                    }
                    MessageBox.Show("Ürün başarıyla silindi.");
                    StokListele();
                }
            }
            else
            {
                MessageBox.Show("Lütfen silmek için bir satır seçin.");
            }
        }
        catch { }
    }
}

```

```

        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Hata: " + ex.Message);
    }
}

// Hücre düzenleme tamamlandığında veritabanını güncelle
private void dataGridView1_CellEndEdit(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        int rowIndex = e.RowIndex;
        int columnIndex = e.ColumnIndex;

        int id =
Convert.ToInt32(dataGridView1.Rows[rowIndex].Cells["id"].Value);
        string columnName = dataGridView1.Columns[columnIndex].Name;
        object newValue =
dataGridView1.Rows[rowIndex].Cells[columnIndex].Value;

        if (newValue == null ||
string.IsNullOrEmpty(newValue.ToString()))
        {
            MessageBox.Show("Lütfen geçerli bir değer girin.");
            StokListele();
            return;
        }

        if (columnName == "adet" && !int.TryParse(newValue.ToString(), out _))
        {
            MessageBox.Show("Adet için geçerli bir sayı girin.");
            StokListele();
            return;
        }
        else if ((columnName == "alis_fiyati" || columnName == "satis_fiyati"
|| columnName == "kazanc") && !decimal.TryParse(newValue.ToString(), out _))
        {
            MessageBox.Show("Fiyat veya kazanç için geçerli bir sayı girin.");
            StokListele();
            return;
        }

        using (SqlConnection con = new SqlConnection(connectionString))
        {
            con.Open();
            string updateQuery = $"UPDATE stok SET {columnName} = @newValue
WHERE id = @id";
            using (SqlCommand cmd = new SqlCommand(updateQuery, con))
            {
                cmd.Parameters.AddWithValue("@newValue", newValue);
                cmd.Parameters.AddWithValue("@id", id);
                cmd.ExecuteNonQuery();
            }

            if (columnName == "alis_fiyati" || columnName == "satis_fiyati")
            {
                string kazancQuery = "UPDATE stok SET kazanc = (satis_fiyati -
alis_fiyati) * adet WHERE id = @id";
                using (SqlCommand kazancCmd = new SqlCommand(kazancQuery,
con))
                {

```

```

        kazancCmd.Parameters.AddWithValue("@id", id);
        kazancCmd.ExecuteNonQuery();
    }
}

MessageBox.Show("Değişiklik başarıyla kaydedildi.");
StokListele();
}
catch (Exception ex)
{
    MessageBox.Show("Hata: Değişiklik kaydedilirken bir sorun oluştu.\n" +
ex.Message);
    StokListele();
}
}

// Ana forma dönme
private void pictureBox1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form1"] as Form1;
    if (form9 != null)
        form9.Show();
    else
    {
        form9 = new Form1();
        form9.Show();
    }
}

// Yeni stok ekleme veya mevcut stoğu güncelleme
private void pictureBox3_Click(object sender, EventArgs e)
{
    try
    {
        // textBox2 hariç diğer alanların dolu olması zorunlu
        if (string.IsNullOrEmpty(textBox1.Text) ||
string.IsNullOrEmpty(textBox3.Text) || string.IsNullOrEmpty(textBox4.Text)
|| string.IsNullOrEmpty(textBox5.Text))
        {
            MessageBox.Show("Lütfen ürün adı, miktar, alış fiyatı ve satış
fiyatı alanlarını doldurun.");
            return;
        }

        int miktar;
        decimal alisFiyati, satisFiyati;
        if (!int.TryParse(textBox3.Text, out miktar) ||
!decimal.TryParse(textBox4.Text, out alisFiyati) || !decimal.TryParse(textBox5.Text,
out satisFiyati))
        {
            MessageBox.Show("Lütfen geçerli bir miktar ve fiyat girin.");
            return;
        }

        using (SqlConnection con = new SqlConnection(connectionString))
        {
            con.Open();
            string checkQuery = "SELECT COUNT(*) FROM stok WHERE (urun_no =
@urunNo OR (urun_no IS NULL AND @urunNo IS NULL)) OR urun_adi = @urunAdi";
            using (SqlCommand checkCmd = new SqlCommand(checkQuery, con))
            {

```

```

        checkCmd.Parameters.AddWithValue("@urunNo",
string.IsNullOrEmpty(textBox2.Text) ? (object)DBNull.Value :
textBox2.Text.Trim());
        checkCmd.Parameters.AddWithValue("@urunAdi",
textBox1.Text.Trim());
        int urunVarMi = (int)checkCmd.ExecuteScalar();

        if (urunVarMi == 0)
        {
            string insertQuery = "INSERT INTO stok (urun_adi, urun_no,
adet, alis_fiyati, satis_fiyati, kazanc) " +
                                "VALUES (@urunAdi, @urunNo, @miktar,
@alisFiyati, @satisFiyati, 0)";
            using (SqlCommand insertCmd = new SqlCommand(insertQuery,
con))
            {
                insertCmd.Parameters.AddWithValue("@urunAdi",
textBox1.Text.Trim());
                insertCmd.Parameters.AddWithValue("@urunNo",
string.IsNullOrEmpty(textBox2.Text) ? (object)DBNull.Value :
textBox2.Text.Trim());
                insertCmd.Parameters.AddWithValue("@miktar", miktar);
                insertCmd.Parameters.AddWithValue("@alisFiyati",
alisFiyati);
                insertCmd.Parameters.AddWithValue("@satisFiyati",
satisFiyati);
                insertCmd.ExecuteNonQuery();
            }
        }
        else
        {
            string updateQuery = "UPDATE stok SET adet = adet +
@miktar WHERE (urun_no = @urunNo OR (urun_no IS NULL AND @urunNo IS NULL)) OR urun_adi
= @urunAdi";
            using (SqlCommand updateCmd = new SqlCommand(updateQuery,
con))
            {
                updateCmd.Parameters.AddWithValue("@urunNo",
string.IsNullOrEmpty(textBox2.Text) ? (object)DBNull.Value :
textBox2.Text.Trim());
                updateCmd.Parameters.AddWithValue("@urunAdi",
textBox1.Text.Trim());
                updateCmd.Parameters.AddWithValue("@miktar", miktar);
                updateCmd.ExecuteNonQuery();
            }
        }

        MessageBox.Show("Stok başarıyla güncellendi.");
        StokListele();
        TextboxlariTemizle(); // TextBox'ları temizle
    }
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message);
}
}

// Ürün satışı
private void pictureBox2_Click(object sender, EventArgs e)
{
    try
    {
        // textBox2 hariç diğer alanların dolu olması zorunlu
    }
}

```

```

        if (string.IsNullOrEmpty(textBox1.Text) ||
string.IsNullOrEmpty(textBox3.Text))
        {
            MessageBox.Show("Lütfen ürün adı ve satış miktarı alanlarını
doldurun.");
            return;
        }

        int satilacakMiktar;
        if (!int.TryParse(textBox3.Text, out satilacakMiktar))
        {
            MessageBox.Show("Lütfen geçerli bir satış miktarı girin.");
            return;
        }

        using (SqlConnection con = new SqlConnection(connectionString))
        {
            con.Open();
            string checkQuery = "SELECT adet, satis_fiyati, alis_fiyati,
kazanc FROM stok WHERE (urun_no = @urunNo OR (urun_no IS NULL AND @urunNo IS NULL)) OR
urun_adi = @urunAdi";
            using (SqlCommand checkCmd = new SqlCommand(checkQuery, con))
            {
                checkCmd.Parameters.AddWithValue("@urunNo",
string.IsNullOrEmpty(textBox2.Text) ? (object)DBNull.Value :
textBox2.Text.Trim());
                checkCmd.Parameters.AddWithValue("@urunAdi",
textBox1.Text.Trim());
                SqlDataReader reader = checkCmd.ExecuteReader();

                if (!reader.Read())
                {
                    MessageBox.Show("Ürün stokta bulunamadı! Lütfen ürün adı
veya ürün numarasını doğru girin.");
                    return;
                }

                int stokMiktari = Convert.ToInt32(reader["adet"]);
                decimal satisFiyati =
Convert.ToDecimal(reader["satis_fiyati"]);
                decimal alisFiyati = Convert.ToDecimal(reader["alis_fiyati"]);
                decimal mevcutKazanc = Convert.ToDecimal(reader["kazanc"]);
                reader.Close();

                if (stokMiktari < satilacakMiktar)
                {
                    MessageBox.Show($"Yetersiz stok! Mevcut: {stokMiktari},
Satış İsteddiğiniz: {satilacakMiktar}");
                    return;
                }

                decimal saleProfit = (satisFiyati - alisFiyati) *
satilacakMiktar;

                string updateQuery = "UPDATE stok SET adet = adet -
@satilacakMiktar, kazanc = kazanc + @saleProfit WHERE (urun_no = @urunNo OR (urun_no
IS NULL AND @urunNo IS NULL)) OR urun_adi = @urunAdi";
                using (SqlCommand updateCmd = new SqlCommand(updateQuery,
con))
                {
                    updateCmd.Parameters.AddWithValue("@satilacakMiktar",
satilacakMiktar);
                    updateCmd.Parameters.AddWithValue("@saleProfit",
saleProfit);

```

```

        updateCmd.Parameters.AddWithValue("@urunNo",
string.IsNullOrEmpty(textBox2.Text) ? (object)DBNull.Value :
textBox2.Text.Trim());
        updateCmd.Parameters.AddWithValue("@urunAdi",
textBox1.Text.Trim());
        updateCmd.ExecuteNonQuery();
    }

    MessageBox.Show($"Satış işlemi başarılı!\nBu satıştan elde
edilen kazanç: {saleProfit:C2}");
    StokListele();
    TextboxlariTemizle(); // TextBox'ları temizle
    }
}
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message);
}
}

// PictureBox5'e basınca DataGridView'deki tüm değişiklikleri kaydet
private void pictureBox5_Click(object sender, EventArgs e)
{
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            con.Open();
            bool hasError = false;

            foreach (DataGridViewRow row in dataGridView1.Rows)
            {
                if (row.IsNewRow) continue;

                int id = Convert.ToInt32(row.Cells["id"].Value);
                string urunAdi = row.Cells["urun_adi"].Value?.ToString();
                string urunNo = row.Cells["urun_no"].Value?.ToString();
                string adetStr = row.Cells["adet"].Value?.ToString();
                string alisFiyatiStr =
row.Cells["alis_fiyati"].Value?.ToString();
                string satisFiyatiStr =
row.Cells["satis_fiyati"].Value?.ToString();
                string kazancStr = row.Cells["kazanc"].Value?.ToString();

                // Doğrulama
                if (string.IsNullOrEmpty(urunAdi))
                {
                    MessageBox.Show($"Satır {row.Index + 1}: Ürün adı boş
olamaz.");
                    hasError = true;
                    continue;
                }
                if (!int.TryParse(adetStr, out int adet))
                {
                    MessageBox.Show($"Satır {row.Index + 1}: Adet için geçerli
bir sayı girin (Örnek: 10).");
                    hasError = true;
                    continue;
                }
                if (!decimal.TryParse(alisFiyatiStr, out decimal alisFiyati))
                {
                    MessageBox.Show($"Satır {row.Index + 1}: Alış fiyatı için
geçerli bir sayı girin (Örnek: 50.00).");
                    hasError = true;

```

```

        continue;
    }
    if (!decimal.TryParse(satisFiyatiStr, out decimal
satisFiyati))
    {
        MessageBox.Show($"Satır {row.Index + 1}: Satış fiyatı için
geçerli bir sayı girin (Örnek: 75.00).");
        hasError = true;
        continue;
    }
    if (!decimal.TryParse(kazancStr, out decimal kazanc))
    {
        MessageBox.Show($"Satır {row.Index + 1}: Kazanç için
geçerli bir sayı girin (Örnek: 25.00).");
        hasError = true;
        continue;
    }

    // Veritabanını güncelle
    string updateQuery = "UPDATE stok SET urun_adi = @urunAdi,
urun_no = @urunNo, adet = @adet, " +
        "alis_fiyati = @alisFiyati, satis_fiyati
= @satisFiyati, kazanc = @kazanc WHERE id = @id";
    using (SqlCommand updateCmd = new SqlCommand(updateQuery,
con))
    {
        updateCmd.Parameters.AddWithValue("@id", id);
        updateCmd.Parameters.AddWithValue("@urunAdi", urunAdi);
        updateCmd.Parameters.AddWithValue("@urunNo",
string.IsNullOrEmpty(urunNo) ? (object)DBNull.Value : urunNo);
        updateCmd.Parameters.AddWithValue("@adet", adet);
        updateCmd.Parameters.AddWithValue("@alisFiyati",
alisFiyati);
        updateCmd.Parameters.AddWithValue("@satisFiyati",
satisFiyati);
        updateCmd.Parameters.AddWithValue("@kazanc", kazanc);
        updateCmd.ExecuteNonQuery();
    }

    if (hasError)
    {
        MessageBox.Show("Bazı satırlarda hatalar bulundu. Lütfen
düzeltip tekrar deneyin.");
    }
    else
    {
        MessageBox.Show("Tüm değişiklikler başarıyla kaydedildi.");
        StokListele();
    }
}
catch (Exception ex)
{
    MessageBox.Show("Hata: Değişiklikler kaydedilirken bir sorun
oluştı.\n" + ex.Message);
}
}
}
}

```

//NOT: KAZANÇ ÜRÜNÜ SATTIKÇA ARTAR :)

SQL :

**Sütun Adı**	**Veri Tipi**	**Maksimum Uzunluk**	**Boş Olabilir**	**Varsayılan Kısıtlama**	**İndeks Adı**	**İndeks Tipi**	**Yabancı Anahtar Adı**	**Referans Tablosu**	**Referans Sütunu**
id	int	4	0		PK_stok__3213E83FFE4CA945	CLUSTERED			
urun_adi	nvarchar	510	0						
urun_no	nvarchar	100	1		UQ__stok__933C3B0D69001019	NONCLUSTERED			
adet	int	4	0						
alis_fiyati	decimal	9	0						
satis_fiyati	decimal	9	0						
kazanc	decimal	9	0						

\*\*\*\*\* Script for SelectTopNRows command from SSMS \*\*\*\*\*//ALIŞLAR



```
SELECT TOP 1000 [id]
, [urun_id]
, [adet]
, [alis_fiyati]
, [tarih]
FROM [veteriner].[dbo].[alislar]
```


\*\*\*\*\* Script for SelectTopNRows command from SSMS \*\*\*\*\*//SATIŞLAR


```
SELECT TOP 1000 [id]
, [urun_id]
, [adet]
, [satis_fiyati]
, [tarih]
FROM [veteriner].[dbo].[satislar]
```

FORM 5 VERESİYE DEFTERİ:










**Ad Soyad**

**Telefon:**

**İşlem:**

**İşlem Ücreti:**   
☐ NAKİT ÖDEME  
☐ KREDİ KARTI

Kayıt Ödeme Temizle

	ID	AdSoyad	TelefonNo	Yapılanİşlem	İşlemÜcreti	ÖdemeSekli	KayıtTarihi	Borç
▶	2019	rdvan	05353002362		100,00	KREDİ KARTI	25.03.2025 20:34	0,00
	2020	Hamdi satıroğlu	05353002363	a	1,00	KREDİ KARTI	27.03.2025 22:34	0,00
*								

C# KODLARI:

```
using System;
using System.Data;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;
```

```
namespace WinFormsApp8
```

```
{
    public partial class Form5 : Form
    {
```

```
        private string connectionString = "Server=RIDVAN;Database=veteriner;Integrated
Security=True;TrustServerCertificate=True;";
```

```
        public Form5()
        {
            InitializeComponent();
            LoadDataGrid();
            checkedListBox1.ItemCheck += CheckedListBox1_ItemCheck;
            textBox1.KeyPress += TextBox1_KeyPress;
            textBox2.KeyPress += TextBox2_KeyPress;
            textBox5.KeyUp += TextBox5_KeyUp;
        }
```

```
        private void TextBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsLetter(e.KeyChar) && !char.IsControl(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar))
            {
                e.Handled = true;
            }
        }
```

```
        private void TextBox2_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))
```

```

    {
        e.Handled = true;
    }
    if (textBox2.Text.Length >= 11 && !char.IsControl(e.KeyChar))
    {
        e.Handled = true;
    }
}

private void CheckedListBox1_ItemCheck(object sender, ItemCheckEventArgs e)
{
    if (e.NewValue == CheckState.Checked)
    {
        for (int i = 0; i < checkedListBox1.Items.Count; i++)
        {
            if (i != e.Index)
            {
                checkedListBox1.SetItemChecked(i, false);
            }
        }
    }
}

private void button1_Click(object sender, EventArgs e)
{
}

private void button2_Click(object sender, EventArgs e)
{
}

private void button3_Click(object sender, EventArgs e)
{
}

private void button5_Click(object sender, EventArgs e)
{
}

private void LoadDataGrid(string filter = "")
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();
        string query = "SELECT * FROM Islemler";
    }
}

```

```

        if (!string.IsNullOrEmpty(filter))
        {
            query += " WHERE AdSoyad LIKE @Filter OR TelefonNo LIKE @Filter";
        }

        using (SqlCommand cmd = new SqlCommand(query, conn))
        {
            if (!string.IsNullOrEmpty(filter))
            {
                cmd.Parameters.AddWithValue("@Filter", "%" + filter + "%");
            }

            using (SqlDataAdapter dataAdapter = new SqlDataAdapter(cmd))
            {
                DataTable dataTable = new DataTable();
                dataAdapter.Fill(dataTable);
                dataGridView1.DataSource = dataTable;
            }
        }
    }
}

```

```

private void ClearFields()
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
}

```

```

private void UncheckAllItems()
{
    for (int i = 0; i < checkedListBox1.Items.Count; i++)
    {
        checkedListBox1.SetItemChecked(i, false);
    }
}

```

```

private void TextBox5_KeyUp(object sender, KeyEventArgs e)
{
    string filter = textBox5.Text.Trim();
    LoadDataGrid(filter);
}

```

```

private void pictureBox1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form9"] as Form1;
    if (form9 != null)
    {

```

```

        form9.Show();
    }
    else
    {
        form9 = new Form1();
        form9.Show();
    }
}

```

```

private void pictureBox4_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form9"] as Form1;
    if (form9 != null)
    {
        form9.Show();
    }
    else
    {
        form9 = new Form1();
        form9.Show();
    }
}

```

```

private void pictureBox2_Click(object sender, EventArgs e)
{
    ClearFields();
    UncheckAllItems();
    MessageBox.Show("Temizlendi!", "Bilgi", MessageBoxButtons.OK,
    MessageBoxIcon.Information);
}

```

```

private void pictureBox3_Click(object sender, EventArgs e)
{
    if (checkedListBox1.CheckedItems.Count == 0)
    {
        MessageBox.Show("Lütfen ödeme şekli seçin!", "Hata",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
}

```

```

    string telefonNo = textBox2.Text.Trim();
    if (telefonNo.Length != 11)
    {
        MessageBox.Show("Telefon numarası 11 haneli olmalıdır!", "Hata",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
}

```

```

if (!decimal.TryParse(textBox4.Text.Trim(), out decimal islemUcreti))

```

```

    {
        MessageBox.Show("Geçersiz işlem ücreti.", "Hata", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        return;
    }

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();

        // Telefon numarası var mı kontrolü
        string checkQuery = "SELECT COUNT(*) FROM Islemler WHERE TelefonNo =
        @TelefonNo";
        using (SqlCommand checkCmd = new SqlCommand(checkQuery, conn))
        {
            checkCmd.Parameters.AddWithValue("@TelefonNo", telefonNo);
            int count = (int)checkCmd.ExecuteScalar();

            string query;
            if (count > 0) // Kayıt varsa güncelle
            {
                query = "UPDATE Islemler SET YapilanIslem = @YapilanIslem, IslemUcreti
                = @IslemUcreti, OdemeSekli = @OdemeSekli, Borç = Borç + @IslemUcreti WHERE
                TelefonNo = @TelefonNo";
            }
            else // Yoksa yeni kayıt ekle
            {
                query = "INSERT INTO Islemler (AdSoyad, TelefonNo, YapilanIslem,
                IslemUcreti, OdemeSekli, Borç, KayitTarihi) VALUES (@AdSoyad, @TelefonNo,
                @YapilanIslem, @IslemUcreti, @OdemeSekli, @Borç, GETDATE())";
            }

            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@AdSoyad", textBox1.Text.Trim());
                cmd.Parameters.AddWithValue("@TelefonNo", telefonNo);
                cmd.Parameters.AddWithValue("@YapilanIslem", textBox3.Text.Trim());
                cmd.Parameters.AddWithValue("@IslemUcreti", islemUcreti);
                cmd.Parameters.AddWithValue("@OdemeSekli",
                checkedListBox1.CheckedItems[0].ToString());
                cmd.Parameters.AddWithValue("@Borç", islemUcreti);
                cmd.ExecuteNonQuery();
            }
        }
    }

    MessageBox.Show("Kayıt başarıyla eklendi veya güncellendi!", "Bilgi",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
    ClearFields();
    LoadDataGrid();

```

```

        UncheckAllItems();
    }

    private void pictureBox7_Click(object sender, EventArgs e)
    {
        if (checkedListBox1.CheckedItems.Count == 0)
        {
            MessageBox.Show("Lütfen ödeme şekli seçin!", "Hata",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        string telefonNo = textBox2.Text.Trim();
        if (!decimal.TryParse(textBox4.Text.Trim(), out decimal odenenTutar))
        {
            MessageBox.Show("Geçersiz işlem ücreti.", "Hata", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
            return;
        }

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            conn.Open();
            string updateQuery = @"
                UPDATE Islemler
                SET Borç = CASE
                    WHEN Borç - @OdenenTutar < 0 THEN 0
                    ELSE Borç - @OdenenTutar
                END, OdemeSekli = @OdemeSekli
                WHERE TelefonNo = @TelefonNo";

            using (SqlCommand updateCmd = new SqlCommand(updateQuery, conn))
            {
                updateCmd.Parameters.AddWithValue("@OdenenTutar", odenenTutar);
                updateCmd.Parameters.AddWithValue("@OdemeSekli",
                checkedListBox1.CheckedItems[0].ToString());
                updateCmd.Parameters.AddWithValue("@TelefonNo", telefonNo);
                updateCmd.ExecuteNonQuery();
            }

            // Borç kontrolü ve mesaj
            string checkQuery = "SELECT Borç FROM Islemler WHERE TelefonNo =
            @TelefonNo";
            decimal borc = 0;

            using (SqlCommand checkCmd = new SqlCommand(checkQuery, conn))
            {
                checkCmd.Parameters.AddWithValue("@TelefonNo", telefonNo);
                borc = Convert.ToDecimal(checkCmd.ExecuteScalar());
            }
        }
    }

```

```

        if (borc == 0)
        {
            MessageBox.Show("Borcu kalmamıştır!", "Bilgi", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("Borç güncellendi!", "Bilgi", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        }

        LoadDataGrid();
    }

    UncheckAllItems();
}
}
}
}

```

\*\*\*\*\*

DELETE FROM veteriner.dbo.stok;

DBCC CHECKIDENT ('veteriner.dbo.stok', RESEED, 0);      id sıfırlama

\*\*\*\*\*

// CheckedListBox'taki tüm seçimleri kaldır

```

for (int i = 0; i < checkedListBox1.Items.Count; i++)
{
    checkedListBox1.SetItemChecked(i, false);
}
MessageBox.Show("Temizlendi!", "Bilgi", MessageBoxButtons.OK,
MessageBoxIcon.Information);

```

\*\*\*\*\*

SQL:

	TableName	ColumnName	Data Type	MaxLength	IsNullable	DefaultConstraint	IndexName	IndexType	ForeignKeyName	ReferencedTable	ReferencedColumn
1	İslemler	ID	int	4	0	NULL	PK_İslemler__3214EC27C7314636	CLUSTERED	NULL	NULL	NULL
2	İslemler	AdSoyad	nvarchar	200	1	NULL	NULL	NULL	NULL	NULL	NULL
3	İslemler	TelefonNo	nvarchar	22	1	NULL	NULL	NULL	NULL	NULL	NULL
4	İslemler	Yapılanİslem	nvarchar	510	1	NULL	NULL	NULL	NULL	NULL	NULL
5	İslemler	İslemUcreti	decimal	9	1	NULL	NULL	NULL	NULL	NULL	NULL
6	İslemler	OdemeSekli	nvarchar	100	1	NULL	NULL	NULL	NULL	NULL	NULL
7	İslemler	Kayıt Tarihi	datetime	8	1	NULL	NULL	NULL	NULL	NULL	NULL
8	İslemler	Borç	decimal	9	1	NULL	NULL	NULL	NULL	NULL	NULL

FORM 8 GEBELİK TAKİP :

DOĞUM YOKSA BEYAZ RENK DOĞUMA 15 VE ÜZERİ VAR İSE YEŞİL RENKLİ DATAGRİEW VIEW1 ,

EĞER 15 GÜN VE ALTI OLURSA DATAGRİEW VIEW1 KIRMIZI OLUR...

MAIL İLE GÖNDERİLEN HER ŞEYİ BİLDİRİM OLARAKTA ALINIR.. :

WinFormsApp8

Gebelik Takip


01.04.2025 22:27:14

Anasayfa

Gebelik Takip

01.04.2025 22:28:04

Anasayfa





Küpe No: 0123

Cinsi: dana

☐ gebe

☒ boş

 Kayıt et

 Güncelle

	mu	KayıtTarihi	DogumTarihi	Telefon	Adres	HayvanSahibi	DogumaKalanSur
►		1.04.2025 22:26		05353002362	çamkonak bab...	rdvan yağlıdere	Bilinmiyor
*							

Gebelik Takip


01.04.2025 22:27:14

Anasayfa

Gebelik Takip

01.04.2025 22:47:36

Anasayfa





Küpe No: 1

Cinsi: a

☐ gebe

☒ boş

 Kayıt et

 Güncelle

	KayıtTarihi	DogumTarihi	Telefon	Adres
►	1.04.2025 22:46	2.04.2025 22:46	05353002362	çamkonak bab...
	1.04.2025 22:30		05347295393	a
*				

mail örneği:

Doğum Uyarısı - Küpe No: 0123 Inbox x



otomasyonveteriner@gmail.com

to me ▼

Küpe No: 0123

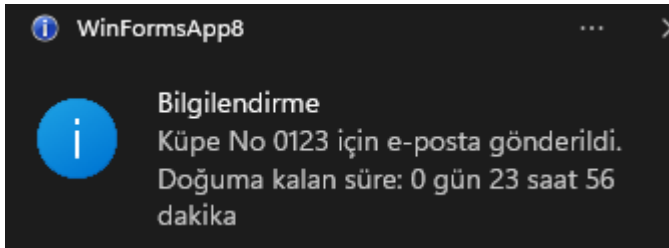
Hayvan Sahibi: rıdvan yağlıdere

Doğuma Kalan Süre: 0 gün 23 saat 59 dakika



not : maille gönderilenleri bildirim olarak verebiliyor ve bildirim sesi de var iphone bildirim sesine benzer.

## BİLDİRİM:



## C# KODLARI:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media;
using System.Collections.Generic;

namespace WinFormsApp8
{
    public partial class Form8 : Form
    {
        private readonly string connectionString = "Server=RIDVAN; Database=veteriner; Integrated Security=True;";
        private readonly System.Windows.Forms.Timer timer;
        private DataTable originalDataTable;
        private DataTable currentDataTable;
        private bool isSaving = false;
        private NotifyIcon notifyIcon;
        private Dictionary<string, DateTime> lastEmailSentTimes; // Track last email sent time per KupeNo

        public Form8()
        {
            InitializeComponent();
            timer = new System.Windows.Forms.Timer { Interval = 1000 };
            timer.Tick += Timer_Tick;
            timer.Start();
        }
    }
}
```

```

dataGridView1.AllowUserToDeleteRows = true;

// Initialize NotifyIcon for system tray notifications
notifyIcon = new NotifyIcon
{
    Icon = SystemIcons.Information,
    Visible = true
};
notifyIcon.BalloonTipClicked += (s, e) => notifyIcon.Visible = false;

// Initialize dictionary to track email times
lastEmailSentTimes = new Dictionary<string, DateTime>();
}

private async void Form8_Load(object sender, EventArgs e)
{
    await UpdateDataTable();
}

private async void Timer_Tick(object sender, EventArgs e)
{
    if (isSaving) return;

    label8.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");
    await UpdateCountdownAndSendEmails();
}

private async Task UpdateDataTable()
{
    timer.Stop();
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            await con.OpenAsync();
            string query = @"
                SELECT
                    g.KupeNo,
                    g.Cinsi,
                    g.GebelikDurumu,
                    g.KayitTarihi,
                    g.DogumTarihi,
                    h.Telefon,
                    h.Adres,
                    h.HayvanSahibi
                FROM Gebelik g
                INNER JOIN HayvanKayit h ON g.KupeNo = h.KupeNo";

            using (SqlDataAdapter da = new SqlDataAdapter(query, con))
            {

```

```

        DataTable dt = new DataTable();
        da.Fill(dt);

        if (!dt.Columns.Contains("DogumaKalanSure"))
            dt.Columns.Add("DogumaKalanSure", typeof(string));

        foreach (DataRow row in dt.Rows)
        {
            if (row["DogumTarihi"] != DBNull.Value)
            {
                DateTime dogumTarihi = Convert.ToDateTime(row["DogumTarihi"]);
                TimeSpan kalanSure = dogumTarihi - DateTime.Now;
                row["DogumaKalanSure"] = kalanSure.TotalSeconds > 0
                    ? $"{kalanSure.Days} gün {kalanSure.Hours} saat
{kalanSure.Minutes} dakika"
                    : "Doğum zamanı geldi!";
            }
            else
            {
                row["DogumaKalanSure"] = "Bilinmiyor";
            }
        }

        originalDataTable = dt.Copy();
        currentDataTable = dt.Copy();
        Invoke((MethodInvoker)(() => { dataGridView1.DataSource =
currentDataTable; }));
    }
}
catch (Exception ex)
{
    ShowSystemTrayNotification("Hata", $"Veri güncellenirken hata oluştu:
{ex.Message}", ToolTipIcon.Error);
}
finally
{
    timer.Start();
}
}

private async Task UpdateCountdownAndSendEmails()
{
    Invoke((MethodInvoker)(async () =>
    {
        foreach (DataGridViewRow row in dataGridView1.Rows)
        {
            if (row.IsNewRow || row.DataBoundItem == null) continue;

```

```

        if (row.Cells["DogumTarihi"].Value != DBNull.Value &&
row.Cells["DogumTarihi"].Value != null)
        {
            if (DateTime.TryParse(row.Cells["DogumTarihi"].Value.ToString(), out
DateTime dogumTarihi))
            {
                TimeSpan kalanSure = dogumTarihi - DateTime.Now;
                row.Cells["DogumaKalanSure"].Value = kalanSure.TotalSeconds > 0
                ? $"{kalanSure.Days} gün {kalanSure.Hours} saat {kalanSure.Minutes}
dakika"
                : "Doğum zamanı geldi!";

                if (kalanSure.TotalDays <= 15 && kalanSure.TotalSeconds > 0)
                {
                    row.DefaultCellStyle.BackColor = Color.Red;
                    row.DefaultCellStyle.ForeColor = Color.White;

                    string kupeNo = row.Cells["KupeNo"].Value?.ToString();
                    if (!string.IsNullOrEmpty(kupeNo))
                    {
                        // Check if 24 hours have passed since the last email for this
                        if (!lastEmailSentTimes.ContainsKey(kupeNo) ||
                            (DateTime.Now - lastEmailSentTimes[kupeNo]).TotalHours >=
24)
                        {
                            await SendEmailNotification(row);
                            if (lastEmailSentTimes.ContainsKey(kupeNo))
                                lastEmailSentTimes[kupeNo] = DateTime.Now;
                            else
                                lastEmailSentTimes.Add(kupeNo, DateTime.Now);

                            // Show system tray notification instead of MessageBox
                            ShowSystemTrayNotification("Bilgilendirme",
                                $"Küpe No {kupeNo} için e-posta gönderildi. Doğuma kalan
süre: {row.Cells["DogumaKalanSure"].Value}",
                                ToolTipIcon.Info);
                        }
                    }
                }
            }
            else if (kalanSure.TotalSeconds > 0)
            {
                row.DefaultCellStyle.BackColor = Color.LightGreen;
                row.DefaultCellStyle.ForeColor = Color.Black;
            }
            else
            {
                row.DefaultCellStyle.BackColor = Color.Gray;
                row.DefaultCellStyle.ForeColor = Color.White;
            }
        }
    }
}

```

```

        }
    }
    else
    {
        row.Cells["DogumaKalanSure"].Value = "Bilinmiyor";
        row.DefaultCellStyle.BackColor = Color.White;
        row.DefaultCellStyle.ForeColor = Color.Black;
    }
}
dataGridView1.Refresh();
}});
}

```

```

private void ShowSystemTrayNotification(string title, string message, ToolTipIcon
icon)
{
    notifyIcon.Visible = true;
    notifyIcon.BalloonTipTitle = title;
    notifyIcon.BalloonTipText = message;
    notifyIcon.BalloonTipIcon = icon;
    notifyIcon.ShowBalloonTip(3000); // Show for 3 seconds
}

```

```

private async Task SendEmailNotification(DataGridViewRow row)
{
    try
    {
        string gondericiEmail = "";
        string uygulamaSifresi = "";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            await conn.OpenAsync();
            string query = "SELECT TOP 1 email, password FROM [dbo].[mail]";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
            {
                if (reader.Read())
                {
                    gondericiEmail = reader["email"].ToString();
                    uygulamaSifresi = reader["password"].ToString();
                }
            }
        }

        if (string.IsNullOrEmpty(gondericiEmail) ||
string.IsNullOrEmpty(uygulamaSifresi))
        {
            ShowSystemTrayNotification("Hata", "Gönderici e-posta veya şifre
veritabanından çekilemedi.", ToolTipIcon.Error);

```

```

        return;
    }

    using (SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587))
    {
        smtpClient.Credentials = new NetworkCredential(gondericiEmail,
uygulamaSifresi);
        smtpClient.EnableSsl = true;

        using (MailMessage mailMessage = new MailMessage())
        {
            mailMessage.From = new MailAddress(gondericiEmail);
            mailMessage.To.Add("otomasyonveteriner@gmail.com");
            mailMessage.Subject = $"Doğum Uyarısı - Küpe No:
{row.Cells["KupeNo"].Value}";
            mailMessage.Body = $"Küpe No: {row.Cells["KupeNo"].Value}\n" +
                $"Hayvan Sahibi: {row.Cells["HayvanSahibi"].Value}\n" +
                $"Doğuma Kalan Süre:
{row.Cells["DogumaKalanSure"].Value}\n" +
                $"Telefon: {row.Cells["Telefon"].Value}\n" +
                $"Adres: {row.Cells["Adres"].Value}";
            mailMessage.IsBodyHtml = false;

            await smtpClient.SendMailAsync(mailMessage);
        }
    }

```

```

PlayNotificationSound("C:\\Users\\ridva\\Desktop\\dudut\\my-project\\veteriner_otomasyon
\\inek.wav");

```

```

    }
    catch (Exception ex)
    {
        ShowSystemTrayNotification("Hata", $"E-posta gönderilirken hata oluştu:
{ex.Message}", ToolTipIcon.Error);
    }
}

```

```

private void PlayNotificationSound(string filePath)
{
    try
    {
        SoundPlayer player = new SoundPlayer(filePath);
        player.Play();
    }
    catch (Exception ex)
    {
        ShowSystemTrayNotification("Ses Hatası", $"Ses çalınırken hata oluştu:
{ex.Message}", ToolTipIcon.Warning);
    }
}

```

```

    }

    private async void pictureBox2_Click(object sender, EventArgs e)
    {
        try
        {
            string kupeNo = textBox1.Text?.Trim();
            string cinsi = textBox2.Text?.Trim();
            string gebelikDurumu = checkedListBox1.CheckedItems.Cast<object>()
                .Any(item => item.ToString().Trim().Equals("Gebe",
StringComparison.OrdinalIgnoreCase)) ? "Gebe" : "Boş";

            if (string.IsNullOrEmpty(kupeNo) || string.IsNullOrEmpty(cinsi))
            {
                ShowSystemTrayNotification("Hata", "Küpe No ve Cinsi boş bırakılamaz!",
ToolTipIcon.Warning);
                return;
            }

            using (SqlConnection con = new SqlConnection(connectionString))
            {
                await con.OpenAsync();

                string checkAnimalQuery = "SELECT COUNT(*) FROM HayvanKayit WHERE
KupeNo = @KupeNo AND Cinsi = @Cinsi";
                using (SqlCommand checkAnimalCmd = new
SqlCommand(checkAnimalQuery, con))
                {
                    checkAnimalCmd.Parameters.AddWithValue("@KupeNo", kupeNo);
                    checkAnimalCmd.Parameters.AddWithValue("@Cinsi", cinsi);

                    int animalCount = (int)await checkAnimalCmd.ExecuteScalarAsync();
                    if (animalCount == 0)
                    {
                        ShowSystemTrayNotification("Hata", "Girilen Küpe No ve Cinsi ile
eşleşen bir hayvan kaydı bulunamadı!", ToolTipIcon.Error);
                        return;
                    }
                }
            }

            string query = "MERGE INTO Gebelik AS target " +
                "USING (SELECT @KupeNo AS KupeNo, @Cinsi AS Cinsi) AS
source " +
                "ON target.KupeNo = source.KupeNo AND target.Cinsi =
source.Cinsi " +
                "WHEN MATCHED THEN UPDATE SET GebelikDurumu =
@GebelikDurumu, DogumTarihi = @DogumTarihi " +
                "WHEN NOT MATCHED THEN INSERT (KupeNo, Cinsi,
GebelikDurumu, KayitTarihi, DogumTarihi) " +

```

```
        "VALUES (@KupeNo, @Cinsi, @GebelikDurumu, @KayitTarihi,  
@DogumTarihi);";
```

```
        using (SqlCommand cmd = new SqlCommand(query, con))  
        {  
            cmd.Parameters.AddWithValue("@KupeNo", kupeNo);  
            cmd.Parameters.AddWithValue("@Cinsi", cinsi);  
            cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);  
            cmd.Parameters.AddWithValue("@DogumTarihi", gebelikDurumu ==  
"Gebe" ? (object)DateTime.Now.AddDays(279) : DBNull.Value);  
            cmd.Parameters.AddWithValue("@KayitTarihi", DateTime.Now);
```

```
        }  
        await cmd.ExecuteNonQuery();  
    }  
    ShowSystemTrayNotification("Bilgi", "İşlem tamamlandı!", ToolTipIcon.Info);  
    await UpdateDataTable();  
}  
catch (Exception ex)  
{  
    ShowSystemTrayNotification("Hata", $"Hata: {ex.Message}", ToolTipIcon.Error);  
}  
}
```

```
private async void pictureBox1_Click(object sender, EventArgs e)  
{  
    await SaveDataGridViewChanges();  
}
```

```
private async Task SaveDataGridViewChanges()  
{  
    try  
    {  
        isSaving = true;  
        timer.Stop();  
  
        dataGridView1.EndEdit();  
        if (dataGridView1.DataSource != null)  
        {  
            currentDataTable.AcceptChanges();  
        }  
  
        using (SqlConnection con = new SqlConnection(connectionString))  
        {  
            await con.OpenAsync();  
  
            var deletedRows = originalDataTable.AsEnumerable()  
                .Where(origRow => !currentDataTable.AsEnumerable()
```



```

        .Any(currRow => currRow["KupeNo"].ToString() ==
origRow["KupeNo"].ToString()))
        .ToList();

    foreach (var deletedRow in deletedRows)
    {
        string kupeNo = deletedRow["KupeNo"]?.ToString();
        if (string.IsNullOrEmpty(kupeNo)) continue;

        string deleteQuery = "DELETE FROM Gebelik WHERE KupeNo =
@KupeNo";
        using (SqlCommand cmd = new SqlCommand(deleteQuery, con))
        {
            cmd.Parameters.AddWithValue("@KupeNo", kupeNo);
            await cmd.ExecuteNonQueryAsync();
        }
    }

    var activeRows = currentDataTable.AsEnumerable()
        .Where(row => row.RowState != DataRowState.Deleted && row.RowState
!= DataRowState.Detached)
        .ToList();

    foreach (var row in activeRows)
    {
        string kupeNo = row["KupeNo"]?.ToString();
        string gebelikDurumu = row["GebelikDurumu"]?.ToString();
        DateTime? dogumTarihi = row["DogumTarihi"] != DBNull.Value ?
(DateTime?)row["DogumTarihi"] : null;

        if (string.IsNullOrEmpty(kupeNo) || string.IsNullOrEmpty(gebelikDurumu))
continue;

        string updateQuery = @"
            UPDATE Gebelik
            SET GebelikDurumu = @GebelikDurumu, DogumTarihi = @DogumTarihi
            WHERE KupeNo = @KupeNo";

        using (SqlCommand cmd = new SqlCommand(updateQuery, con))
        {
            cmd.Parameters.AddWithValue("@KupeNo", kupeNo);
            cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);
            cmd.Parameters.AddWithValue("@DogumTarihi", dogumTarihi ??
(object)DBNull.Value);

            await cmd.ExecuteNonQueryAsync();
        }
    }

    await UpdateDataTable();

```

```

    }

    ShowSystemTrayNotification("Bilgi", "Değişiklikler ve silme işlemleri başarıyla
    kaydedildi.", ToolTipIcon.Info);
    }
    catch (Exception ex)
    {
        ShowSystemTrayNotification("Hata", $"Değişiklikler kaydedilirken bir hata oluştu:
    {ex.Message}", ToolTipIcon.Error);
    }
    finally
    {
        isSaving = false;
        timer.Start();
    }
}

private void pictureBox3_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form9"] as Form1 ?? new Form1();
    form9.Show();
}

protected override void OnFormClosing(FormClosingEventArgs e)
{
    notifyIcon.Visible = false;
    notifyIcon.Dispose();
    base.OnFormClosing(e);
}
}
}
}

```

SQL :

	TableName	ColumnName	DataType	MaxLength	IsNullable	DefaultConstraint	IndexName	IndexType	ForeignKeyName	ReferencedTable	ReferencedColumn
1	Gebelik	id	int	4	0	NULL	PK_Gebelik__3213E83F080BB076	CLUSTERED	NULL	NULL	NULL
2	Gebelik	KupeNo	nvarchar	100	0	NULL	NULL	NULL	FK_Gebelik_HayvanKayıt	HayvanKayıt	KupeNo
3	Gebelik	GebelikDurumu	nvarchar	100	1	NULL	NULL	NULL	NULL	NULL	NULL
4	Gebelik	GebelikTarihi	datetime	8	1	NULL	NULL	NULL	NULL	NULL	NULL
5	Gebelik	DogumSuresi	datetime	8	1	NULL	NULL	NULL	NULL	NULL	NULL
6	Gebelik	HayvanSahibi	nvarchar	200	1	NULL	NULL	NULL	NULL	NULL	NULL
7	Gebelik	Adres	nvarchar	510	1	NULL	NULL	NULL	NULL	NULL	NULL
8	Gebelik	TelefonNo	nvarchar	22	1	NULL	NULL	NULL	NULL	NULL	NULL
9	Gebelik	Cinsi	nvarchar	100	1	NULL	NULL	NULL	NULL	NULL	NULL

.

\*\*\*\*\*

ortada başlatır

StartPosition

CenterScreen

▼

WindowState

Normal

\*\*\*\*\*

27.03.2025 00:48:56

// Label üzerinde saati ve tarihi güncelleme label8.Text =  
DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");

\*\*\*\*\*

## FORM 15 : AŞI TAKİP

Aşı Takip

Küpe No:

☐ Yapıldı

Cinsi:

cinsi:

☐ Yapılmadı

Aşı Adı:

Aşı adı:

Periyot:

Doz süresi:

		Cinsi	Aşı Adı	Periyot	Durum	Kayıt Tarihi	Hayvan Sahibi	Telefon	Adres
*									

NOT : AŞI DAKİ GERİ SAYIM BİTİNCE OTOMATİK MEN 30 GÜN SONRAYA GERİ SAYIM KURUYOR.

\*\*\*\*\*

Aşı adına tabloda önceden kayıtlı aşıların adını öneren kod ekledim:

Aşı Adı Önerisi



Bunu mu kastediyorsunuz: a?

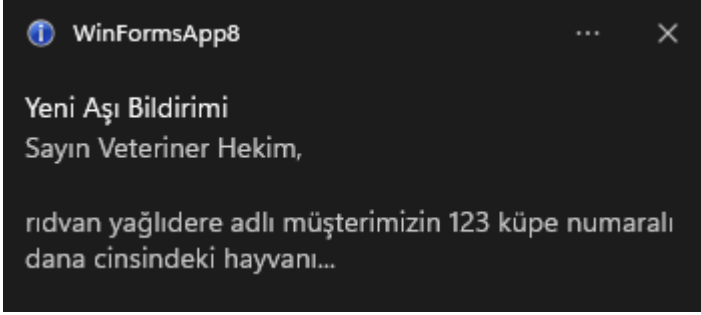
Evet

Hayır

periyot yazan textbox süreyi kastediyor ve oradaki süreyi alıp geri sayım başlatır.

\*\*\*\*\*

Bildirim gönderebiliyor :



\*\*\*\*\*

gönderdiği bildirimi mail olarakta iletiyor :

## Veteriner Otomasyon - Aşı Süresi Bildirimi Inbox x



otomasyonveteriner@gmail.com

to me ▾

Sayın Veteriner Hekim,

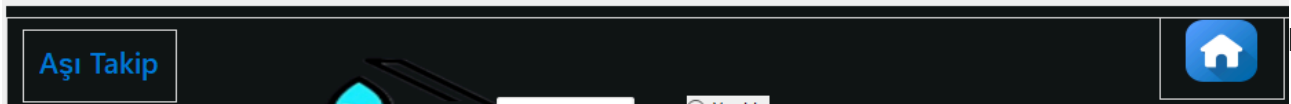
rıdvan yağlıdere adlı müşterimizin 123 küpe numaralı dana cinsindeki hayvanı için:

• a aşısının süresi 0 gün 23 saat kaldı.

İletişim: rıdvan yağlıdere, Telefon: 05353002362.

Veteriner Otomasyon Sistemi

\*\*\*\*\*



Aşı Takip



ltında

Aşı Takip

Küpe No: 123

Cinsi: dana

Aşı Adı: a

Periyot: 1

☒ Yapıldı

☐ Yapılmadı

AşıAdı	Periyot	Durum	KayıtTarihi	HayvanSahibi	Telefon	Adres	Diğer Doz
a	1	Yapıldı	1.04.2025 23:08	rıdvan yağlıdere	05353002362	camkocak bah	00 Gün 23:59:42

```

C# KODLARI:
using System;
using System.Data;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;
using System.Net;
using System.Net.Mail;
using System.Collections.Generic;
using System.Text;
using System.Media; // Ses çalma için eklendi

namespace WinFormsApp8
{
    public partial class Form15 : Form
    {
        private System.Windows.Forms.Timer countdownTimer;
        private System.Windows.Forms.Timer gridUpdateTimer;
        private System.Windows.Forms.Timer emailTimer;
        private DateTime? targetTime;
        private string connectionString = "Server=RIDVAN;Database=veteriner;Integrated
Security=True;TrustServerCertificate=True;";
        private System.Windows.Forms.Label labelCountdown;
        private NotifyIcon notifyIcon; // Bildirim simgesi için eklendi
        private string notificationSoundPath =
@"C:\Users\ridva\Desktop\dutdut\my-project\veteriner_otomasyon\bildirim.mp3";

        public Form15()
        {
            InitializeComponent();
            InitializeLabelCountdown();
            InitializeNotifyIcon(); // Bildirim simgesini başlat
            LoadDataGridView();
            InitializeGridUpdateTimer();
            InitializeEmailTimer();
            dataGridView1.KeyDown += DataGridView1_KeyDown;
            textBox3.TextChanged += TextBox3_TextChanged;
        }

        private void InitializeNotifyIcon()
        {
            notifyIcon = new NotifyIcon
            {
                Icon = SystemIcons.Information, // Sistem simgesi (isteğe bağlı olarak özel bir
simge atanabilir)
                Visible = true
            };
            notifyIcon.BalloonTipClicked += (s, e) => notifyIcon.Visible = false; // Bildirime
tıklandığında kapanır
        }
    }
}

```

```

private void InitializeLabelCountdown()
{
    labelCountdown = new System.Windows.Forms.Label
    {
        AutoSize = true,
        Location = new System.Drawing.Point(50, 200),
        Name = "labelCountdown",
        Size = new System.Drawing.Size(100, 17),
        TabIndex = 0,
        Text = "00 Gün 00:00:00"
    };
    Controls.Add(labelCountdown);
}

private void TextBox3_TextChanged(object sender, EventArgs e)
{
    string input = textBox3.Text.Trim();
    if (string.IsNullOrEmpty(input)) return;

    string suggestedAsiAdi = SuggestAsiAdi(input);
    if (!string.IsNullOrEmpty(suggestedAsiAdi))
    {
        DialogResult result = MessageBox.Show($"Bunu mu kastediyorsunuz: {suggestedAsiAdi}?", "Aşı Adı Önerisi", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
        {
            textBox3.Text = suggestedAsiAdi;
        }
    }
}

private string SuggestAsiAdi(string input)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string query = "SELECT AşıAdı FROM AsiBilgisi WHERE AşıAdı LIKE @Input"
+ "%";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Input", input);
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return reader["AşıAdı"].ToString();
                    }
                }
            }
        }
    }
}

```

```

    }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Aşı adı önerisi alınırken bir hata oluştu: " + ex.Message,
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
return null;
}

```

```

private void buttonStartCountdown_Click(object sender, EventArgs e)
{
    InitializeCountdownTimer();
}

```

```

private void InitializeCountdownTimer()
{
    if (!int.TryParse(textBox4.Text, out int days) || days <= 0)
    {
        MessageBox.Show("Lütfen geçerli bir gün sayısı girin!", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

```

```

    targetTime = DateTime.Now.AddDays(days);

    if (countdownTimer == null)
    {
        countdownTimer = new System.Windows.Forms.Timer { Interval = 1000 };
        countdownTimer.Tick += CountdownTimer_Tick;
    }
    countdownTimer.Start();
    UpdateCountdownLabel();
}

```

```

private void CountdownTimer_Tick(object sender, EventArgs e)
{
    if (targetTime == null)
    {
        countdownTimer?.Stop();
        return;
    }

```

```

    TimeSpan remainingTime = targetTime.Value - DateTime.Now;
    if (remainingTime.TotalSeconds <= 0)
    {
        countdownTimer?.Stop();
        labelCountdown.Text = "Süre doldu!";
    }

```

```

        MessageBox.Show("Geri sayım sona erdi!", "Bilgi", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
    }
    else
    {
        UpdateCountdownLabel();
    }
}

private void UpdateCountdownLabel()
{
    if (targetTime == null) return;

    TimeSpan remainingTime = targetTime.Value - DateTime.Now;
    labelCountdown.Text = $"{remainingTime.Days:D2} Gün
{remainingTime.Hours:D2}:{remainingTime.Minutes:D2}:{remainingTime.Seconds:D2}";
}

private void InitializeGridUpdateTimer()
{
    gridUpdateTimer = new System.Windows.Forms.Timer { Interval = 1000 };
    gridUpdateTimer.Tick += GridUpdateTimer_Tick;
    gridUpdateTimer.Start();
}

private void InitializeEmailTimer()
{
    emailTimer = new System.Windows.Forms.Timer { Interval = 24 * 60 * 60 * 1000 };
// 24 saat
    emailTimer.Tick += EmailTimer_Tick;
    emailTimer.Start();

    EmailTimer_Tick(this, EventArgs.Empty);
}

private void GridUpdateTimer_Tick(object sender, EventArgs e)
{
    UpdateDataGridViewCountdown();
}

private void EmailTimer_Tick(object sender, EventArgs e)
{
    SendEmailsForKupeNos();
}

private void UpdateDataGridViewCountdown()
{
    if (dataGridView1.DataSource == null) return;

    DataTable dataTable = (DataTable)dataGridView1.DataSource;

```



```

foreach (DataRow row in dataTable.Rows)
{
    if (row.RowState == DataRowState.Deleted) continue;

    if (row["DigerDoz"] != DBNull.Value)
    {
        if (DateTime.TryParse(row["DigerDoz"].ToString(), out DateTime targetDate))
        {
            TimeSpan remainingTime = targetDate - DateTime.Now;

            if (remainingTime.TotalSeconds <= 0)
            {
                if (row["Periyot"] != DBNull.Value &&
int.TryParse(row["Periyot"].ToString(), out int periyotDays))
                {
                    DateTime newDigerDoz = DateTime.Now.AddDays(periyotDays);
                    row["DigerDoz"] = newDigerDoz;
                    UpdateDigerDozInDatabase(row["KupeNo"].ToString(),
row["AşıAdı"].ToString(), newDigerDoz);

                    string message = $"⚠ {row["KupeNo"]} küpe numaralı hayvanın
{row["AşıAdı"]} aşısının süresi dolmuştur!\n" +
                    $"Yeni aşı tarihi: {newDigerDoz:dd.MM.yyyy HH:mm:ss}";
                    MessageBox.Show(message, "Aşı Süresi Bildirimi",
MessageBoxButtons.OK, MessageBoxIcon.Warning);

                    string emailBody = $"Sayın Veteriner Hekim,\n\n" +
                    $"{row["HayvanSahibi"]} adlı müşterimizin {row["KupeNo"]}
küpe numaralı {row["Cinsi"]} cinsindeki hayvanının " +
                    $"{row["AşıAdı"]} aşısının süresi dolmuştur. Yeni aşı tarihi:
{newDigerDoz:dd.MM.yyyy HH:mm:ss}.\n\n" +
                    $"İletişim: {row["HayvanSahibi"]}, Telefon:
{row["Telefon"]}.\n\n" +
                    "Veteriner Otomasyon Sistemi";
                    SendEmailNotification(emailBody);

                    remainingTime = newDigerDoz - DateTime.Now;
                }
                else
                {
                    int defaultPeriyotDays = 30;
                    DateTime newDigerDoz =
DateTime.Now.AddDays(defaultPeriyotDays);
                    row["DigerDoz"] = newDigerDoz;
                    row["Periyot"] = defaultPeriyotDays;
                    UpdateDigerDozAndPeriyotInDatabase(row["KupeNo"].ToString(),
row["AşıAdı"].ToString(), newDigerDoz, defaultPeriyotDays);

```

```

        string message = $" ⚠ {row["KupeNo"]} küpe numaralı hayvanın
{row["AşıAdı"]} aşısının süresi dolmuştur!\n" +
        $"Yeni aşı tarihi: {newDigerDoz:dd.MM.yyyy HH:mm:ss}";
        MessageBox.Show(message, "Aşı Süresi Bildirimi",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);

        string emailBody = $"Sayın Veteriner Hekim,\n\n" +
        $"{row["HayvanSahibi"]} adlı müşterimizin {row["KupeNo"]}
küpe numaralı {row["Cinsi"]} cinsindeki hayvanının " +
        $"{row["AşıAdı"]} aşısının süresi dolmuştur. Yeni aşı tarihi:
{newDigerDoz:dd.MM.yyyy HH:mm:ss}.\n\n" +
        $"İletişim: {row["HayvanSahibi"]}, Telefon:
{row["Telefon"]}.\n\n" +
        "Veteriner Otomasyon Sistemi";
        SendEmailNotification(emailBody);

        remainingTime = newDigerDoz - DateTime.Now;
    }
}

row["DigerDozDisplay"] = $"{remainingTime.Days:D2} Gün
{remainingTime.Hours:D2}:{remainingTime.Minutes:D2}:{remainingTime.Seconds:D2}";
}
else
{
    row["DigerDozDisplay"] = "Geçersiz tarih!";
}
}
else
{
    row["DigerDozDisplay"] = "";
}
}

UpdateDataGridViewRowColors();
dataGridView1.Refresh();
}

private void UpdateDigerDozInDatabase(string kupeNo, string asiAdi, DateTime
newDigerDoz)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string updateQuery = "UPDATE AsiBilgisi SET DigerDoz = @DigerDoz
WHERE KupeNo = @KupeNo AND AşıAdı = @AsiAdi";
            using (SqlCommand command = new SqlCommand(updateQuery,
connection))

```

```

        {
            command.Parameters.AddWithValue("@DigerDoz", newDigerDoz);
            command.Parameters.AddWithValue("@KupeNo", kupeNo);
            command.Parameters.AddWithValue("@AsiAdi", asiAdi);
            command.ExecuteNonQuery();
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("DigerDoz güncellenirken hata: " + ex.Message, "Hata",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void UpdateDigerDozAndPeriyotInDatabase(string kupeNo, string asiAdi,
DateTime newDigerDoz, int periyotDays)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string updateQuery = "UPDATE AsiBilgisi SET DigerDoz = @DigerDoz,
Periyot = @Periyot WHERE KupeNo = @KupeNo AND AşıAdı = @AsiAdi";
            using (SqlCommand command = new SqlCommand(updateQuery,
connection))
            {
                command.Parameters.AddWithValue("@DigerDoz", newDigerDoz);
                command.Parameters.AddWithValue("@Periyot", periyotDays);
                command.Parameters.AddWithValue("@KupeNo", kupeNo);
                command.Parameters.AddWithValue("@AsiAdi", asiAdi);
                command.ExecuteNonQuery();
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("DigerDoz ve Periyot güncellenirken hata: " + ex.Message,
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void SendEmailsForKupeNos()
{
    if (dataGridView1.DataSource == null) return;

    DataTable dataTable = (DataTable)dataGridView1.DataSource;

```

```

Dictionary<string, List<DataRow>> kupeNoGroups = new Dictionary<string,
List<DataRow>>();
foreach (DataRow row in dataTable.Rows)
{
    if (row.RowState == DataRowState.Deleted) continue;

    string kupeNo = row["KupeNo"].ToString();
    if (!kupeNoGroups.ContainsKey(kupeNo))
    {
        kupeNoGroups[kupeNo] = new List<DataRow>();
    }
    kupeNoGroups[kupeNo].Add(row);
}

foreach (var kupeNoGroup in kupeNoGroups)
{
    string kupeNo = kupeNoGroup.Key;
    List<DataRow> rows = kupeNoGroup.Value;

    StringBuilder emailBodyBuilder = new StringBuilder();
    string hayvanSahibi = "";
    string telefon = "";
    string cinsi = "";
    bool shouldSendEmail = false;

    foreach (DataRow row in rows)
    {
        if (row["DigerDoz"] != DBNull.Value)
        {
            if (DateTime.TryParse(row["DigerDoz"].ToString(), out DateTime
targetDate))
            {
                TimeSpan remainingTime = targetDate - DateTime.Now;
                if (remainingTime.TotalSeconds > 0 && remainingTime.TotalDays <= 15)
                {
                    string asiAdi = row["AşıAdı"].ToString();
                    hayvanSahibi = row["HayvanSahibi"].ToString();
                    telefon = row["Telefon"].ToString();
                    cinsi = row["Cinsi"].ToString();

                    emailBodyBuilder.AppendLine($"• {asiAdi} aşısının süresi
{remainingTime.Days} gün {remainingTime.Hours} saat kaldı.");
                    shouldSendEmail = true;
                }
            }
        }
    }

    if (shouldSendEmail)
    {

```

```

        string emailBody = $"Sayın Veteriner Hekim,\n\n" +
            $"{hayvanSahibi} adlı müşterimizin {kupeNo} küpe numaralı {cinsi}
cinsindeki hayvanı için:\n\n" +
            emailBodyBuilder.ToString() +
            $"İletişim: {hayvanSahibi}, Telefon: {telefon}.\n\n" +
            "Veteriner Otomasyon Sistemi";
        SendEmailNotification(emailBody);
    }
}
}

```

```

private void SendEmailNotification(string emailBody)
{
    try
    {
        string gondericiEmail = "";
        string uygulamaSifresi = "";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            conn.Open();
            string query = "SELECT TOP 1 email, password FROM [dbo].[mail]";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    gondericiEmail = reader["email"].ToString();
                    uygulamaSifresi = reader["password"].ToString();
                }
            }
        }

        if (string.IsNullOrEmpty(gondericiEmail) ||
string.IsNullOrEmpty(uygulamaSifresi))
        {
            return;
        }

        using (SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587))
        {
            smtpClient.Credentials = new NetworkCredential(gondericiEmail,
uygulamaSifresi);
            smtpClient.EnableSsl = true;
            smtpClient.Timeout = 10000;

            using (MailMessage mailMessage = new MailMessage())
            {
                mailMessage.From = new MailAddress(gondericiEmail);
                mailMessage.To.Add("otomasyonveteriner@gmail.com");
            }
        }
    }
}

```

```

        mailMessage.Subject = "Veteriner Otomasyon - Aşı Süresi Bildirimi";
        mailMessage.Body = emailBody;
        mailMessage.IsBodyHtml = false;

        smtpClient.Send(mailMessage);

        // Bildirimi sağ alt köşede göster ve ses çal
        notifyIcon.BalloonTipTitle = "Yeni Aşı Bildirimi";
        notifyIcon.BalloonTipText = emailBody.Length > 100 ?
emailBody.Substring(0, 100) + "...": emailBody;
        notifyIcon.ShowBalloonTip(5000); // 5 saniye görünür

        // Ses dosyasını çal
        using (SoundPlayer player = new SoundPlayer(notificationSoundPath))
        {
            player.Play();
        }
    }
}
catch (SmtpException ex)
{
    // Hata sessizce geçiliyor
}
catch (Exception ex)
{
    // Hata sessizce geçiliyor
}
}

private void LoadDataGridView()
{
    string query = @"
        SELECT a.KupeNo, a.Cinsi, a.AşıAdı, a.Periyot, a.Durum, a.DigerDoz,
a.KayitTarihi, h.HayvanSahibi, h.Telefon, h.Adres
        FROM AsiBilgisi a
        INNER JOIN HayvanKayit h ON a.KupeNo = h.KupeNo";

    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);
            DataTable dataTable = new DataTable();
            dataAdapter.Fill(dataTable);

            if (!dataTable.Columns.Contains("DigerDozDisplay"))
            {
                dataTable.Columns.Add("DigerDozDisplay", typeof(string));
            }
        }
    }
}

```

```

foreach (DataRow row in dataTable.Rows)
{
    if (row["DigerDoz"] != DBNull.Value)
    {
        DateTime targetDate = Convert.ToDateTime(row["DigerDoz"]);
        TimeSpan remainingTime = targetDate - DateTime.Now;
        row["DigerDozDisplay"] = remainingTime.TotalSeconds <= 0
            ? "Süre doldu!"
            : $"{remainingTime.Days:D2} Gün
{remainingTime.Hours:D2}:{remainingTime.Minutes:D2}:{remainingTime.Seconds:D2}";
    }
    else
    {
        row["DigerDozDisplay"] = "";
    }
}

dataGridView1.DataSource = dataTable;

if (dataGridView1.Columns["DigerDoz"] != null)
{
    dataGridView1.Columns["DigerDoz"].Visible = false;
}
if (dataGridView1.Columns["DigerDozDisplay"] != null)
{
    dataGridView1.Columns["DigerDozDisplay"].HeaderText = "Diğer Doz";
    dataGridView1.Columns["DigerDozDisplay"].Width = 120;
}

UpdateDataGridViewRowColors();
}
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message, "Hata", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}

private void UpdateDataGridViewRowColors()
{
    foreach (DataGridViewRow gridRow in dataGridView1.Rows)
    {
        if (gridRow.DataBoundItem == null) continue;

        DataRowView rowView = (DataRowView)gridRow.DataBoundItem;
        DataRow row = rowView.Row;

        if (row.RowState == DataRowState.Deleted) continue;
    }
}

```

```

gridRow.DefaultCellStyle.BackColor = System.Drawing.Color.White;

string durum = row["Durum"]?.ToString();
if (durum == "Yapıldı")
{
    gridRow.DefaultCellStyle.BackColor = System.Drawing.Color.LightGreen;
}
else if (durum == "Yapılmadı")
{
    gridRow.DefaultCellStyle.BackColor = System.Drawing.Color.White;
}

if (row["DigerDoz"] != DBNull.Value)
{
    if (DateTime.TryParse(row["DigerDoz"].ToString(), out DateTime targetDate))
    {
        TimeSpan remainingTime = targetDate - DateTime.Now;
        if (remainingTime.TotalSeconds > 0 && remainingTime.TotalDays <= 15)
        {
            gridRow.DefaultCellStyle.BackColor = System.Drawing.Color.LightCoral;
        }
    }
}
}

private void DataGridView1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Delete && dataGridView1.SelectedRows.Count > 0)
    {
        DataTable dataTable = (DataTable)dataGridView1.DataSource;
        foreach (DataGridViewRow row in dataGridView1.SelectedRows)
        {
            if (!row.IsNewRow)
            {
                DataRowView rowView = (DataRowView)row.DataBoundItem;
                if (rowView != null)
                {
                    rowView.Row.Delete();
                }
            }
        }
        dataGridView1.Refresh();
        e.Handled = true;
    }
}

private void pictureBox2_Click(object sender, EventArgs e)
{

```



```

dataGridView1.EndEdit();

try
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        int updatedRows = 0;
        int deletedRows = 0;

        string selectQuery = "SELECT KupeNo, AşıAdı FROM AsiBilgisi";
        DataTable originalTable = new DataTable();
        using (SqlDataAdapter da = new SqlDataAdapter(selectQuery, connection))
        {
            da.Fill(originalTable);
        }

        DataTable currentTable = (DataTable)dataGridView1.DataSource;

        foreach (DataRow row in currentTable.Rows)
        {
            if (row.RowState == DataRowState.Deleted)
            {
                string kupeNo = row["KupeNo", DataRowVersion.Original]?.ToString();
                string asiAdi = row["AşıAdı", DataRowVersion.Original]?.ToString();

                if (!string.IsNullOrEmpty(kupeNo) && !string.IsNullOrEmpty(asiAdi))
                {
                    string deleteQuery = "DELETE FROM AsiBilgisi WHERE KupeNo = @KupeNo AND AşıAdı = @AsiAdı";
                    using (SqlCommand cmd = new SqlCommand(deleteQuery,
connection))
                    {
                        cmd.Parameters.AddWithValue("@KupeNo", kupeNo);
                        cmd.Parameters.AddWithValue("@AsiAdı", asiAdi);
                        int rowsAffected = cmd.ExecuteNonQuery();
                        if (rowsAffected > 0) deletedRows++;
                    }
                }
            }
        }

        currentTable.AcceptChanges();

        foreach (DataRow row in currentTable.Rows)
        {
            if (row.RowState == DataRowState.Deleted) continue;

            string kupeNo = row["KupeNo"]?.ToString()?.Trim();
            string asiAdi = row["AşıAdı"]?.ToString()?.Trim();

```

```

        if (string.IsNullOrEmpty(kupeNo) || string.IsNullOrEmpty(asiAdi))
        {
            continue;
        }

        string durum = row["Durum"]?.ToString()?.Trim();
        object digerDoz = row["DigerDoz"] == DBNull.Value ? (object)DBNull.Value
: Convert.ToDateTime(row["DigerDoz"]);
        string cinsi = row["Cinsi"]?.ToString()?.Trim();
        object kayitTarihi = row["KayitTarihi"] == DBNull.Value ? DateTime.Now :
Convert.ToDateTime(row["KayitTarihi"]);

        string checkQuery = "SELECT COUNT(*) FROM AsiBilgisi WHERE
KupeNo = @KupeNo AND AşıAdı = @AsiAdi";
        using (SqlCommand checkCmd = new SqlCommand(checkQuery,
connection))
        {
            checkCmd.Parameters.AddWithValue("@KupeNo", kupeNo);
            checkCmd.Parameters.AddWithValue("@AsiAdi", asiAdi);
            int count = (int)checkCmd.ExecuteScalar();

            if (count > 0)
            {
                string updateQuery = "UPDATE AsiBilgisi SET Cinsi = @Cinsi, Durum
= @Durum, DigerDoz = @DigerDoz, KayitTarihi = @KayitTarihi WHERE KupeNo =
@KupeNo AND AşıAdı = @AsiAdi";
                using (SqlCommand cmd = new SqlCommand(updateQuery,
connection))
                {
                    cmd.Parameters.AddWithValue("@KupeNo", kupeNo);
                    cmd.Parameters.AddWithValue("@AsiAdi", asiAdi);
                    cmd.Parameters.AddWithValue("@Cinsi", cinsi);
                    cmd.Parameters.AddWithValue("@Durum", durum);
                    cmd.Parameters.AddWithValue("@DigerDoz", digerDoz);
                    cmd.Parameters.AddWithValue("@KayitTarihi", kayitTarihi);
                    int rowsAffected = cmd.ExecuteNonQuery();
                    if (rowsAffected > 0) updatedRows++;
                }
            }
        }
    }
}

LoadDataGridView();

if (updatedRows == 0 && deletedRows == 0)
{
    MessageBox.Show("Değişiklik yapılmadı!", "Bilgi",
MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

```

        else
        {
            MessageBox.Show($"Değişiklikler kaydedildi!\nGüncellenen:
{updatedRows}, Silinen: {deletedRows}",
                "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message, "Hata", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
}
}

private void pictureBox1_Click(object sender, EventArgs e)
{
    if (radioButton2.Checked)
    {
        labelCountdown.Text = "Yapılmadı";
        if (countdownTimer != null)
        {
            countdownTimer.Stop();
            targetTime = null;
        }
    }
    else if (radioButton1.Checked)
    {
        if (string.IsNullOrEmpty(textBox1.Text.Trim()) ||
string.IsNullOrEmpty(textBox2.Text.Trim()) ||
string.IsNullOrEmpty(textBox3.Text.Trim()))
        {
            MessageBox.Show("Lütfen Küpe No, Cinsi ve Aşı Adı alanlarını doldurun.",
                "Eksik Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        InitializeCountdownTimer();
    }
    HandleDatabaseOperation();
}

private void HandleDatabaseOperation()
{
    string kupeNo = textBox1.Text.Trim();
    string cinsi = textBox2.Text.Trim();
    string asiAdi = textBox3.Text.Trim();

    if (string.IsNullOrEmpty(kupeNo) || string.IsNullOrEmpty(cinsi) ||
string.IsNullOrEmpty(asiAdi))
    {

```

```

        MessageBox.Show("Lütfen Küpe No, Cinsi ve Aşı Adı alanlarını doldurun.",
"Eksik Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (!radioButton1.Checked && !radioButton2.Checked)
    {
        MessageBox.Show("Lütfen bir durum seçin.", "Eksik Seçim",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (!int.TryParse(textBox4.Text, out int days) || days <= 0)
    {
        MessageBox.Show("Lütfen geçerli bir gün sayısı girin!", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();

            string checkQuery = "SELECT COUNT(*) FROM AsiBilgisi WHERE KupeNo =
@KupeNo AND AşıAdı = @AsiAdi";
            using (SqlCommand checkCommand = new SqlCommand(checkQuery,
connection))
            {
                checkCommand.Parameters.AddWithValue("@KupeNo", kupeNo);
                checkCommand.Parameters.AddWithValue("@AsiAdi", asiAdi);
                int count = (int)checkCommand.ExecuteScalar();

                if (count > 0)
                {
                    string updateQuery = "UPDATE AsiBilgisi SET Cinsi = @Cinsi, Durum =
@Durum, DigerDoz = @DigerDoz, KayitTarihi = @KayitTarihi, Periyot = @Periyot WHERE
KupeNo = @KupeNo AND AşıAdı = @AsiAdi";
                    using (SqlCommand updateCommand = new
SqlCommand(updateQuery, connection))
                    {
                        updateCommand.Parameters.AddWithValue("@KupeNo", kupeNo);
                        updateCommand.Parameters.AddWithValue("@Cinsi", cinsi);
                        updateCommand.Parameters.AddWithValue("@AsiAdi", asiAdi);
                        updateCommand.Parameters.AddWithValue("@Durum",
radioButton1.Checked ? "Yapıldı" : "Yapılmadı");
                        updateCommand.Parameters.AddWithValue("@DigerDoz",
radioButton1.Checked ? (object)DateTime.Now.AddDays(days) : DBNull.Value);

```

```

        updateCommand.Parameters.AddWithValue("@KayitTarihi",
DateTime.Now);
        updateCommand.Parameters.AddWithValue("@Periyot", days);

        updateCommand.ExecuteNonQuery();
    }

    MessageBox.Show($"Aşı durumu güncellendi: {(radioButton1.Checked ?
"Yapıldı" : "Yapılmadı")}.", "Başarılı", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
    else
    {
        DateTime? digerDozDateTime = radioButton1.Checked ?
DateTime.Now.AddDays(days) : (DateTime?)null;

        string insertQuery = "INSERT INTO AsiBilgisi (KupeNo, Cinsi, AşıAdı,
Periyot, Durum, DigerDoz, KayitTarihi) VALUES (@KupeNo, @Cinsi, @AsiAdi, @Periyot,
@Durum, @DigerDoz, @KayitTarihi)";
        using (SqlCommand insertCommand = new SqlCommand(insertQuery,
connection))
        {
            insertCommand.Parameters.AddWithValue("@KupeNo", kupeNo);
            insertCommand.Parameters.AddWithValue("@Cinsi", cinsi);
            insertCommand.Parameters.AddWithValue("@AsiAdi", asiAdi);
            insertCommand.Parameters.AddWithValue("@Periyot", days);
            insertCommand.Parameters.AddWithValue("@Durum",
radioButton1.Checked ? "Yapıldı" : "Yapılmadı");
            insertCommand.Parameters.AddWithValue("@DigerDoz",
(object)digerDozDateTime ?? DBNull.Value);
            insertCommand.Parameters.AddWithValue("@KayitTarihi",
DateTime.Now);

            insertCommand.ExecuteNonQuery();
        }

        MessageBox.Show($"Aşı bilgisi kaydedildi: {(radioButton1.Checked ?
"Yapıldı" : "Yapılmadı")}.", "Başarılı", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
}

LoadDataGridView();
}
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message, "Hata", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
}

```

```
}
```

```
private void pictureBox3_Click(object sender, EventArgs e)
```

```
{
```

```
    textBox1.Clear();
```

```
    textBox2.Clear();
```

```
    textBox3.Clear();
```

```
    textBox4.Clear();
```

```
    radioButton1.Checked = false;
```

```
    radioButton2.Checked = false;
```

```
    labelCountdown.Text = "00 Gün 00:00:00";
```

```
    if (countdownTimer != null)
```

```
    {
```

```
        countdownTimer.Stop();
```

```
        targetTime = null;
```

```
    }
```

```
}
```

```
private void pictureBox13_Click(object sender, EventArgs e)
```

```
{
```

```
    this.Hide();
```

```
    Form1 form1 = Application.OpenForms["Form1"] as Form1;
```

```
    if (form1 != null)
```

```
    {
```

```
        form1.Show();
```

```
    }
```

```
    else
```

```
    {
```

```
        form1 = new Form1();
```

```
        form1.Show();
```

```
    }
```

```
}
```

```
protected override void OnFormClosing(FormClosingEventArgs e)
```

```
{
```

```
    notifyIcon.Dispose(); // Form kapanırken NotifyIcon'u temizle
```

```
    base.OnFormClosing(e);
```

```
}
```

```
}
```

```
}
```

SQL:

```
SQLQuery2.sql - RI...(RIDVAN\ridva (54)) x
/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [AşıAdı]
, [Periyot]
, [HayvanID]
, [HayvanSahibi]
, [Telefon]
, [KupeNo]
, [Cinsi]
, [Adres]
, [Durum]
, [DigerDoz]
, [KayitTarihi]
FROM [veteriner].[dbo].[AsiBilgisi]
```

100 %

Results Messages


TableName	ColumnName	DataType	MaxLength	IsNullable	DefaultConstraint	IndexName	IndexType	ForeignKeyName	ReferencedTable	ReferencedColumn
-----------	------------	----------	-----------	------------	-------------------	-----------	-----------	----------------	-----------------	------------------

FORM7: MAIL AYARLARI:

mail adresi ve google dan alınan eesi buray girilir ve girilen bu uygulama şifresine göre google maile erişim mail gönderilebilir:




not : raporlar hariç hiçbir mail c# kodlarında yazılı olan mail adresi dışında bir yere gitmez ,  
değiştirmek istenir ise eğer c#kodundan değiştirilmeli.

Kayıt Bölümü



Mail adresi:

Şifre:

 Temizle  Kayıt Et  Güncelle

C# KODLARI:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace WinFormsApp8
{
    public partial class Form7 : Form
    {
        // SQL Server bağlantı dizesi
        string connectionString = "Server=RIDVAN;Database=veteriner;Integrated
ekle Security=True;";

        public Form7()
        {
            InitializeComponent();
            LoadData(); // Form açıldığında verileri yükle
            dataGridView1.KeyDown += DataGridview1_KeyDown; // Delete tuşu için olay
        }

        private void pictureBox2_Click(object sender, EventArgs e)
        {
            string email = textBox1.Text.Trim();
            string password = textBox2.Text.Trim();

            // E-posta format kontrolü
            if (!email.Contains("@"))
            {
                MessageBox.Show("Geçerli bir e-posta adresi girin!", "Hata",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            // Boş alan kontrolü
            if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(password))
            {
                MessageBox.Show("E-posta ve şifre alanları boş bırakılamaz!", "Hata",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            using (SqlConnection con = new SqlConnection(connectionString))
            {
                try
                {
                    con.Open();
                    string query = "INSERT INTO mail (email, password) VALUES (@email,
@password)";

                    using (SqlCommand cmd = new SqlCommand(query, con))
                    {
                        cmd.Parameters.AddWithValue("@email", email);
                        cmd.Parameters.AddWithValue("@password", password);
                        cmd.ExecuteNonQuery();
                    }

                    MessageBox.Show("Kayıt başarıyla eklendi!", "Bilgi",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
                    LoadData();
                }
                catch (SqlException ex)
                {
                    if (ex.Number == 2627) // UNIQUE constraint hatası

```



```

        {
            MessageBox.Show("Bu e-posta adresi zaten kayıtlı!", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            MessageBox.Show("Hata: " + ex.Message, "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    textBox1.Clear();
    textBox2.Clear();
}

// Verileri DataGridView'e yükleme fonksiyonu
private void LoadData()
{
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open();
            string query = "SELECT id, email, password FROM mail";
            SqlDataAdapter da = new SqlDataAdapter(query, con);
            DataTable dt = new DataTable();
            da.Fill(dt);

            // DataGridView'e veriyi bağla ve sütun ayarlarını yap
            dataGridView1.DataSource = dt;
            dataGridView1.Columns["id"].ReadOnly = true; // ID sütununu salt
okunur yap

        }
        catch (Exception ex)
        {
            MessageBox.Show("Veri yüklenirken hata oluştu: " + ex.Message,
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

private void pictureBox1_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox2.Clear();
}

// Delete tuşuna basıldığında satırı silme
private void DataGridView1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Delete && dataGridView1.SelectedRows.Count > 0)
    {
        foreach (DataGridViewRow row in dataGridView1.SelectedRows)
        {
            if (!row.IsNewRow) // Yeni satır değilse
            {
                dataGridView1.Rows.Remove(row); // Satırı DataGridView'den
kaldır

            }
        }
        e.Handled = true; // Olayın başka bir işlem yapmasını engelle
    }
}

private void pictureBox4_Click(object sender, EventArgs e)

```

```

{
    // DataGridView'deki düzenlemeleri tamamla
    dataGridView1.EndEdit();

    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open();
            int updatedRows = 0;
            int deletedRows = 0;

            // Mevcut veritabanı kayıtlarını al
            string selectQuery = "SELECT id FROM mail";
            DataTable originalTable = new DataTable();
            using (SqlDataAdapter da = new SqlDataAdapter(selectQuery, con))
            {
                da.Fill(originalTable);
            }

            // DataGridView'deki her satırı kontrol et ve güncelle
            foreach (DataGridViewRow row in dataGridView1.Rows)
            {
                if (row.IsNewRow) continue; // Yeni satırları atla
                if (row.Cells["id"].Value == null) continue; // ID yoksa atla

                int id = Convert.ToInt32(row.Cells["id"].Value);
                string email = row.Cells["email"].Value?.ToString()?.Trim();
                string password =
row.Cells["password"].Value?.ToString()?.Trim();

                if (string.IsNullOrEmpty(email) ||
string.IsNullOrEmpty(password))
                {
                    string deleteQuery = "DELETE FROM mail WHERE id = @id";
                    using (SqlCommand cmd = new SqlCommand(deleteQuery, con))
                    {
                        cmd.Parameters.AddWithValue("@id", id);
                        int rowsAffected = cmd.ExecuteNonQuery();
                        if (rowsAffected > 0) deletedRows++;
                    }
                }
                else
                {
                    string updateQuery = "UPDATE mail SET email = @email,
password = @password WHERE id = @id";
                    using (SqlCommand cmd = new SqlCommand(updateQuery, con))
                    {
                        cmd.Parameters.AddWithValue("@id", id);
                        cmd.Parameters.AddWithValue("@email", email);
                        cmd.Parameters.AddWithValue("@password", password);
                        int rowsAffected = cmd.ExecuteNonQuery();
                        if (rowsAffected > 0) updatedRows++;
                    }
                }
            }

            // Veritabanında olup DataGridView'de olmayan satırları sil
            DataTable currentTable = (DataTable)dataGridView1.DataSource;
            foreach (DataRow originalRow in originalTable.Rows)
            {
                int originalId = Convert.ToInt32(originalRow["id"]);
                bool exists = false;

                foreach (DataGridViewRow row in dataGridView1.Rows)

```

```

        {
            if (row.IsNewRow || row.Cells["id"].Value == null)
                continue;

            if (Convert.ToInt32(row.Cells["id"].Value) == originalId)
            {
                exists = true;
                break;
            }
        }

        if (!exists)
        {
            string deleteQuery = "DELETE FROM mail WHERE id = @id";
            using (SqlCommand cmd = new SqlCommand(deleteQuery, con))
            {
                cmd.Parameters.AddWithValue("@id", originalId);
                int rowsAffected = cmd.ExecuteNonQuery();
                if (rowsAffected > 0) deletedRows++;
            }
        }
    }

    // Verileri tekrar yükle
    LoadData();

    if (updatedRows == 0 && deletedRows == 0)
    {
        MessageBox.Show("Herhangi bir değişiklik yapılmadı!", "Bilgi",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        MessageBox.Show($"Değişiklikler başarıyla
            kaydedildi!\nGüncellenen: {updatedRows}, Silinen: {deletedRows}",
            "Bilgi", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message, "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void pictureBox3_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form1"] as Form1;
    if (form9 != null)
        form9.Show();
    else
    {
        form9 = new Form1();
        form9.Show();
    }
}
}
}

```

SQL:

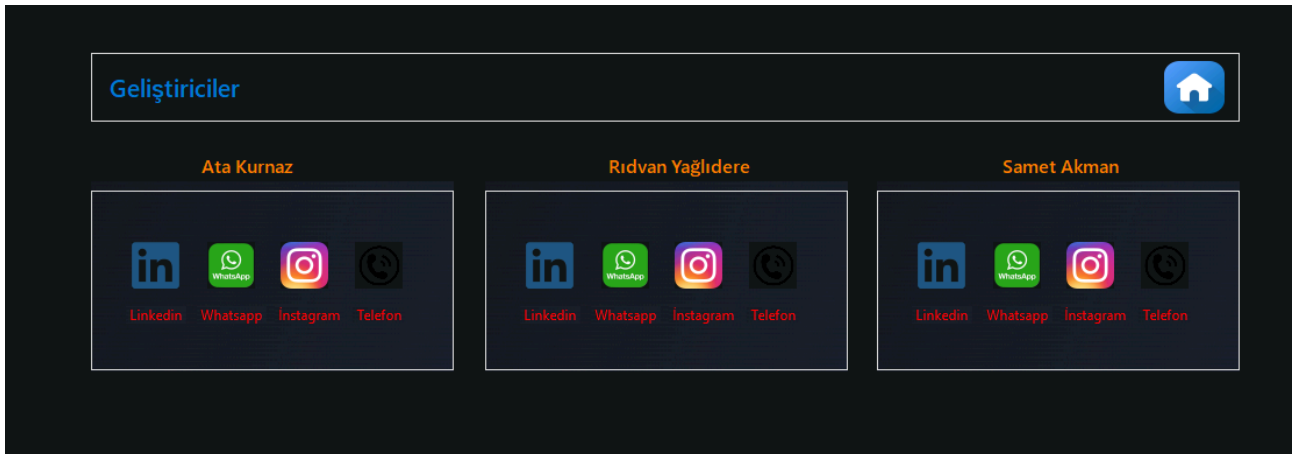
```
SQLQuery3.sql - RI...(RIDVAN\ridva (55)) x SQLQuery2.sql - RI...(RIDVAN\ridva (54))
/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [id]
      ,[email]
      ,[password]
FROM [veteriner].[dbo].[mail]
```

100 %

Results Messages

TableName	ColumnName	DataType	MaxLength	IsNullable	DefaultConstraint	IndexName	IndexType	ForeignKeyName	ReferencedTable	ReferencedColumn
-----------	------------	----------	-----------	------------	-------------------	-----------	-----------	----------------	-----------------	------------------

## FORM 9: GELİŞTİRİCİLER



NOT:sql kullanmadım burda.

C# KODLARI:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```

using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormsApp8
{
    public partial class Form9 : Form
    {
        public Form9()
        {
            InitializeComponent();
        }

        // pictureBox13'e tıklandığında Form1'i aç ve mevcut formu gizle
        private void pictureBox13_Click(object sender, EventArgs e)
        {
            Form1 form = new Form1();
            form.Show();
            this.Hide();
        }

        // pictureBox3'e tıklandığında Instagram URL'sini aç
        private void pictureBox3_Click(object sender, EventArgs e)
        {
            string url =
"https://www.instagram.com/ridvanyaglidere_?utm_source=ig_web_button_share_sheet&i
gsh=ZDNIZDc0MzIxNw=="; // Açılacak URL

            try
            {
                // URL'nin geçerli olup olmadığını kontrol et
                if (Uri.TryCreate(url, UriKind.Absolute, out Uri uriResult) &&
                    (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme ==
Uri.UriSchemeHttps))
                {
                    // ProcessStartInfo ile URL'yi varsayılan tarayıcıda aç
                    var psi = new System.Diagnostics.ProcessStartInfo
                    {
                        FileName = url,
                        UseShellExecute = true // İşletim sistemi kabuğunu kullanarak aç
                    };
                    System.Diagnostics.Process.Start(psi);
                }
                else
                {
                    MessageBox.Show("Geçersiz URL formatı.", "Hata",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            catch (Exception ex)
            {

```

```

        // Hata durumunda kullanıcıyı bilgilendir
        MessageBox.Show("URL açılırken bir hata oluştu: " + ex.Message +
            "\nLütfen varsayılan bir tarayıcı ayarlayın.",
            "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void pictureBox1_Click(object sender, EventArgs e)
{
    string url =
"https://www.linkedin.com/in/r%C4%B1dvan-ya%C4%9Fl%C4%B1dere-041800265/"; //
LinkedIn URL'si

    try
    {
        // URL'nin geçerli olup olmadığını kontrol et
        if (Uri.TryCreate(url, UriKind.Absolute, out Uri uriResult) &&
            (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme ==
Uri.UriSchemeHttps))
        {
            // ProcessStartInfo ile URL'yi varsayılan tarayıcıda aç
            var psi = new System.Diagnostics.ProcessStartInfo
            {
                FileName = url,
                UseShellExecute = true // İşletim sistemi kabuğunu kullanarak aç
            };
            System.Diagnostics.Process.Start(psi);
        }
        else
        {
            MessageBox.Show("Geçersiz URL formatı.", "Hata",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        // Hata durumunda kullanıcıyı bilgilendir
        MessageBox.Show("URL açılırken bir hata oluştu: " + ex.Message +
            "\nLütfen varsayılan bir tarayıcı ayarlayın.",
            "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void pictureBox4_Click(object sender, EventArgs e)
{
    string telefonNumarasi = "0535 300 2362";
    string isim = "RIDVAN YAĞLIDERE";

    // Telefon numarası ve ismi bir mesaj kutusunda göster

```

```

        MessageBox.Show($"İsim: {isim}\nTelefon Numarası: {telefonNumarasi}",
            "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void pictureBox2_Click(object sender, EventArgs e)
    {
        string telefonNumarasi = "9005353002362";
        string whatsappLink = $"https://wa.me/{telefonNumarasi}"; // WhatsApp bağlantısı

        try
        {
            // URL'nin geçerli olup olmadığını kontrol et
            if (Uri.TryCreate(whatsappLink, UriKind.Absolute, out Uri uriResult) &&
                (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme ==
Uri.UriSchemeHttps))
            {
                // ProcessStartInfo ile WhatsApp bağlantısını varsayılan tarayıcıda aç
                var psi = new System.Diagnostics.ProcessStartInfo
                {
                    FileName = whatsappLink,
                    UseShellExecute = true // İşletim sistemi kabuğunu kullanarak aç
                };
                System.Diagnostics.Process.Start(psi);
            }
            else
            {
                MessageBox.Show("Geçersiz WhatsApp URL formatı.", "Hata",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        catch (Exception ex)
        {
            // Hata durumunda kullanıcıyı bilgilendir
            MessageBox.Show("WhatsApp bağlantısı açılırken bir hata oluştu: " +
ex.Message +
            "\nLütfen WhatsApp uygulamasını kontrol edin veya varsayılan bir tarayıcı
ayarlayın.",
                "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void pictureBox9_Click(object sender, EventArgs e)
    {
        string telefonNumarasi = "+90 541 482 17 99";
        string isim = "Ata kurnaz";

        // Telefon numarası ve ismi bir mesaj kutusunda göster
    }

```

```

        MessageBox.Show($"İsim: {isim}\nTelefon Numarası: {telefonNumarasi}",
            "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void pictureBox12_Click(object sender, EventArgs e)
    {
        string telefonNumarasi = "9005414821799";
        string whatsappLink = $"https://wa.me/{telefonNumarasi}"; // WhatsApp bağlantısı

        try
        {
            // URL'nin geçerli olup olmadığını kontrol et
            if (Uri.TryCreate(whatsappLink, UriKind.Absolute, out Uri uriResult) &&
                (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme ==
Uri.UriSchemeHttps))
            {
                // ProcessStartInfo ile WhatsApp bağlantısını varsayılan tarayıcıda aç
                var psi = new System.Diagnostics.ProcessStartInfo
                {
                    FileName = whatsappLink,
                    UseShellExecute = true // İşletim sistemi kabuğunu kullanarak aç
                };
                System.Diagnostics.Process.Start(psi);
            }
            else
            {
                MessageBox.Show("Geçersiz WhatsApp URL formatı.", "Hata",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        catch (Exception ex)
        {
            // Hata durumunda kullanıcıyı bilgilendir
            MessageBox.Show("WhatsApp bağlantısı açılırken bir hata oluştu: " +
ex.Message +
            "\nLütfen WhatsApp uygulamasını kontrol edin veya varsayılan bir tarayıcı
ayarlayın.",
                "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void pictureBox10_Click(object sender, EventArgs e)
    {
        string url =
"https://www.instagram.com/krnzata/?utm_source=ig_web_button_share_sheet"; //
Açılacak URL

        try
        {

```



```

        // URL'nin geçerli olup olmadığını kontrol et
        if (Uri.TryCreate(url, UriKind.Absolute, out Uri uriResult) &&
            (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme ==
Uri.UriSchemeHttps))
        {
            // ProcessStartInfo ile URL'yi varsayılan tarayıcıda aç
            var psi = new System.Diagnostics.ProcessStartInfo
            {
                FileName = url,
                UseShellExecute = true // İşletim sistemi kabuğunu kullanarak aç
            };
            System.Diagnostics.Process.Start(psi);
        }
        else
        {
            MessageBox.Show("Geçersiz URL formatı.", "Hata",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        // Hata durumunda kullanıcıyı bilgilendir
        MessageBox.Show("URL açılırken bir hata oluştu: " + ex.Message +
            "\nLütfen varsayılan bir tarayıcı ayarlayın.",
            "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void pictureBox6_Click(object sender, EventArgs e)
{
    string telefonNumarasi = "+905353956302";
    string whatsappLink = $"https://wa.me/{telefonNumarasi}"; // WhatsApp bağlantısı

    try
    {
        // URL'nin geçerli olup olmadığını kontrol et
        if (Uri.TryCreate(whatsappLink, UriKind.Absolute, out Uri uriResult) &&
            (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme ==
Uri.UriSchemeHttps))
        {
            // ProcessStartInfo ile WhatsApp bağlantısını varsayılan tarayıcıda aç
            var psi = new System.Diagnostics.ProcessStartInfo
            {
                FileName = whatsappLink,
                UseShellExecute = true // İşletim sistemi kabuğunu kullanarak aç
            };
            System.Diagnostics.Process.Start(psi);
        }
        else
        {

```

```

        MessageBox.Show("Geçersiz WhatsApp URL formatı.", "Hata",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
catch (Exception ex)
{
    // Hata durumunda kullanıcıyı bilgilendir
    MessageBox.Show("WhatsApp bağlantısı açılırken bir hata oluştu: " +
ex.Message +
        "\nLütfen WhatsApp uygulamasını kontrol edin veya varsayılan bir tarayıcı
ayarlayın.",
        "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void pictureBox8_Click(object sender, EventArgs e)
{
    string telefonNumarasi = "+90 535 395 63 02";
    string isim = "samet akman";

    // Telefon numarası ve ismi bir mesaj kutusunda göster
    MessageBox.Show($"İsim: {isim}\nTelefon Numarası: {telefonNumarasi}",
        "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

private void pictureBox7_Click(object sender, EventArgs e)
{
    string url =
"https://www.instagram.com/smtakmann/?utm_source=ig_web_button_share_sheet"; //
Açılacak URL

    try
    {
        // URL'nin geçerli olup olmadığını kontrol et
        if (Uri.TryCreate(url, UriKind.Absolute, out Uri uriResult) &&
            (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme ==
Uri.UriSchemeHttps))
        {
            // ProcessStartInfo ile URL'yi varsayılan tarayıcıda aç
            var psi = new System.Diagnostics.ProcessStartInfo
            {
                FileName = url,
                UseShellExecute = true // İşletim sistemi kabuğunu kullanarak aç
            };
            System.Diagnostics.Process.Start(psi);
        }
        else
        {
            MessageBox.Show("Geçersiz URL formatı.", "Hata",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

```

    }
}
catch (Exception ex)
{
    // Hata durumunda kullanıcıyı bilgilendir
    MessageBox.Show("URL açılırken bir hata oluştu: " + ex.Message +
        "\nLütfen varsayılan bir tarayıcı ayarlayın.",
        "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
}
}
}

```

## FORM : RAPORLAR

sql kullanmadım bu sayfa için.

### RAPORLAR(1)

Raporlar Bölümü
2.sayfa

KAYITLAR

İsim'e göre filtreleme

	Kupe No	Cinsi	Adres
▶	0123	dana	çamko
	1	a	a
*			

Gebelik

Küpe no göre filtreleme

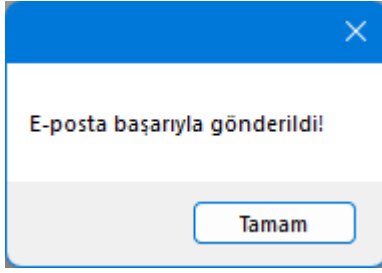
	ID	Kupe No	Cinsi
▶	1022	11	dana
	1023	41	dana
	1024	0123	dana
	1025	1	a
*			

☐ Kayıtlarım
☐ Stok Bilgileri
☐ Veresiye Defteri
☐ Gebelik Durumu
☐ Aşı takip

Gönderilecek Adres:

☒ Gönder
☐ Temizle

textbox a yazılan e posta adresine radiobuton larda seçilen tablo iletiliyor mail olarak ,birden fazla radiobutonn seçilemiyor :



C#

Veteriner Otomasyon Raporu - İşlemler - 01.04.2025 23:32:02 [Inbox x](#)



otomasyonveteriner@gmail.com

to me

11:32 PM (0 minutes ago)

Merhaba,

Bu e-posta, veteriner otomasyon sisteminden seçilen tabloya ait verileri içermektedir.

Rapor Tarihi: 01.04.2025 23:32:02

Seçilen Tablo: İşlemler

Bu rapor, veteriner otomasyon sisteminden seçilen tabloya ait verileri içermektedir. Aşağıda, seçtiğiniz tablodaki veriler detaylı bir şekilde listelenmiştir. Bu veriler, sistemdeki güncel kayıtları yansıtmaktadır ve raporun oluşturulduğu tarih ve saat yukarıda belirtilmiştir.

Tablo Verileri:

ID	AdSoyad	TelefonNo	Yapılanİşlem	İşlemÜcreti	OdemeSekli	KayıtTarihi	Borç
2019	ırdan	05353002362		100,00	KREDİ KARTI	25.03.2025 20:34:29	0,00
2020	Hamdi satiroğlu	05353002363	a	1,00	KREDİ KARTI	27.03.2025 22:34:25	0,00

İyi günler,

Veteriner Otomasyon Ekibi

KODLARI:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Net.Mail;
using System.Net;
using System.Windows.Forms;
using System.Text;
```

```
namespace WinFormsApp8
```

```
{
```

```
    public partial class Form6 : Form
```

```
    {
```

```
        private string connectionString = "Server=localhost;Database=veteriner;Integrated Security=True;";
```

```
        private DataTable originalDataTable; // DataGridView1 için orijinal verileri saklamak
```

```
        private DataTable originalPregnancyDataTable; // DataGridView2 için orijinal verileri saklamak
```

```
        public Form6()
```

```
        {
```

```
            InitializeComponent();
```

```
            LoadData(); // DataGridView1 için verileri yükle
```

```
            LoadPregnancyData(); // DataGridView2 için verileri yükle
```

```
            textBox2.TextChanged += TextBox2_TextChanged; // DataGridView1 için
```

```
TextChanged olayını ekle
```

```

        textBox3.TextChanged += TextBox3_TextChanged; // DataGridView2 için
        TextChanged olayını ekle
    }

    // Verileri DataGridView1'e yüklemek için method
    private void LoadData()
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            try
            {
                con.Open();
                string query = "SELECT TOP 1000 KupeNo, Cinsi, Adres, HayvanSahibi,
Telefon, HayvanID FROM HayvanKayit";
                SqlDataAdapter da = new SqlDataAdapter(query, con);
                originalDataTable = new DataTable();
                da.Fill(originalDataTable);

                // Verileri DataGridView1'e yükle
                dataGridView1.DataSource = originalDataTable;

                // DataGridView1 özelleştirmeleri
                CustomizeDataGridView1();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Veri yüklenirken hata oluştu: " + ex.Message);
            }
        }
    }

    // DataGridView1 düzenlemeleri
    private void CustomizeDataGridView1()
    {
        dataGridView1.Columns["KupeNo"].Width = 100;
        dataGridView1.Columns["Cinsi"].Width = 150;
        dataGridView1.Columns["Adres"].Width = 200;
        dataGridView1.Columns["HayvanSahibi"].Width = 150;
        dataGridView1.Columns["Telefon"].Width = 120;
        dataGridView1.Columns["HayvanID"].Width = 80;

        dataGridView1.Columns["KupeNo"].HeaderText = "Kupe No";
        dataGridView1.Columns["Cinsi"].HeaderText = "Cinsi";
        dataGridView1.Columns["Adres"].HeaderText = "Adres";
        dataGridView1.Columns["HayvanSahibi"].HeaderText = "Hayvan Sahibi";
        dataGridView1.Columns["Telefon"].HeaderText = "Telefon";
        dataGridView1.Columns["HayvanID"].HeaderText = "Hayvan ID";

        dataGridView1.AutoSizeColumnsMode(DataGridViewAutoSizeColumnsMode.AllCells);
    }

```

```

}

// Verileri DataGridView2'ye yüklemek için method
private void LoadPregnancyData()
{
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open();
            string query = "SELECT TOP 1000 [ID], [KupeNo], [Cinsi], [GebelikDurumu],
[KayitTarihi], [DogumTarihi], [DogumaKalanSure] FROM [veteriner].[dbo].[Gebelik]";
            SqlDataAdapter da = new SqlDataAdapter(query, con);
            originalPregnancyDataTable = new DataTable();
            da.Fill(originalPregnancyDataTable);

            // Verileri DataGridView2'ye yükle
            dataGridView2.DataSource = originalPregnancyDataTable;

            // DataGridView2 özelleştirmeleri
            CustomizeDataGridView2();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Veri yüklenirken hata oluştu: " + ex.Message);
        }
    }
}

// DataGridView2 düzenlemeleri
private void CustomizeDataGridView2()
{
    dataGridView2.Columns["ID"].Width = 80;
    dataGridView2.Columns["KupeNo"].Width = 100;
    dataGridView2.Columns["Cinsi"].Width = 150;
    dataGridView2.Columns["GebelikDurumu"].Width = 150;
    dataGridView2.Columns["KayitTarihi"].Width = 120;
    dataGridView2.Columns["DogumTarihi"].Width = 120;
    dataGridView2.Columns["DogumaKalanSure"].Width = 120;

    dataGridView2.Columns["ID"].HeaderText = "ID";
    dataGridView2.Columns["KupeNo"].HeaderText = "Kupe No";
    dataGridView2.Columns["Cinsi"].HeaderText = "Cinsi";
    dataGridView2.Columns["GebelikDurumu"].HeaderText = "Gebelik Durumu";
    dataGridView2.Columns["KayitTarihi"].HeaderText = "Kayıt Tarihi";
    dataGridView2.Columns["DogumTarihi"].HeaderText = "Doğum Tarihi";
    dataGridView2.Columns["DogumaKalanSure"].HeaderText = "Doğuma Kalan
Süre";

```

```
dataGridView2.AutoSizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);
}
```

```
// TextBox2 ile filtreleme işlemi (DataGridView1)
```

```
private void TextBox2_TextChanged(object sender, EventArgs e)
```

```
{
```

```
    if (originalDataTable != null)
```

```
    {
```

```
        string filterText = textBox2.Text.Trim();
```

```
        DataTable filteredTable = originalDataTable.Clone();
```

```
        foreach (DataRow row in originalDataTable.Rows)
```

```
        {
```

```
            if (row["KupeNo"].ToString().Contains(filterText) ||
```

```
                row["Cinsi"].ToString().Contains(filterText) ||
```

```
                row["Adres"].ToString().Contains(filterText) ||
```

```
                row["HayvanSahibi"].ToString().Contains(filterText) ||
```

```
                row["Telefon"].ToString().Contains(filterText))
```

```
            {
```

```
                filteredTable.ImportRow(row);
```

```
            }
```

```
        }
```

```
        dataGridView1.DataSource = filteredTable;
```

```
    }
```

```
}
```

```
// TextBox3 ile filtreleme işlemi (DataGridView2)
```

```
private void TextBox3_TextChanged(object sender, EventArgs e)
```

```
{
```

```
    if (originalPregnancyDataTable != null)
```

```
    {
```

```
        string filterText = textBox3.Text.Trim();
```

```
        DataTable filteredTable = originalPregnancyDataTable.Clone();
```

```
        foreach (DataRow row in originalPregnancyDataTable.Rows)
```

```
        {
```

```
            if (row["ID"].ToString().Contains(filterText) ||
```

```
                row["KupeNo"].ToString().Contains(filterText) ||
```

```
                row["Cinsi"].ToString().Contains(filterText) ||
```

```
                row["GebelikDurumu"].ToString().Contains(filterText) ||
```

```
                row["KayitTarihi"].ToString().Contains(filterText) ||
```

```
                (row["DogumTarihi"] != DBNull.Value &&
```

```
row["DogumTarihi"].ToString().Contains(filterText)) ||
```

```
                (row["DogumaKalanSure"] != DBNull.Value &&
```

```
row["DogumaKalanSure"].ToString().Contains(filterText)))
```

```
            {
```

```
                filteredTable.ImportRow(row);
```

```
            }
```

```
        }
```

```
}
```

```

        dataGridView2.DataSource = filteredTable;
    }
}

// Ana sayfaya dön butonu
private void pictureBox1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form9 = Application.OpenForms["Form1"] as Form1;
    if (form9 != null)
    {
        form9.Show();
    }
    else
    {
        form9 = new Form1();
        form9.Show();
    }
}

// Email gönderme işlemi
private void pictureBox5_Click(object sender, EventArgs e)
{
    try
    {
        // 1. Gönderici e-posta ve şifresini mail tablosundan al
        string gondericiEmail = "";
        string uygulamaSifresi = "";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            conn.Open();
            string query = "SELECT TOP 1 email, password FROM [dbo].[mail]";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                {
                    if (reader.Read())
                    {
                        gondericiEmail = reader["email"].ToString();
                        uygulamaSifresi = reader["password"].ToString();
                    }
                }
            }
        }

        if (string.IsNullOrEmpty(gondericiEmail) ||
            string.IsNullOrEmpty(uygulamaSifresi))
        {
            MessageBox.Show("Gönderici e-posta veya şifre veritabanından  
çekilemedi.");
            return;
        }
    }
}

```



```

    }

    // 2. Alıcı e-posta adresini textBox1'den al
    string alıcıEmail = textBox1.Text.Trim();
    if (string.IsNullOrEmpty(alıcıEmail))
    {
        MessageBox.Show("Lütfen alıcı e-posta adresini girin (textBox1).");
        return;
    }

    // 3. Hangi RadioButton seçiliyse o tabloyu al
    string selectedQuery = "";
    string tableName = "";

    if (radioButton1.Checked)
    {
        selectedQuery = "SELECT TOP 1000 [HayvanID], [HayvanSahibi], [Telefon], [KupeNo], [Cinsi], [Adres], [GebelikDurumu] FROM [veteriner].[dbo].[HayvanKayit]";
        tableName = "Hayvan Kayıt";
    }
    else if (radioButton2.Checked)
    {
        selectedQuery = "SELECT TOP 1000 [id], [urun_adi], [urun_no], [adet], [alis_fiyati], [satis_fiyati], [kazanc] FROM [veteriner].[dbo].[stok]";
        tableName = "Stok";
    }
    else if (radioButton3.Checked)
    {
        selectedQuery = "SELECT TOP 1000 [ID], [AdSoyad], [TelefonNo], [YapilanIslem], [IslemUcreti], [OdemeSekli], [KayitTarihi], [Borç] FROM [veteriner].[dbo].[Islemler]";
        tableName = "İşlemler";
    }
    else if (radioButton4.Checked)
    {
        selectedQuery = "SELECT TOP 1000 [ID], [KupeNo], [Cinsi], [GebelikDurumu], [KayitTarihi], [DogumTarihi], [DogumaKalanSure] FROM [veteriner].[dbo].[Gebelik]";
        tableName = "Gebelik";
    }
    else if (radioButton5.Checked)
    {
        selectedQuery = "SELECT TOP 1000 [AşıAdı], [Periyot], [HayvanID], [HayvanSahibi], [Telefon], [KupeNo], [Cinsi], [Adres], [Durum], [DigerDoz], [KayitTarihi] FROM [veteriner].[dbo].[AsiBilgisi]";
        tableName = "Aşı Bilgisi";
    }
    else
    {
        MessageBox.Show("Lütfen bir tablo seçin (RadioButton).");
    }

```

```

        return;
    }

    // 4. Veriyi al ve e-posta gövdesine ekle
    StringBuilder emailBody = new StringBuilder();
    emailBody.AppendLine("Merhaba,");
    emailBody.AppendLine();
    emailBody.AppendLine("Bu e-posta, veteriner otomasyon sisteminden seçilen
tabloya ait verileri içermektedir.");
    emailBody.AppendLine($"Rapor Tarihi: {DateTime.Now:dd.MM.yyyy
HH:mm:ss}");
    emailBody.AppendLine($"Seçilen Tablo: {tableName}");
    emailBody.AppendLine();
    emailBody.AppendLine("Bu rapor, veteriner otomasyon sisteminden seçilen
tabloya ait verileri içermektedir. Aşağıda, seçtiğiniz tablodaki veriler detaylı bir şekilde
listelenmiştir. Bu veriler, sistemdeki güncel kayıtları yansıtmaktadır ve raporun
oluşturulduğu tarih ve saat yukarıda belirtilmiştir.");
    emailBody.AppendLine();

    // Veriyi al
    DataTable tableData = new DataTable();
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();
        SqlDataAdapter da = new SqlDataAdapter(selectedQuery, conn);
        da.Fill(tableData);
    }

    // Veri kontrolü
    if (tableData.Columns.Count == 0 || tableData.Rows.Count == 0)
    {
        emailBody.AppendLine("Tabloda veri bulunamadı.");
    }
    else
    {
        // Sütun başlıklarını ekle
        emailBody.AppendLine("Tablo Verileri:");
        emailBody.AppendLine(new string('-', 50));
        StringBuilder headerLine = new StringBuilder();
        foreach (DataColumn column in tableData.Columns)
        {
            headerLine.Append($"{column.ColumnName,-20} | ");
        }
        emailBody.AppendLine(headerLine.ToString().TrimEnd(' ', '|'));
        emailBody.AppendLine(new string('-', 50));

        // Satırları ekle
        foreach (DataRow row in tableData.Rows)
        {
            StringBuilder rowLine = new StringBuilder();

```

```

        foreach (var item in row.ItemArray)
        {
            string cellValue = item?.ToString() ?? "NULL";
            // Özel karakterleri temizle
            cellValue = cellValue.Replace("\r\n", " ").Replace("\n", " ").Replace("\r", "
").Trim();

            rowLine.Append($"{cellValue,-20} | ");
        }
        emailBody.AppendLine(rowLine.ToString().TrimEnd(' ', '|'));
    }
    emailBody.AppendLine(new string('-', 50));
}

emailBody.AppendLine();
emailBody.AppendLine("İyi günler,");
emailBody.AppendLine("Veteriner Otomasyon Ekibi");

// 5. E-posta gönder
using (SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587))
{
    smtpClient.Credentials = new NetworkCredential(gondericiEmail,
uygulamaSifresi);
    smtpClient.EnableSsl = true;

    using (MailMessage mailMessage = new MailMessage())
    {
        mailMessage.From = new MailAddress(gondericiEmail);
        mailMessage.To.Add(aliciEmail);
        mailMessage.Subject = $"Veteriner Otomasyon Raporu - {tableName} -
{DateTime.Now:dd.MM.yyyy HH:mm:ss}";
        mailMessage.Body = emailBody.ToString();
        mailMessage.IsBodyHtml = false;

        smtpClient.Send(mailMessage);
    }
}

MessageBox.Show("E-posta başarıyla gönderildi!");
}
catch (Exception ex)
{
    MessageBox.Show("E-posta gönderilirken hata oluştu: " + ex.Message +
"\nStackTrace: " + ex.StackTrace);
}
}

private void pictureBox2_Click(object sender, EventArgs e)
{
    this.Hide();
    Form11 form11 = Application.OpenForms["Form11"] as Form11;

```

```

        if (form11 != null)
        {
            form11.Show();
        }
        else
        {
            form11 = new Form11();
            form11.Show();
        }
    }

    private void pictureBox13_Click(object sender, EventArgs e)
    {
        Form1 form = new Form1();
        form.Show();
        this.Hide();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        // button4 işlevselliği buraya eklenebilir
    }

    private void pictureBox1_Click_1(object sender, EventArgs e) // Temizle butonu
    {
        // TextBox'ları temizle
        textBox1.Clear();
        textBox2.Clear();
        textBox3.Clear();

        // RadioButton'ları sıfırla
        radioButton1.Checked = false;
        radioButton2.Checked = false;
        radioButton3.Checked = false;
        radioButton4.Checked = false;
        radioButton5.Checked = false;
    }
}
}
}
*****

```

## RAPORLAR 2.SAYFA

Raporlar Bölümü ( 2 )
←

Veresiye Defteri

	ID	Ad Soyad	Telefon No
▶	2019	ırdvan	05353002362
	2020	Hamdi satiroğlu	05353002363
*			

Aşı Takip

	Aşı Adı	Periyot	Hayvan ID
▶	a	30	
	a	1	
	a	1	
*			

## C# KODLARI:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace WinFormsApp8
{
    public partial class Form11 : Form
    {
        private string connectionString =
            "Server=localhost;Database=veteriner;Integrated Security=True;";
        private DataTable originalIslemlerDataTable; // DataGridView1 için orijinal
        verileri saklamak
        private DataTable originalAsiBilgisiDataTable; // DataGridView3 için orijinal
        verileri saklamak

        public Form11()
        {
            InitializeComponent();
            LoadIslemlerData(); // DataGridView1 için verileri yükle
            LoadAsiBilgisiData(); // DataGridView3 için verileri yükle
            textBox2.TextChanged += TextBox2_TextChanged; // DataGridView1 için
            TextChanged olayını ekle
            textBox1.TextChanged += TextBox1_TextChanged; // DataGridView3 için
            TextChanged olayını ekle
        }

        // Verileri DataGridView1'e yüklemek için method (Islemler tablosu)
        private void LoadIslemlerData()
        {
            using (SqlConnection con = new SqlConnection(connectionString))
            {
                try
                {
                    con.Open();
                    string query = "SELECT TOP 1000 [ID], [AdSoyad], [TelefonNo],
[YapilanIslem], [IslemUcreti], [OdemeSekli], [KayitTarihi], [Borç] FROM
[veteriner].[dbo].[Islemler]";
                    SqlDataAdapter da = new SqlDataAdapter(query, con);
                    originalIslemlerDataTable = new DataTable();
                    da.Fill(originalIslemlerDataTable);

                    // Verileri DataGridView1'e yükle
                    dataGridView1.DataSource = originalIslemlerDataTable;

                    // DataGridView1 özelleştirmeleri
                    CustomizeDataGridView1();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Veri yüklenirken hata oluştu: " + ex.Message);
                }
            }
        }

        // DataGridView1 düzenlemeleri
        private void CustomizeDataGridView1()
        {
            dataGridView1.Columns["ID"].Width = 80;
            dataGridView1.Columns["AdSoyad"].Width = 150;
        }
    }
}
```

```

dataGridView1.Columns["TelefonNo"].Width = 120;
dataGridView1.Columns["YapilanIslem"].Width = 150;
dataGridView1.Columns["IslemUcreti"].Width = 100;
dataGridView1.Columns["OdemeSekli"].Width = 100;
dataGridView1.Columns["KayitTarihi"].Width = 120;
dataGridView1.Columns["Borç"].Width = 100;

dataGridView1.Columns["ID"].HeaderText = "ID";
dataGridView1.Columns["AdSoyad"].HeaderText = "Ad Soyad";
dataGridView1.Columns["TelefonNo"].HeaderText = "Telefon No";
dataGridView1.Columns["YapilanIslem"].HeaderText = "Yapılan İşlem";
dataGridView1.Columns["IslemUcreti"].HeaderText = "İşlem Ücreti";
dataGridView1.Columns["OdemeSekli"].HeaderText = "Ödeme Şekli";
dataGridView1.Columns["KayitTarihi"].HeaderText = "Kayıt Tarihi";
dataGridView1.Columns["Borç"].HeaderText = "Borç";

dataGridView1.AutoSizeColumnsMode(DataGridViewAutoSizeColumnsMode.AllCells);
}

// Verileri DataGridView3'e yüklemek için method (AsiBilgisi tablosu)
private void LoadAsiBilgisiData()
{
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open();
            string query = "SELECT TOP 1000 [AşıAdı], [Periyot], [HayvanID],
[HayvanSahibi], [Telefon], [KupeNo], [Cinsi], [Adres], [Durum], [DigerDoz],
[KayitTarihi] FROM [veteriner].[dbo].[AsiBilgisi]";
            SqlDataAdapter da = new SqlDataAdapter(query, con);
            originalAsiBilgisiDataTable = new DataTable();
            da.Fill(originalAsiBilgisiDataTable);

            // Verileri DataGridView3'e yükle
            dataGridView3.DataSource = originalAsiBilgisiDataTable;

            // DataGridView3 özelleştirmeleri
            CustomizeDataGridView3();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Veri yüklenirken hata oluştu: " + ex.Message);
        }
    }
}

// DataGridView3 düzenlemeleri
private void CustomizeDataGridView3()
{
    dataGridView3.Columns["AşıAdı"].Width = 120;
    dataGridView3.Columns["Periyot"].Width = 100;
    dataGridView3.Columns["HayvanID"].Width = 80;
    dataGridView3.Columns["HayvanSahibi"].Width = 150;
    dataGridView3.Columns["Telefon"].Width = 120;
    dataGridView3.Columns["KupeNo"].Width = 100;
    dataGridView3.Columns["Cinsi"].Width = 100;
    dataGridView3.Columns["Adres"].Width = 150;
    dataGridView3.Columns["Durum"].Width = 100;
    dataGridView3.Columns["DigerDoz"].Width = 100;
    dataGridView3.Columns["KayitTarihi"].Width = 120;

    dataGridView3.Columns["AşıAdı"].HeaderText = "Aşı Adı";
    dataGridView3.Columns["Periyot"].HeaderText = "Periyot";
    dataGridView3.Columns["HayvanID"].HeaderText = "Hayvan ID";

```

```

dataGridView3.Columns["HayvanSahibi"].HeaderText = "Hayvan Sahibi";
dataGridView3.Columns["Telefon"].HeaderText = "Telefon"; // Hata burada
düzeltildi
dataGridView3.Columns["KupeNo"].HeaderText = "Kupe No";
dataGridView3.Columns["Cinsi"].HeaderText = "Cinsi";
dataGridView3.Columns["Adres"].HeaderText = "Adres";
dataGridView3.Columns["Durum"].HeaderText = "Durum";
dataGridView3.Columns["DigerDoz"].HeaderText = "Diğer Doz";
dataGridView3.Columns["KayitTarihi"].HeaderText = "Kayıt Tarihi";

dataGridView3.AutoSizeColumnsMode(DataGridViewAutoSizeColumnsMode.AllCells);
}

// TextBox2 ile filtreleme işlemi (DataGridView1 - İşlemler)
private void TextBox2_TextChanged(object sender, EventArgs e)
{
    if (originalIslemlerDataTable != null)
    {
        string filterText = textBox2.Text.Trim();
        DataTable filteredTable = originalIslemlerDataTable.Clone();

        foreach (DataRow row in originalIslemlerDataTable.Rows)
        {
            if (row["ID"].ToString().Contains(filterText) ||
                row["AdSoyad"].ToString().Contains(filterText) ||
                row["TelefonNo"].ToString().Contains(filterText) ||
                row["YapilanIslem"].ToString().Contains(filterText) ||
                row["IslemUcreti"].ToString().Contains(filterText) ||
                row["OdemeSekli"].ToString().Contains(filterText) ||
                row["KayitTarihi"].ToString().Contains(filterText) ||
                row["Borç"].ToString().Contains(filterText))
            {
                filteredTable.ImportRow(row);
            }
        }
        dataGridView1.DataSource = filteredTable;
    }
}

// TextBox1 ile filtreleme işlemi (DataGridView3 - AsiBilgisi)
private void TextBox1_TextChanged(object sender, EventArgs e)
{
    if (originalAsiBilgisiDataTable != null)
    {
        string filterText = textBox1.Text.Trim();
        DataTable filteredTable = originalAsiBilgisiDataTable.Clone();

        foreach (DataRow row in originalAsiBilgisiDataTable.Rows)
        {
            if (row["AşıAdı"].ToString().Contains(filterText) ||
                row["Periyot"].ToString().Contains(filterText) ||
                row["HayvanID"].ToString().Contains(filterText) ||
                row["HayvanSahibi"].ToString().Contains(filterText) ||
                row["Telefon"].ToString().Contains(filterText) ||
                row["KupeNo"].ToString().Contains(filterText) ||
                row["Cinsi"].ToString().Contains(filterText) ||
                row["Adres"].ToString().Contains(filterText) ||
                row["Durum"].ToString().Contains(filterText) ||
                (row["DigerDoz"] != DBNull.Value &&
row["DigerDoz"].ToString().Contains(filterText)) ||
                row["KayitTarihi"].ToString().Contains(filterText))
            {
                filteredTable.ImportRow(row);
            }
        }
    }
}

```

```

        dataGridView3.DataSource = filteredTable;
    }
}

private void pictureBox2_Click(object sender, EventArgs e)
{
    this.Hide();
    Form6 form6 = Application.OpenForms["Form6"] as Form6;
    if (form6 != null)
    {
        form6.Show();
    }
    else
    {
        form6 = new Form6();
        form6.Show();
    }
}
}
}

```

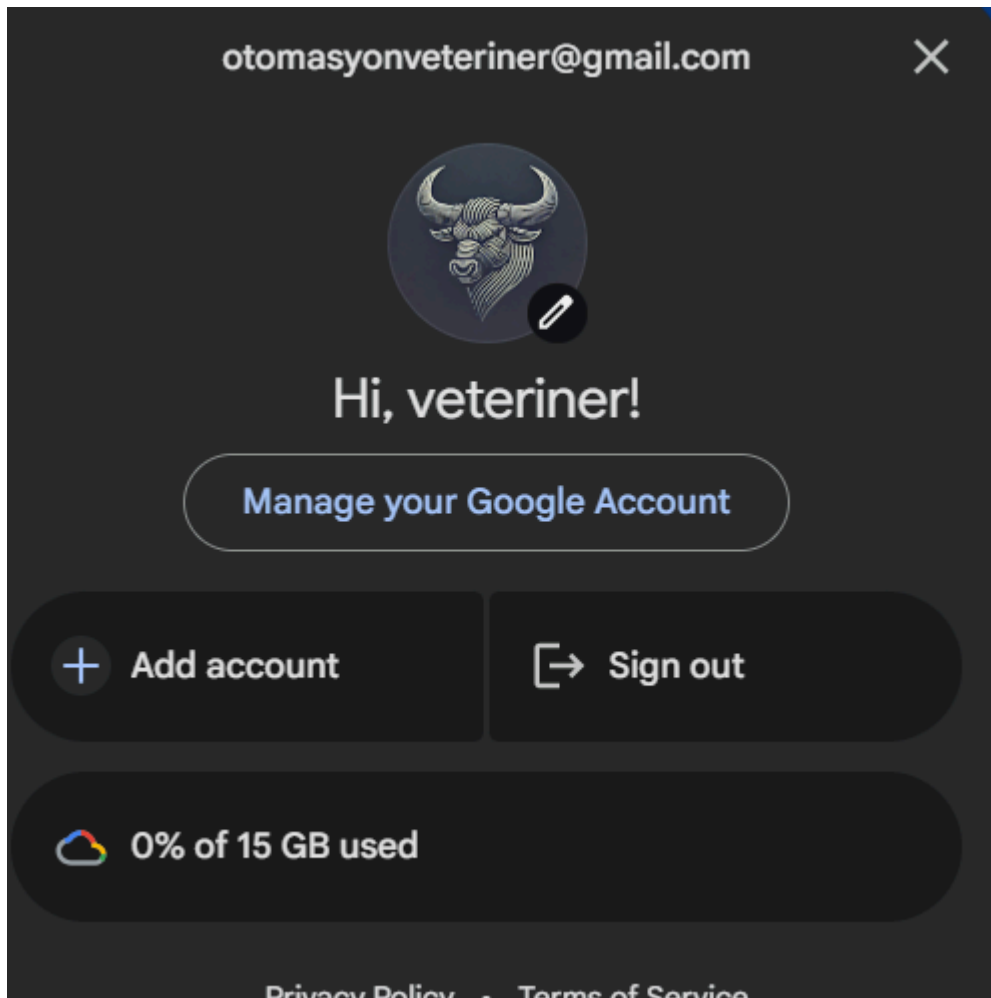
MAİL GÖNDERME

<https://youtu.be/DRVU88C7Mc4?si=2f67k9LQDXXc8KKu> ---GOOGLE UYGULAMA ŞİFRESİ İÇİN İZLENEBİLİR.

Google uygulama şifresi alma

1.ADIM:

Google hesaplarımı yönet yapılır

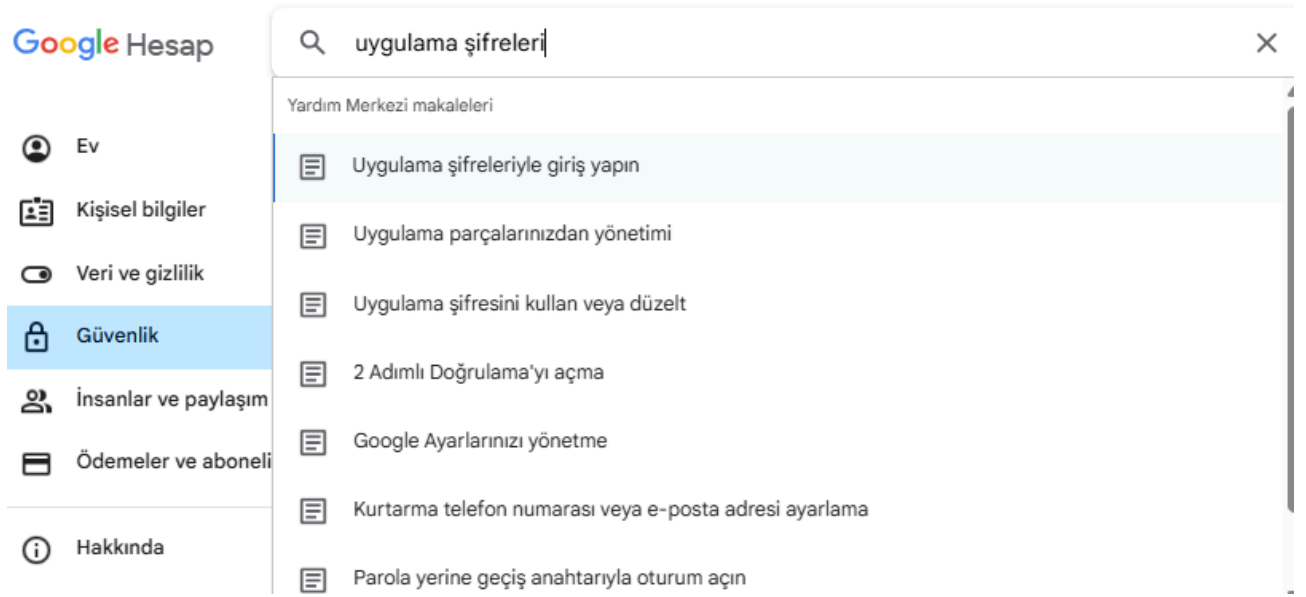




## 2.ADIM

çift faktörlü doğrulama açılır

## 3.ADIM



## 4.ADIM

şifre girilir

## 5.ADIM

uygulama adı girilir ve google şifre atar o kullanılır.

### ← Uygulama şifreleri

Uygulama şifreleri, modern güvenlik standartlarını desteklemeyen eski uygulama ve hizmetlerde Google Hesabınızda oturum açmanıza yardımcı olur.

Uygulama şifreleri, modern güvenlik standartlarını kullanan güncel uygulama ve hizmetlerden daha az güvenlidir. Uygulama şifresi oluşturmadan önce, uygulamanızda oturum açmak için buna ihtiyaç olup olmadığını kontrol etmeniz gerekir.

[Daha fazla bilgi](#)

#### Uygulama şifreleriniz

veterinerotomasyon_aktivasyon	Oluşturulma: 3 Nis, son kullanım: 02:59	
veteriner_omu	Oluşturulma: 31 Mar, son kullanım: 01:17	

Örnek kodlar:

mail gönderme

\*\*\*\*\*

```
using System;
using System.Data.SqlClient;
using System.Net;
using System.Net.Mail;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        try
```

```
        {
```

```
            // Windows Authentication ile SQL Server bağlantı dizesi
```

```
            string connectionString = "Server=RIDVAN;Database=veteriner;Integrated
Security=True;";
```

```
            string gondericiEmail = "";
```

```
            string uygulamaSifresi = "";
```

```
            // Veritabanından e-posta ve uygulama şifresini çekme
```

```
            using (SqlConnection conn = new SqlConnection(connectionString))
```

```
            {
```

```
                conn.Open();
```

```
                string query = "SELECT TOP 1 email, password FROM [dbo].[mail]";
```

```
                using (SqlCommand cmd = new SqlCommand(query, conn))
```

```
                using (SqlDataReader reader = cmd.ExecuteReader())
```

```
                {
```

```
                    if (reader.Read())
```

```
                    {
```

```
                        gondericiEmail = reader["email"].ToString();
```

```
                        uygulamaSifresi = reader["password"].ToString();
```

```
                    }
```

```
                }
```

```
            }
```

```
            if (string.IsNullOrEmpty(gondericiEmail) || string.IsNullOrEmpty(uygulamaSifresi))
```

```

        {
            Console.WriteLine("⚠ Gönderici e-posta veya şifre veritabanından
çekilemedi.");
            return;
        }

        // SMTP sunucusu ve e-posta gönderme
        using (SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587))
        {
            smtpClient.Credentials = new NetworkCredential(gondericiEmail,
uygulamaSifresi);
            smtpClient.EnableSsl = true;

            using (MailMessage mailMessage = new MailMessage())
            {
                mailMessage.From = new MailAddress(gondericiEmail);
                mailMessage.To.Add("otomasyonveteriner@gmail.com"); // Alıcı e-posta
adresi
                mailMessage.Subject = "Veteriner Otomasyon - E-posta Testi";
                mailMessage.Body = "Bu bir test e-postasıdır.";
                mailMessage.IsBodyHtml = false;

                smtpClient.Send(mailMessage);
            }
        }

        Console.WriteLine("✅ E-posta başarıyla gönderildi!");
    }
    catch (Exception ex)
    {
        Console.WriteLine("❌ Bir hata oluştu: " + ex.Message);
    }
}

```

---

## FORM 12: AKTİVASYON KAYIT EKRANI

```

using System;
using System.Net;
using System.Net.Mail;
using System.Net.NetworkInformation; // MAC adresi için
using System.Text;
using System.Windows.Forms;
using MailKit.Net.Imap;
using MailKit;
using MimeKit;

namespace WinFormsApp8

```

```

{
public partial class Form12 : Form
{
    private string sistemEmail = "otomasyonveteriner@gmail.com";
    private string emailPassword = "ftkx mebp kdfn ufrj"; // Gmail uygulama şifresi
    private string currentTelefonNo; // Onay kontrolü için telefon numarasını sakla
    private string currentAktivasyonKodu; // Onay kontrolü için aktivasyon kodunu sakla

    public Form12()
    {
        InitializeComponent();
    }

    private void pictureBox4_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form10 form10 = Application.OpenForms["Form10"] as Form10;
        if (form10 != null)
            form10.Show();
        else
        {
            form10 = new Form10();
            form10.Show();
        }
    }

    private void pictureBox3_Click(object sender, EventArgs e)
    {
        try
        {
            string isimSoyisim = TruncateString(textBox1.Text, 255);
            string telefonNo = TruncateString(textBox3.Text, 11);
            string adres = TruncateString(richTextBox1.Text, 255);
            string macAddress = GetMacAddress(); // MAC adresini al
            string ipAddress = GetIpAddress(); // IP adresini al

            if (string.IsNullOrEmpty(telefonNo))
            {
                MessageBox.Show("Lütfen telefon numaranızı giriniz!");
                return;
            }

            // Telefon numarasının daha önce kullanılıp kullanılmadığını kontrol et
            if (HasPhoneNumberReceivedCode(telefonNo))
            {
                MessageBox.Show("Bu telefon numarası daha önce kullanılmış, başka bir
numara girin!");
                return;
            }
        }
    }
}

```

```
string aktivasyonKodu = GenerateUniqueActivationCode(macAddress,
ipAddress); // MAC ve IP'ye göre kod üret
currentTelefonNo = telefonNo; // Onay kontrolü için sakla
currentAktivasyonKodu = aktivasyonKodu; // Onay kontrolü için sakla
```

```
SendActivationEmail(sistemEmail, aktivasyonKodu, isimSoyisim, adres,
telefonNo, macAddress, ipAddress);
```

```
MessageBox.Show("Kayıt başarıyla eklendi! Aktivasyon kodu e-posta adresinize
gönderildi. Lütfen giriş yapınız.");
```

```
// Kullanıcı aktivasyonunu kontrol et
if (CheckEmailForActivation(telefonNo, aktivasyonKodu))
{
    SendStatusEmail(sistemEmail, "Giriş Başarılı", telefonNo, aktivasyonKodu);
    textBox1.Clear();
    textBox3.Clear();
    richTextBox1.Clear();

    this.Hide();
    Form10 form10 = Application.OpenForms["Form10"] as Form10;
    if (form10 != null)
        form10.Show();
    else
    {
        form10 = new Form10();
        form10.Show();
    }
}
else
{
    SendStatusEmail(sistemEmail, "Giriş Başarısız - Hatalı Kod", telefonNo,
aktivasyonKodu);
    MessageBox.Show("E-posta ile gönderilen telefon numarası veya aktivasyon
kodu eşleşmedi! Aktivasyon kodu iptal edildi.");
}
}
catch (Exception ex)
{
    MessageBox.Show("Hata oluştu: " + ex.Message);
}
}
```

```
// Telefon numarasının daha önce kod alıp almadığını kontrol et
private bool HasPhoneNumberReceivedCode(string telefonNo)
{
    try
    {
        using (var client = new ImapClient())
        {
```

```

        client.Connect("imap.gmail.com", 993, true);
        client.Authenticate(sistemEmail, emailPassword);

        var inbox = client.Inbox;
        inbox.Open(FolderAccess.ReadOnly);

        var messages = inbox.Fetch(0, -1, MessageSummaryItems.Full |
MessageSummaryItems.Uniqueld);
        foreach (var message in messages)
        {
            var email = inbox.GetMessage(message.Uniqueld);
            string emailBody = email.TextBody ?? "";
            if (emailBody.Contains(telefonNo) && emailBody.Contains("Aktivasyon
Kodunuz"))
            {
                client.Disconnect(true);
                return true; // Telefon numarası daha önce kod almış
            }
        }

        client.Disconnect(true);
        return false; // Telefon numarası kod almamış
    }
}
catch (Exception ex)
{
    MessageBox.Show("Telefon numarası kontrolü sırasında hata: " + ex.Message);
    return false;
}
}

// E-posta ile aktivasyon kodunu kontrol et
private bool CheckEmailForActivation(string telefonNo, string aktivasyonKodu)
{
    try
    {
        using (var client = new ImapClient())
        {
            client.Connect("imap.gmail.com", 993, true);
            client.Authenticate(sistemEmail, emailPassword);

            var inbox = client.Inbox;
            inbox.Open(FolderAccess.ReadOnly);

            var messages = inbox.Fetch(0, -1, MessageSummaryItems.Full |
MessageSummaryItems.Uniqueld);
            for (int i = Math.Max(0, messages.Count - 5); i < messages.Count; i++)
            {
                var message = inbox.GetMessage(messages[i].Uniqueld);
                string emailBody = message.TextBody ?? "";

```

```

        if (emailBody.Contains(telefonNo) &&
emailBody.Contains(aktivasyonKodu))
        {
            client.Disconnect(true);
            return true;
        }
    }

    client.Disconnect(true);
    return false;
}
}
catch (Exception ex)
{
    MessageBox.Show("E-posta kontrolü sırasında hata: " + ex.Message);
    return false;
}
}

```

// MAC adresini alma

private string GetMacAddress()

```

{
    try
    {
        NetworkInterface[] interfaces = NetworkInterface.GetAllNetworkInterfaces();
        foreach (NetworkInterface ni in interfaces)
        {
            if (ni.OperationalStatus == OperationalStatus.Up)
            {
                return ni.GetPhysicalAddress().ToString();
            }
        }
        return "UnknownMAC";
    }
    catch (Exception)
    {
        return "ErrorMAC";
    }
}

```

// IP adresini alma

private string GetIpAddress()

```

{
    try
    {
        string hostName = Dns.GetHostName();
        IPAddress[] addresses = Dns.GetHostAddresses(hostName);
        foreach (IPAddress address in addresses)
        {

```

```

        if (address.AddressFamily ==
System.Net.Sockets.AddressFamily.InterNetwork) // IPv4
        {
            return address.ToString();
        }
    }
    return "UnknownIP";
}
catch (Exception)
{
    return "ErrorIP";
}
}

// MAC ve IP'ye göre benzersiz aktivasyon kodu üret
private string GenerateUniqueActivationCode(string macAddress, string ipAddress)
{
    Random random = new Random();
    const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    StringBuilder code = new StringBuilder();

    // MAC ve IP'den türetilmiş bir başlangıç ekle
    string macPart = macAddress.Length >= 6 ? macAddress.Substring(0, 6) :
macAddress.PadRight(6, '0');
    string ipPart = ipAddress.Replace(".", "").Length >= 4 ? ipAddress.Replace(".",
""").Substring(0, 4) : ipAddress.Replace(".", "").PadRight(4, '0');
    code.Append(macPart.Substring(0, 3).ToUpper()); // MAC'in ilk 3 karakteri
    code.Append(ipPart.Substring(0, 2).ToUpper()); // IP'nin ilk 2 karakteri

    // Rastgele 6 karakter ekle
    bool hasLetter = false, hasDigit = false;
    while (code.Length < 11 || !hasLetter || !hasDigit)
    {
        char nextChar = chars[random.Next(chars.Length)];
        code.Append(nextChar);
        if (char.IsLetter(nextChar)) hasLetter = true;
        if (char.IsDigit(nextChar)) hasDigit = true;
    }

    return code.ToString().Substring(0, 11); // 11 karaktere sabitle
}

// Aktivasyon e-postası gönder (MAC ve IP eklendi)
private void SendActivationEmail(string email, string aktivasyonKodu, string
isimSoyisim, string adres, string telefonNo, string macAddress, string ipAddress)
{
    try
    {
        MailMessage mail = new MailMessage();
        SmtplibClient smtpServer = new SmtplibClient("smtp.gmail.com");
    }
}

```



```

mail.From = new MailAddress(sistemEmail);
mail.To.Add(email);
mail.Subject = "Aktivasyon Kodunuz";
mail.Body = $"Merhaba,\n\n" +
    $"Veteriner otomasyon sistemine kaydınız başarıyla alınmıştır. Aşağıda
kayıt bilgileriniz ve aktivasyon kodunuz yer almaktadır:\n\n" +
    $"İsim Soyisim: {isimSoyisim}\n" +
    $"Adres: {adres}\n" +
    $"Telefon Numarası: {telefonNo}\n" +
    $"MAC Adresi: {macAddress}\n" +
    $"IP Adresi: {ipAddress}\n" +
    $"Aktivasyon Kodunuz: {aktivasyonKodu}\n\n" +
    $"Bu kodu kullanarak sistemdeki kaydınızı aktive edebilirsiniz.\n" +
    $"Herhangi bir sorunuz olursa bizimle iletişime geçmekten
çekinmeyin.\n\n" +
    $"İyi günler dileriz!";

smtpServer.Port = 587;
smtpServer.Credentials = new NetworkCredential(sistemEmail, emailPassword);
smtpServer.EnableSsl = true;

smtpServer.Send(mail);
}
catch (Exception ex)
{
    MessageBox.Show("E-posta gönderilirken hata oluştu: " + ex.Message);
}
}

// Durum e-postası gönder
private void SendStatusEmail(string adminEmail, string status, string telefonNo, string
aktivasyonKodu)
{
    try
    {
        MailMessage mail = new MailMessage();
        SmtplibClient smtpServer = new SmtplibClient("smtp.gmail.com");

        mail.From = new MailAddress(sistemEmail);
        mail.To.Add(adminEmail);
        mail.Subject = "Kayıt Durum Bildirimi";
        mail.Body = $"Sayın Admin,\n\n" +
            $"Aşağıdaki kayıt için işlem durumu: {status}\n\n" +
            $"Telefon Numarası: {telefonNo}\n" +
            $"Aktivasyon Kodu: {aktivasyonKodu}\n\n" +
            $"Bilgilendirme: \n" +
            $"- Durum 'Giriş Başarılı' ise: Kullanıcı giriş yaptı.\n" +
            $"- Durum 'Giriş Başarısız' ise: Kullanıcı giriş yapamadı, aktivasyon kodu
iptal edildi.\n\n" +

```

\$"İyi çalışmalar!";

```
smtpServer.Port = 587;  
smtpServer.Credentials = new NetworkCredential(sistemEmail, emailPassword);  
smtpServer.EnableSsl = true;
```

```
smtpServer.Send(mail);  
MessageBox.Show($"Durum e-postası gönderildi: {status}");
```

```
}  
catch (Exception ex)  
{
```

```
    MessageBox.Show("Durum e-postası gönderilirken hata oluştu: " +  
ex.Message);  
}  
}
```

// String kesme fonksiyonu

```
private string TruncateString(string value, int maxLength)  
{  
    return string.IsNullOrEmpty(value) ? value : value.Substring(0,  
Math.Min(value.Length, maxLength));  
}
```

```
private void pictureBox2_Click(object sender, EventArgs e)
```

```
{  
    textBox1.Clear();  
    textBox3.Clear();  
}
```

```
}  
}
```

---

## FORM 10: AKTİVASYON EKRANI

The screenshot displays a software interface for the activation section. At the top, a dark blue header bar contains the text 'AKTİVASYON BÖLÜMÜ' in white. To the right of the header, there is a red circular icon with a white person silhouette and the word 'Destek' below it, and a red square button with a white 'X'. Below the header, the main area has a dark blue background. It features an illustration of two people, a woman and a man, in the center. Below the illustration, there are two input fields: 'Telefon no:' followed by a light blue rectangular box, and 'Aktivasyon Kodu:' followed by a light blue rectangular box. At the bottom of the screen, there are two circular icons: a blue one with a white person silhouette and a green one with a white checkmark.

```

using System;
using System.IO;
using System.Net;
using System.Net.Mail;
using System.Net.NetworkInformation;
using System.Text;
using System.Windows.Forms;
using MailKit;
using MailKit.Net.Imap;
using MimeKit;

namespace WinFormsApp8
{
    public partial class Form10 : Form
    {
        private const string Email = "otomasyonveteriner@gmail.com";
        private const string Sifre = "ftkx mebp kdfn ufrj";
        private const string Anahtar = "VetOtomasyon123";
        private readonly string dosyaYolu = Path.Combine(Application.StartupPath,
"activation.txt");
        private readonly string aktifDosya = Path.Combine(Application.StartupPath,
"isActivated.txt");
        private int hataSayisi = 0;

        public Form10()
        {
            InitializeComponent();
            AktivasyonKontrol();
        }

        private void AktivasyonKontrol()
        {
            if (!File.Exists(dosyaYolu)) return;

            string[] veriler = File.ReadAllLines(dosyaYolu);
            if (veriler.Length <= 1) return;

            if (Coz(veriler[1], Anahtar) == MakineKimligi())
            {
                File.WriteAllText(aktifDosya, "Activated");
                Application.Exit(); // Form10 kapanır ve uygulama biter
            }
        }
    }
}

```

```

    }
    else
    {
        MessageBox.Show("Program başka bir bilgisayarda aktif!");
        Application.Exit();
    }
}

```

```

private string MakineKimligi() => MacAdresi();

```

```

private string MacAdresi()
{
    try
    {
        return NetworkInterface.GetAllNetworkInterfaces()
            .FirstOrDefault(ni => ni.OperationalStatus == OperationalStatus.Up)
            ?.GetPhysicalAddress().ToString() ?? "BilinmeyenMAC";
    }
    catch
    {
        return "HataMAC";
    }
}

```

```

private string IpAdresi()
{
    try
    {
        return Dns.GetHostAddresses(Dns.GetHostName())
            .FirstOrDefault(addr => addr.AddressFamily ==
System.Net.Sockets.AddressFamily.InterNetwork)
            ?.ToString() ?? "BilinmeyenIP";
    }
    catch
    {
        return "HataIP";
    }
}

```

```

private string Sifrele(string metin, string anahtar)
{
    StringBuilder sonuc = new StringBuilder();
    for (int i = 0; i < metin.Length; i++)
        sonuc.Append((char)(metin[i] ^ anahtar[i % anahtar.Length]));
    return Convert.ToBase64String(Encoding.UTF8.GetBytes(sonuc.ToString()));
}

```

```

private string Coz(string sifreliMetin, string anahtar)
{
    byte[] baytlar = Convert.FromBase64String(sifreliMetin);
}

```

```

string metin = Encoding.UTF8.GetString(baytlar);
StringBuilder sonuc = new StringBuilder();
for (int i = 0; i < metin.Length; i++)
    sonuc.Append((char)(metin[i] ^ anahtar[i % anahtar.Length]));
return sonuc.ToString();
}

private void pictureBox2_Click(object sender, EventArgs e)
{
    try
    {
        string telefon = textBox3.Text.Trim();
        string kod = textBox1.Text.Trim();

        if (string.IsNullOrEmpty(telefon))
        {
            MessageBox.Show("Telefon numarası giriniz!");
            return;
        }

        if (string.IsNullOrEmpty(kod))
        {
            if (TelefonKoduAldiMi(telefon))
            {
                MessageBox.Show("Bu numara zaten kod aldı. Mevcut kodu kullanın.");
                return;
            }

            string yeniKod = KodUret();
            AktivasyonMailiGonder(yeniKod, telefon);
            MessageBox.Show("Kod e-postanıza gönderildi. Kodu giriniz.");
            return;
        }

        if (MailKontrol(telefon, kod))
        {
            string makineld = MakineKimligi();
            File.WriteAllLines(dosyaYolu, new[] { "Activated", Sifrele(makineld, Anahtar)
        });
            File.WriteAllText(aktifDosya, "Activated");
            BasariMailiGonder(telefon, kod);
            MessageBox.Show("Giriş Başarılı - Uygulama kapanacak. Tekrar açtığınızda
direkt Form1 başlayacak.");
            Application.Exit(); // Aktivasyon sonrası kapanır
        }
        else
        {
            hataSayisi++;
            if (hataSayisi >= 8)
            {

```

```

        MessageBox.Show("8 kez hatalı giriş! Geliştiriciyle iletişime geçin.");
        Hide();
        new Form9().ShowDialog();
        hataSayisi = 0;
    }
    else
    {
        MessageBox.Show($"Hatalı giriş! Kalan hak: {8 - hataSayisi}");
    }
}
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message);
}
}

private bool TelefonKoduAldiMi(string telefon)
{
    using var istemci = new ImapClient();
    try
    {
        istemci.Connect("imap.gmail.com", 993, true);
        istemci.Authenticate(Email, Sifre);
        var gelen = istemci.Inbox;
        gelen.Open(FolderAccess.ReadOnly);
        var mesajlar = gelen.Fetch(0, -1, MessageSummaryItems.Uniquelid);
        foreach (var mesaj in mesajlar)
        {
            var email = gelen.GetMessage(mesaj.Uniquelid);
            if (email.TextBody != null && email.TextBody.Contains(telefon) &&
email.TextBody.Contains("Aktivasyon Kodunuz"))
            {
                istemci.Disconnect(true);
                return true;
            }
        }
        istemci.Disconnect(true);
        return false;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Telefon kontrol hatası: " + ex.Message);
        return false;
    }
}

private string KodUret()
{
    Random rastgele = new Random();

```

```

const string karakterler = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
while (true)
{
    string kod = new string(Enumerable.Repeat(karakterler, 11).Select(s =>
s[rastgele.Next(s.Length)]).ToArray());
    if (kod.Any(char.IsLetter) && kod.Any(char.IsDigit))
        return kod;
}
}

private void AktivasyonMailiGonder(string kod, string telefon)
{
    try
    {
        using var mail = new MailMessage();
        using var smtp = new SmtpClient("smtp.gmail.com")
        {
            Port = 587,
            Credentials = new NetworkCredential(Email, Sifre),
            EnableSsl = true
        };
        mail.From = new MailAddress(Email);
        mail.To.Add(Email);
        mail.Subject = "Aktivasyon Kodunuz";
        mail.Body = $"Telefon: {telefon}\nMAC: {MacAdresi()}\nIP: {IpAdresi()}\nKod:
{kod}";
        smtp.Send(mail);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Mail gönderim hatası: " + ex.Message);
    }
}

private void BasariMailiGonder(string telefon, string kod)
{
    try
    {
        using var mail = new MailMessage();
        using var smtp = new SmtpClient("smtp.gmail.com")
        {
            Port = 587,
            Credentials = new NetworkCredential(Email, Sifre),
            EnableSsl = true
        };
        mail.From = new MailAddress(Email);
        mail.To.Add(Email);
        mail.Subject = "Giriş Başarılı";
        mail.Body = $"Telefon: {telefon}\nMAC: {MacAdresi()}\nIP: {IpAdresi()}\nKod:
{kod}\nKod artık sorulmayacak.";
    }
}

```

```

        smtp.Send(mail);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Başarı maili hatası: " + ex.Message);
    }
}

private bool MailKontrol(string telefon, string kod)
{
    using var istemci = new ImapClient();
    try
    {
        istemci.Connect("imap.gmail.com", 993, true);
        istemci.Authenticate(Email, Sifre);
        var gelen = istemci.Inbox;
        gelen.Open(FolderAccess.ReadOnly);
        var mesajlar = gelen.Fetch(0, -1, MessageSummaryItems.Uniquelid);
        foreach (var mesaj in mesajlar)
        {
            var email = gelen.GetMessage(mesaj.Uniquelid);
            if (email.TextBody != null && email.TextBody.Contains(telefon) &&
email.TextBody.Contains(kod))
            {
                istemci.Disconnect(true);
                return true;
            }
        }
        istemci.Disconnect(true);
        return false;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Mail kontrol hatası: " + ex.Message);
        return false;
    }
}

private void pictureBox3_Click(object sender, EventArgs e) => FormDegistir(new
Form9());
private void button3_Click(object sender, EventArgs e) => Application.Exit();
private void pictureBox1_Click(object sender, EventArgs e) => FormDegistir(new
Form12());

private void FormDegistir(Form form)
{
    Hide();
    form.ShowDialog();
}
}

```



```

public partial class Form1 : Form
{
    public Form1(bool aktif)
    {
        InitializeComponent();
        if (!aktif) // Aktif değilse bir şey yapma (bu zaten olmayacak)
        {
            MessageBox.Show("Hata: Form1 direkt açılmamalı!");
            Application.Exit();
        }
    }
}

```

---

#### PROGRAM.CS

```

using System;
using System.IO;
using System.Windows.Forms;

namespace WinFormsApp8
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            string isActivatedFilePath = Path.Combine(Application.StartupPath,
"isActivated.txt");
            Application.Run(File.Exists(isActivatedFilePath) ? new Form1(true) : new
Form10());
        }
    }
}

```

---

mail adresi : [otomasyonveteriner@gmail.com](mailto:otomasyonveteriner@gmail.com)


uygulama şifresi: vdsu gmlh yfke mcbo


---


AKTİVASYON KAYIT


## AKTİVASYON KAYIT

AKTİVASYON KAYIT

  
AnaSayfa

  
  
Telefon no:  
  

  
Kayıt et

  
Temizle

Adres :

```
using System;

using System.Net;

using System.Net.Mail;

using System.Net.NetworkInformation; // MAC adresi için

using System.Text;

using System.Windows.Forms;

using MailKit.Net.Imap;

using MailKit;

using MimeKit;

namespace WinFormsApp8
{
    public partial class Form12 : Form
    {
        private string sistemEmail = "otomasyonveteriner@gmail.com";

        private string emailPassword = "ftkx mebp kdfn ufrj"; // Gmail uygulama
şifresi

        private string currentTelefonNo; // Onay kontrolü için telefon numarasını
sakla
```

sakla     private string currentAktivasyonKodu; // Onay kontrolü için aktivasyon kodunu

```
public Form12()
```

```
{
```

```
    InitializeComponent();
```

```
}
```

```
private void pictureBox4_Click(object sender, EventArgs e)
```

```
{
```

```
    this.Hide();
```

```
    Form10 form10 = Application.OpenForms["Form10"] as Form10;
```

```
    if (form10 != null)
```

```
        form10.Show();
```

```
    else
```

```
    {
```

```
        form10 = new Form10();
```

```
        form10.Show();
```

```
    }
```

```
}
```

```
private void pictureBox3_Click(object sender, EventArgs e)
```

```
{
```

```
    try
```

```
    {
```

```
        string isimSoyisim = TruncateString(textBox1.Text, 255);
```

```
        string telefonNo = TruncateString(textBox3.Text, 11);
```

```
        string adres = TruncateString(richTextBox1.Text, 255);
```

```
        string macAddress = GetMacAddress(); // MAC adresini al
```

```
        string ipAddress = GetIpAddress(); // IP adresini al
```

```
        if (string.IsNullOrEmpty(telefonNo))
        {
            MessageBox.Show("Lütfen telefon numaranızı giriniz!");
            return;
        }

        // Telefon numarasının daha önce kullanılıp kullanılmadığını kontrol
et

        if (HasPhoneNumberReceivedCode(telefonNo))
        {
            MessageBox.Show("Bu telefon numarası daha önce kullanılmış, başka
bir numara girin!");
            return;
        }

        string aktivasyonKodu = GenerateUniqueActivationCode(macAddress,
ipAddress); // MAC ve IP'ye göre kod üret

        currentTelefonNo = telefonNo; // Onay kontrolü için sakla

        currentAktivasyonKodu = aktivasyonKodu; // Onay kontrolü için sakla

        SendActivationEmail(sistemEmail, aktivasyonKodu, isimSoyisim, adres,
telefonNo, macAddress, ipAddress);

        MessageBox.Show("Kayıt başarıyla eklendi! Aktivasyon kodu e-posta
adresinize gönderildi. Lütfen giriş yapınız.");

        // Kullanıcı aktivasyonunu kontrol et

        if (CheckEmailForActivation(telefonNo, aktivasyonKodu))
        {
            SendStatusEmail(sistemEmail, "Giriş Başarılı", telefonNo,
aktivasyonKodu);
```

```

        textBox1.Clear();

        textBox3.Clear();

        richTextBox1.Clear();

        this.Hide();

        Form10 form10 = Application.OpenForms["Form10"] as Form10;

        if (form10 != null)

            form10.Show();

        else

        {

            form10 = new Form10();

            form10.Show();

        }

    }

    else

    {

        SendStatusEmail(sistemEmail, "Giriş Başarısız - Hatalı Kod",
telefonNo, aktivasyonKodu);

        MessageBox.Show("E-posta ile gönderilen telefon numarası veya
aktivasyon kodu eşleşmedi! Aktivasyon kodu iptal edildi.");

    }

}

catch (Exception ex)

{

    MessageBox.Show("Hata oluştu: " + ex.Message);

}

}

// Telefon numarasının daha önce kod alıp almadığını kontrol et

private bool HasPhoneNumberReceivedCode(string telefonNo)

```

```

{
    try
    {
        using (var client = new ImapClient())
        {
            client.Connect("imap.gmail.com", 993, true);
            client.Authenticate(sistemEmail, emailPassword);

            var inbox = client.Inbox;
            inbox.Open(FolderAccess.ReadOnly);

            var messages = inbox.Fetch(0, -1, MessageSummaryItems.Full |
MessageSummaryItems.UniqueId);

            foreach (var message in messages)
            {
                var email = inbox.GetMessage(message.UniqueId);
                string emailBody = email.TextBody ?? "";

                if (emailBody.Contains(telefonNo) &&
emailBody.Contains("Aktivasyon Kodunuz"))
                {
                    client.Disconnect(true);
                    return true; // Telefon numarası daha önce kod almış
                }
            }

            client.Disconnect(true);
            return false; // Telefon numarası kod almamış
        }
    }
    catch (Exception ex)

```

```

        {
            MessageBox.Show("Telefon numarası kontrolü sırasında hata: " +
ex.Message);

            return false;
        }
    }

    // E-posta ile aktivasyon kodunu kontrol et
    private bool CheckEmailForActivation(string telefonNo, string aktivasyonKodu)
    {
        try
        {
            using (var client = new ImapClient())
            {
                client.Connect("imap.gmail.com", 993, true);

                client.Authenticate(sistemEmail, emailPassword);

                var inbox = client.Inbox;

                inbox.Open(FolderAccess.ReadOnly);

                var messages = inbox.Fetch(0, -1, MessageSummaryItems.Full |
MessageSummaryItems.UniqueId);

                for (int i = Math.Max(0, messages.Count - 5); i < messages.Count;
i++)
                {
                    var message = inbox.GetMessage(messages[i].UniqueId);

                    string emailBody = message.TextBody ?? "";

                    if (emailBody.Contains(telefonNo) &&
emailBody.Contains(aktivasyonKodu))
                    {

                        client.Disconnect(true);

```

```

        return true;
    }
}

    client.Disconnect(true);

    return false;
}
}

catch (Exception ex)
{
    MessageBox.Show("E-posta kontrolü sırasında hata: " + ex.Message);

    return false;
}
}

// MAC adresini alma
private string GetMacAddress()
{
    try
    {
        NetworkInterface[] interfaces =
NetworkInterface.GetAllNetworkInterfaces();

        foreach (NetworkInterface ni in interfaces)
        {
            if (ni.OperationalStatus == OperationalStatus.Up)
            {
                return ni.GetPhysicalAddress().ToString();
            }
        }

        return "UnknownMAC";
    }
}

```



```

    }

    catch (Exception)

    {

        return "ErrorMAC";

    }

}

// IP adresini alma
private string GetIpAddress()
{
    try
    {

        string hostName = Dns.GetHostName();

        IPAddress[] addresses = Dns.GetHostAddresses(hostName);

        foreach (IPAddress address in addresses)
        {

            if (address.AddressFamily ==
System.Net.Sockets.AddressFamily.InterNetwork) // IPv4

            {

                return address.ToString();

            }

        }

        return "UnknownIP";

    }

    catch (Exception)

    {

        return "ErrorIP";

    }

}

```

```

// MAC ve IP'ye göre benzersiz aktivasyon kodu üret

private string GenerateUniqueActivationCode(string macAddress, string
ipAddress)
{
    Random random = new Random();

    const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

    StringBuilder code = new StringBuilder();

    // MAC ve IP'den türetilmiş bir başlangıç ekle

    string macPart = macAddress.Length >= 6 ? macAddress.Substring(0, 6) :
macAddress.PadRight(6, '0');

    string ipPart = ipAddress.Replace(".", "").Length >= 4 ?
ipAddress.Replace(".", "").Substring(0, 4) : ipAddress.Replace(".", "").PadRight(4,
'0');

    code.Append(macPart.Substring(0, 3).ToUpper()); // MAC'in ilk 3 karakteri

    code.Append(ipPart.Substring(0, 2).ToUpper()); // IP'nin ilk 2 karakteri

    // Rastgele 6 karakter ekle

    bool hasLetter = false, hasDigit = false;

    while (code.Length < 11 || !hasLetter || !hasDigit)
    {
        char nextChar = chars[random.Next(chars.Length)];

        code.Append(nextChar);

        if (char.IsLetter(nextChar)) hasLetter = true;

        if (char.IsDigit(nextChar)) hasDigit = true;
    }

    return code.ToString().Substring(0, 11); // 11 karaktere sabitle
}

// Aktivasyon e-postası gönder (MAC ve IP eklendi)

```

```

private void SendActivationEmail(string email, string aktivasyonKodu, string
isimSoyisim, string adres, string telefonNo, string macAddress, string ipAddress)
{
    try
    {
        MailMessage mail = new MailMessage();

        SmtplibClient smtpServer = new SmtplibClient("smtp.gmail.com");

        mail.From = new MailAddress(sistemEmail);

        mail.To.Add(email);

        mail.Subject = "Aktivasyon Kodunuz";

        mail.Body = $"Merhaba,\n\n" +

            $"Veteriner otomasyon sistemine kaydınız başarıyla
alınmıştır. Aşağıda kayıt bilgileriniz ve aktivasyon kodunuz yer almaktadır:\n\n" +

            $"İsim Soyisim: {isimSoyisim}\n" +

            $"Adres: {adres}\n" +

            $"Telefon Numarası: {telefonNo}\n" +

            $"MAC Adresi: {macAddress}\n" +

            $"IP Adresi: {ipAddress}\n" +

            $"Aktivasyon Kodunuz: {aktivasyonKodu}\n\n" +

            $"Bu kodu kullanarak sistemdeki kaydınızı aktive
edebilirsiniz.\n" +

            $"Herhangi bir sorunuz olursa bizimle iletişime geçmekten
çekinmeyin.\n\n" +

            $"İyi günler dileriz!";

        smtpServer.Port = 587;

        smtpServer.Credentials = new NetworkCredential(sistemEmail,
emailPassword);

        smtpServer.EnableSsl = true;

        smtpServer.Send(mail);
    }
}

```

```

    }

    catch (Exception ex)

    {

        MessageBox.Show("E-posta gönderilirken hata oluştu: " + ex.Message);

    }

}

// Durum e-postası gönder

private void SendStatusEmail(string adminEmail, string status, string
telefonNo, string aktivasyonKodu)

{

    try

    {

        MailMessage mail = new MailMessage();

        SmtplibClient smtpServer = new SmtplibClient("smtp.gmail.com");

        mail.From = new MailAddress(sistemEmail);

        mail.To.Add(adminEmail);

        mail.Subject = "Kayıt Durum Bildirimi";

        mail.Body = $"Sayın Admin,\n\n" +

            $"Aşağıdaki kayıt için işlem durumu: {status}\n\n" +

            $"Telefon Numarası: {telefonNo}\n" +

            $"Aktivasyon Kodu: {aktivasyonKodu}\n\n" +

            $"Bilgilendirme: \n" +

            + $"- Durum 'Giriş Başarılı' ise: Kullanıcı giriş yaptı.\n"

            + $"- Durum 'Giriş Başarısız' ise: Kullanıcı giriş yapamadı,
            aktivasyon kodu iptal edildi.\n\n" +

            $"İyi çalışmalar!";

        smtpServer.Port = 587;

```

```

        smtpServer.Credentials = new NetworkCredential(sistemEmail,
emailPassword);

        smtpServer.EnableSsl = true;

        smtpServer.Send(mail);

        MessageBox.Show($"Durum e-postası gönderildi: {status}");
    }

    catch (Exception ex)
    {
        MessageBox.Show("Durum e-postası gönderilirken hata oluştu: " +
ex.Message);
    }
}

// String kesme fonksiyonu
private string TruncateString(string value, int maxLength)
{
    return string.IsNullOrEmpty(value) ? value : value.Substring(0,
Math.Min(value.Length, maxLength));
}

private void pictureBox2_Click(object sender, EventArgs e)
{
    textBox1.Clear();

    textBox3.Clear();
}

private void Form12_Load(object sender, EventArgs e)
{
}
}

```

```
}
```

```
-----  
  
/ Başlık yazı tipi ve rengi  
    dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI",  
11, FontStyle.Bold); // Başlık fontunu 11 yaptım  
    dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.White;  
    dataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.FromArgb(33,  
150, 243);  
    dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =  
DataGridViewContentAlignment.MiddleCenter;  
  
    // Kenarlıklar  
    dataGridView1.EnableHeadersVisualStyles = false;  
  
    // Satır veri yazı tipi ve stilleri  
    dataGridView1.DefaultCellStyle.Font = new Font("Segoe UI Semibold", 10,  
FontStyle.Regular); // Daha tok bir font  
    dataGridView1.DefaultCellStyle.ForeColor = Color.Black;  
    dataGridView1.DefaultCellStyle.BackColor = Color.White;  
    dataGridView1.DefaultCellStyle.SelectionBackColor = Color.LightBlue;  
    dataGridView1.DefaultCellStyle.SelectionForeColor = Color.Black;  
    dataGridView1.DefaultCellStyle.Alignment =  
DataGridViewContentAlignment.MiddleLeft; // İstersen ortalı da yapabilirim  
  
    // Sütun genişliklerini ayarla  
    //dataGridView1.Columns["id"].Width = 100;  
    //dataGridView1.Columns["email"].Width = 250;  
    //dataGridView1.Columns["password"].Width = 150;  
  
    dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.None;  
  
    // Başlık yazı tipi ve rengi  
    dataGridView2.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI",  
11, FontStyle.Bold); // Başlık fontunu 11 yaptım  
    dataGridView2.ColumnHeadersDefaultCellStyle.ForeColor = Color.White;  
    dataGridView2.ColumnHeadersDefaultCellStyle.BackColor = Color.FromArgb(33,  
150, 243);  
    dataGridView2.ColumnHeadersDefaultCellStyle.Alignment =  
DataGridViewContentAlignment.MiddleCenter;  
  
    // Kenarlıklar  
    dataGridView2.EnableHeadersVisualStyles = false;  
  
    // Satır veri yazı tipi ve stilleri  
    dataGridView2.DefaultCellStyle.Font = new Font("Segoe UI Semibold", 10,  
FontStyle.Regular); // Daha tok bir font  
    dataGridView2.DefaultCellStyle.ForeColor = Color.Black;  
    dataGridView2.DefaultCellStyle.BackColor = Color.White;  
    dataGridView2.DefaultCellStyle.SelectionBackColor = Color.LightBlue;  
    dataGridView2.DefaultCellStyle.SelectionForeColor = Color.Black;  
    dataGridView2.DefaultCellStyle.Alignment =  
DataGridViewContentAlignment.MiddleLeft; // İstersen ortalı da yapabilirim  
  
    // Sütun genişliklerini ayarla  
    //dataGridView2.Columns["id"].Width = 100;  
    //dataGridView2.Columns["email"].Width = 250;  
    //dataGridView2.Columns["password"].Width = 150;  
  
-----
```

not: besirasyonda veri tabanı sıkıntısı yaşadığım için sqlite3 e geçtim çok kolaymış mssql'e göre

\*\*\*\*\*

#### FORM 16 BESİRASYOM

```
using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net.Mail;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

namespace WinFormsApp8

{
```

```

public partial class Form16 : Form
{
    private readonly string connectionString = @"Data
Source=C:\Users\ridva\Desktop\SQLİTE3_VETERİNEROTOMASYON\veteriner_otomasyon.db;Version=3;"
;

    private List<Yem> mevcutYemler = new List<Yem>();

    private string? sonHesaplamaMesaji;

    private string? sonOneriPrefix;

    private double eksikHP = 0, eksikP = 0, eksikKM = 0, eksikME = 0, eksikCa = 0;

    private double gunlukYemMiktari = 0;

    private Panel label6Panel;

    private bool isSuggestionFormOpen = false; // Öneri formunun açık olup olmadığını
    takip etmek için

    public Form16()
    {
        InitializeComponent();

        button1.Click += new EventHandler(button1_Click);

        button2.Click += new EventHandler(button2_Click);

        button3.Click += new EventHandler(button3_Click);

        button4.Click += new EventHandler(button4_Click);

        textBox3.TextChanged += new EventHandler(textBox3_TextChanged);

        ConfigureAutoCompleteForTextBox3();

        SetupLabel6Panel();

        this.AutoScroll = true;

        label6.Text = "Yem stoğu boş. Lütfen yem ekleyin.";
    }

    private void SetupLabel6Panel()
    {

```



```

        Point label6Location = label6.Location;

        Size label6Size = new Size(400, 300);

        label6Panel = new Panel
        {
            Location = label6Location,

            Size = label6Size,

            AutoScroll = true,

            BorderStyle = BorderStyle.FixedSingle
        };

        label6.Location = new Point(0, 0);

        label6.AutoSize = true;

        label6.MaximumSize = new Size(label6Panel.ClientSize.Width -
SystemInformation.VerticalScrollBarWidth, 0);

        label6.BackColor = Color.LightGray;

        label6.ForeColor = Color.DarkBlue;

        label6.Font = new Font("Arial", 10, FontStyle.Bold);

        label6.Padding = new Padding(5);

        label6.TextAlign = ContentAlignment.TopLeft;

        label6Panel.Controls.Add(label6);

        this.Controls.Remove(label6);

        this.Controls.Add(label6Panel);
    }

    public class Yem
    {
        public required string Ad { get; set; }

        public double KM { get; set; }
    }

```

```
        public double ME { get; set; }

        public double HP { get; set; }

        public double Ca { get; set; }

        public double P { get; set; }

        public double Miktar { get; set; }

    }
```

```
public class ihtiyaç

{

    public double KM { get; set; }

    public double ME { get; set; }

    public double HP { get; set; }

    public double Ca { get; set; }

    public double P { get; set; }

}
```

```
private string? GetHayvanTuru()

{

    if (radioButton1.Checked) return "YERLİ İRK ERKEK DANA".Trim().ToUpper();

    if (radioButton2.Checked) return "KÜLTÜR İRKİ ERKEK DANA".Trim().ToUpper();

    if (radioButton3.Checked) return "YERLİ İRK DIŞI DANA".Trim().ToUpper();

    if (radioButton4.Checked) return "KÜLTÜR İRKİ DIŞI DANA".Trim().ToUpper();

    return null;

}
```

```
private async Task<bool> TestConnectionAsync()

{

    try

    {

        using (var conn = new SQLiteConnection(connectionString))
```

```

        {

            await conn.OpenAsync();

            return true;

        }

    }

    catch (Exception ex)

    {

        MessageBox.Show($"Veritabanı bağlantı testi başarısız: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return false;

    }

}

```

```

private async Task<List<Yem>> YemiYukleAsync()

{

    var yemler = new List<Yem>();

    using (var conn = new SQLiteConnection(connectionString))

    {

        try

        {

            await conn.OpenAsync();

            string query = "SELECT Yemler, KM_Yuzde, ME_Mcal_kg, HP_Yuzde,
Ca_Yuzde, P_Yuzde FROM Yem";

            using (var cmd = new SQLiteCommand(query, conn))

            using (var reader = await cmd.ExecuteReaderAsync())

            {

                while (await reader.ReadAsync())

                {

                    string? yemAdi = reader["Yemler"] as string;

                    if (string.IsNullOrEmpty(yemAdi)) continue;

```

```

        yemler.Add(new Yem
        {
            Ad = yemAdi,

            KM = reader["KM_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["KM_Yuzde"]),

            ME = reader["ME_Mcal_kg"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["ME_Mcal_kg"]),

            HP = reader["HP_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["HP_Yuzde"]),

            Ca = reader["Ca_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["Ca_Yuzde"]),

            P = reader["P_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["P_Yuzde"]),

            Miktar = 0
        });
    }

}

}

catch (Exception ex)

{

    MessageBox.Show($"Yem verileri yüklenirken hata: {ex.Message}", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

return yemler;

}

private async Task<List<Yem>> GetYemSuggestionsAsync(string prefix)

{

    var suggestions = new List<Yem>();

    using (var conn = new SQLiteConnection(connectionString))

    {

        try

```

```

    {

        await conn.OpenAsync();

        string query = "SELECT Yemler, KM_Yuzde, ME_Mcal_kg, HP_Yuzde,
Ca_Yuzde, P_Yuzde FROM Yem WHERE Yemler LIKE @Prefix || '%';

        using (var cmd = new SQLiteCommand(query, conn))

        {

            cmd.Parameters.AddWithValue("@Prefix", prefix);

            using (var reader = await cmd.ExecuteReaderAsync())

            {

                while (await reader.ReadAsync())

                {

                    string? yemAdi = reader["Yemler"] as string;

                    if (string.IsNullOrEmpty(yemAdi)) continue;

                    suggestions.Add(new Yem

                    {

                        Ad = yemAdi,

                        KM = reader["KM_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["KM_Yuzde"]),

                        ME = reader["ME_Mcal_kg"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["ME_Mcal_kg"]),

                        HP = reader["HP_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["HP_Yuzde"]),

                        Ca = reader["Ca_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["Ca_Yuzde"]),

                        P = reader["P_Yuzde"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["P_Yuzde"]),

                        Miktar = 0

                    });

                }

            }

        }

    }

```

```

        catch (Exception ex)
        {
            MessageBox.Show($"Yem önerileri yüklenirken hata: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    return suggestions;
}

private async void ConfigureAutoCompleteForTextBox3()
{
    try
    {
        textBox3.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
        textBox3.AutoCompleteSource = AutoCompleteSource.CustomSource;

        var suggestions = await GetYemSuggestionsAsync("");

        AutoCompleteStringCollection autoCompleteCollection = new
AutoCompleteStringCollection();

        autoCompleteCollection.AddRange(suggestions.Select(y => y.Ad).ToArray());

        textBox3.AutoCompleteCustomSource = autoCompleteCollection;
    }

    catch (Exception ex)
    {
        MessageBox.Show($"Otomatik tamamlama yüklenirken hata: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private async void textBox3_TextChanged(object? sender, EventArgs e)
{

```

```

try
{
    string input = textBox3.Text?.Trim() ?? string.Empty;

    if (input.Length < 2) return;

    string prefix = input.Substring(0, Math.Min(2, input.Length));

    if (prefix == sonOneriPrefix || isSuggestionFormOpen) return;

    sonOneriPrefix = prefix;

    isSuggestionFormOpen = true; // Form açıldığını işaretle

    var suggestions = await GetYemSuggestionsAsync(prefix);

    if (!suggestions.Any())
    {
        MessageBox.Show($"İlk 2 harfi '{prefix}' olan yem bulunamadı.", "Yem Önerileri", MessageBoxButtons.OK, MessageBoxIcon.Information);

        sonOneriPrefix = null;

        isSuggestionFormOpen = false;

        return;
    }

    using (var suggestionForm = new Form())
    {
        suggestionForm.Text = "Yem Önerileri";

        suggestionForm.Size = new Size(600, 300);

        suggestionForm.StartPosition = FormStartPosition.CenterParent;

        suggestionForm.FormBorderStyle = FormBorderStyle.FixedDialog;

        suggestionForm.MaximizeBox = false;

        suggestionForm.MinimizeBox = false;
    }
}

```

```

DataGridView dataGridView = new DataGridView
{
    Dock = DockStyle.Fill,
    AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
    ReadOnly = true,
    SelectionMode = DataGridViewSelectionMode.FullRowSelect,
    AllowUserToAddRows = false,
    RowHeadersVisible = false
};

dataGridView.Columns.Add("Ad", "Yem Adı");
dataGridView.Columns.Add("KM", "KM (%)");
dataGridView.Columns.Add("ME", "ME (Mcal/kg)");
dataGridView.Columns.Add("HP", "HP (%)");
dataGridView.Columns.Add("Ca", "Ca (%)");
dataGridView.Columns.Add("P", "P (%)");

foreach (var suggestion in suggestions)
{
    dataGridView.Rows.Add(suggestion.Ad, suggestion.KM, suggestion.ME,
suggestion.HP, suggestion.Ca, suggestion.P);
}

dataGridView.CellClick += (s, args) =>
{
    if (args.RowIndex >= 0)
    {
        textBox3.Text =
dataGridView.Rows[args.RowIndex].Cells["Ad"].Value?.ToString() ?? string.Empty;

        suggestionForm.Close();
    }
}

```



```

};

suggestionForm.Controls.Add(dataGridView);

suggestionForm.FormClosed += (s, args) =>
{
    isSuggestionFormOpen = false; // Form kapandığında durumu sıfırla
    sonOneriPrefix = null;
};

suggestionForm.ShowDialog();
}
}

catch (Exception ex)
{
    MessageBox.Show($"Yem önerisi sırasında hata: {ex.Message}", "Hata",
    MessageBoxButtons.OK, MessageBoxIcon.Error);

    sonOneriPrefix = null;

    isSuggestionFormOpen = false;
}
}

private async Task<Ihtiyac?> IhtiyaciYukleAsync(string hayvanTuru, double agirlik,
double gunlukArtis)
{
    using (var conn = new SQLiteConnection(connectionString))
    {
        try
        {
            await conn.OpenAsync();

```

```

        string query = "SELECT CA_kg, GCAA_g FROM Hayvan_Turleri WHERE
UPPER(Hayvan_Turu) = UPPER(@HayvanTuru)";

        List<double> agirliklar = new List<double>();

        List<double> gunlukArtislar = new List<double>();

        using (var cmd = new SQLiteCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@HayvanTuru", hayvanTuru);

            using (var reader = await cmd.ExecuteReaderAsync())
            {
                while (await reader.ReadAsync())
                {
                    if (reader["CA_kg"] != DBNull.Value)
                        agirliklar.Add(Convert.ToDouble(reader["CA_kg"]));

                    if (reader["GCAA_g"] != DBNull.Value)
                        gunlukArtislar.Add(Convert.ToDouble(reader["GCAA_g"]));
                }
            }
        }

        if (!agirliklar.Any() || !gunlukArtislar.Any())
        {
            DialogResult result = MessageBox.Show(
                $"Hayvan türü: {hayvanTuru} için veritabanında kayıt
bulunamadı! Varsayılan değerler kullanılacak.\n" +
                "Devam etmek istiyor musunuz?",
                "Uyarı", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

            if (result == DialogResult.No)
            {
                return null;
            }
        }
    }
}

```

```

    }

    return new ihtiyaç
    {
        KM = 10,
        ME = 25,
        HP = 500,
        Ca = 40,
        P = 20
    };
}

double enYakinAgirlik = agirliklar.OrderBy(x => Math.Abs(x -
agirlik)).First();

double enYakinGunlukArtis = gunlukArtislar.OrderBy(x => Math.Abs(x -
gunlukArtis)).First();

if (enYakinAgirlik != agirlik || enYakinGunlukArtis != gunlukArtis)
{
    MessageBox.Show(
        $"Girilen ağırlık ({agirlik} kg) ve günlük artış ({gunlukArtis}
kg) değerleri yuvarlandı.\n" +
        $"Kullanılan değerler: Ağırlık = {enYakinAgirlik} kg, Günlük
Artış = {enYakinGunlukArtis} kg.",
        "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

string ihtiyaçQuery = "SELECT KM_kg, ME_Mcal, HP_g, Ca_g, P_g FROM
Hayvan_Turleri " +
        "WHERE UPPER(Hayvan_Turu) = UPPER(@HayvanTuru) AND
CA_kg = @Agirlik AND GCAA_g = @GunlukArtis";

using (var cmd = new SQLiteCommand(ihtiyaçQuery, conn))

```

```

{

    cmd.Parameters.AddWithValue("@HayvanTuru", hayvanTuru);

    cmd.Parameters.AddWithValue("@Agirlik", enYakinAgirlik);

    cmd.Parameters.AddWithValue("@GunlukArtis", enYakinGunlukArtis);


    using (var reader = await cmd.ExecuteReaderAsync())
    {

        if (await reader.ReadAsync())

        {

            return new ihtiyac

            {

                KM = reader["KM_kg"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["KM_kg"]),

                ME = reader["ME_Mcal"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["ME_Mcal"]),

                HP = reader["HP_g"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["HP_g"]),

                Ca = reader["Ca_g"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["Ca_g"]),

                P = reader["P_g"] == DBNull.Value ? 0 :
Convert.ToDouble(reader["P_g"])

            };

        }

        else

        {

            MessageBox.Show(

                $"Hayvan türü: {hayvanTuru} için yuvarlanmış değerlerle
(Ağırlık: {enYakinAgirlik} kg, Günlük Artış: {enYakinGunlukArtis} kg) kayıt bulunamadı!\n"
+

                "Varsayılan değerler kullanılacak.",

                "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

            return new ihtiyac

            {

                KM = 10,

```

```

        ME = 25,

        HP = 500,

        Ca = 40,

        P = 20

    };

}

}

}

}

    catch (Exception ex)

    {

        MessageBox.Show($"İhtiyaç verileri yüklenirken hata: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return null;

    }

}

}

```

```

private Dictionary<string, double> YemDagit(List<Yem> yemler, ihtiyac ihtiyac,
double gunlukYemMiktari)

{

    var sonuc = new Dictionary<string, double>();

    double kalanKM = ihtiyac.KM;

    double kalanME = ihtiyac.ME;

    double kalanHP = ihtiyac.HP;

    double kalanCa = ihtiyac.Ca;

    double kalanP = ihtiyac.P;

    double toplamMiktar = 0;

    var uygunYemler = yemler.Where(y => y.Miktar > 0).ToList();

    if (!uygunYemler.Any())

```

```

    {
        return sonuc;
    }

    double toplamPuan = 0;

    var puanlar = new Dictionary<string, double>();

    foreach (var yem in uygunYemler)
    {
        double puan = 0;

        ihtiyac.HP);
        if (kalanHP > 0 && yem.HP > 0) puan += (yem.HP / 100) * (kalanHP /

        if (kalanP > 0 && yem.P > 0) puan += (yem.P / 100) * (kalanP / ihtiyac.P);

        ihtiyac.KM);
        if (kalanKM > 0 && yem.KM > 0) puan += (yem.KM / 100) * (kalanKM /

        if (kalanME > 0 && yem.ME > 0) puan += yem.ME * (kalanME / ihtiyac.ME);

        ihtiyac.Ca);
        if (kalanCa > 0 && yem.Ca > 0) puan += (yem.Ca / 100) * (kalanCa /

        puanlar[yem.Ad] = puan;

        toplamPuan += puan;
    }

    if (toplamPuan <= 0)
    {
        double toplamMevcutMiktar = uygunYemler.Sum(y => y.Miktar);

        if (toplamMevcutMiktar <= 0) return sonuc;

        foreach (var yem in uygunYemler)
        {
            double oran = yem.Miktar / toplamMevcutMiktar;

```

```

double kullanilacakMiktar = Math.Min(yem.Miktar, gunlukYemMiktari *
oran);

if (kullanilacakMiktar > 0)
{
    sonuc[yem.Ad] = kullanilacakMiktar;
    toplamMiktar += kullanilacakMiktar;

    kalanKM -= kullanilacakMiktar * (yem.KM / 100);
    kalanME -= kullanilacakMiktar * yem.ME;
    kalanHP -= (kullanilacakMiktar * (yem.HP / 100)) * 1000;
    kalanCa -= (kullanilacakMiktar * (yem.Ca / 100)) * 1000;
    kalanP -= (kullanilacakMiktar * (yem.P / 100)) * 1000;
}
}
else
{
    foreach (var yem in uygunYemler)
    {
        double puan = puanlar[yem.Ad];
        if (puan <= 0) continue;

        double oran = puan / toplamPuan;
        double kullanilacakMiktar = Math.Min(yem.Miktar, gunlukYemMiktari *
oran);

        if (kullanilacakMiktar > 0)
        {
            sonuc[yem.Ad] = kullanilacakMiktar;
            toplamMiktar += kullanilacakMiktar;

            kalanKM -= kullanilacakMiktar * (yem.KM / 100);

```

```

        kalanME -= kullanilacakMiktar * yem.ME;

        kalanHP -= (kullanilacakMiktar * (yem.HP / 100)) * 1000;

        kalanCa -= (kullanilacakMiktar * (yem.Ca / 100)) * 1000;

        kalanP -= (kullanilacakMiktar * (yem.P / 100)) * 1000;

    }

}

if (toplamMiktar < gunlukYemMiktari)

{

    double kalanMiktar = gunlukYemMiktari - toplamMiktar;

    var kalanYemler = uygunYemler.Where(y => y.Miktar >
sonuc.GetValueOrDefault(y.Ad, 0))

                                .OrderByDescending(y => puanlar[y.Ad])

                                .ToList();

    foreach (var yem in kalanYemler)

    {

        if (kalanMiktar <= 0) break;

        double mevcutKullanilmis = sonuc.GetValueOrDefault(yem.Ad, 0);

        double ekMiktar = Math.Min(yem.Miktar - mevcutKullanilmis,
kalanMiktar);

        if (ekMiktar > 0)

        {

            sonuc[yem.Ad] = mevcutKullanilmis + ekMiktar;

            toplamMiktar += ekMiktar;

            kalanMiktar -= ekMiktar;

            kalanKM -= ekMiktar * (yem.KM / 100);

            kalanME -= ekMiktar * yem.ME;

            kalanHP -= (ekMiktar * (yem.HP / 100)) * 1000;

```



```

        kalanCa -= (ekMiktar * (yem.Ca / 100)) * 1000;

        kalanP -= (ekMiktar * (yem.P / 100)) * 1000;

    }

}

}

}

    eksikKM = Math.Max(0, kalanKM);
    eksikME = Math.Max(0, kalanME);
    eksikHP = Math.Max(0, kalanHP);
    eksikCa = Math.Max(0, kalanCa);
    eksikP = Math.Max(0, kalanP);

    return sonuc;
}

private string SayiyiYaziyaCevir(double sayi, bool isGram = false)
{
    if (sayi == 0) return "sıfır";

    string[] birler = { "", "bir", "iki", "üç", "dört", "beş", "altı", "yedi",
"sekiz", "dokuz" };
    string[] onlar = { "", "on", "yirmi", "otuz", "kırk", "elli", "altmış",
"yetmiş", "seksen", "doksan" };
    string[] binler = { "", "bin", "milyon", "milyar" };

    string birim = isGram ? "gram" : "kilogram";

    long tamKisim = (long)sayi;

    double ondalikKisim = sayi - tamKisim;

    string sonuc = "";

```

```

if (tamKisim == 0 && ondalikKisim == 0)
{
    return $"sıfır {birim}";
}

int grupSayisi = 0;
while (tamKisim > 0)
{
    int grup = (int)(tamKisim % 1000);
    tamKisim /= 1000;

    string grupYazi = "";
    if (grup > 0)
    {
        if (grup >= 100)
        {
            int yuzler = grup / 100;
            if (yuzler > 1) grupYazi += birler[yuzler] + " yüz ";
            else grupYazi += "yüz ";
            grup %= 100;
        }

        if (grup >= 10)
        {
            grupYazi += onlar[grup / 10] + " ";
            grup %= 10;
        }

        if (grup > 0)
        {

```

```

        grupYazi += birler[grup] + " ";
    }

    grupYazi = grupYazi.Trim();

    if (grupSayisi > 0 && grup == 1 && binler[grupSayisi] == "bin")
    {
        grupYazi = "";
    }

    if (!string.IsNullOrEmpty(grupYazi))
    {
        grupYazi += " " + binler[grupSayisi];
    }

    sonuc = grupYazi.Trim() + " " + sonuc;
}

grupSayisi++;
}

if (ondalikKisim > 0)
{
    int ondalikTam = (int)(ondalikKisim * 100);
    string ondalikYazi = "";
    if (ondalikTam >= 10)
    {
        ondalikYazi += onlar[ondalikTam / 10] + " ";
        ondalikTam %= 10;
    }
}

```

```

        if (ondalikTam > 0)
        {
            ondalikYazi += birler[ondalikTam] + " ";
        }

        if (!string.IsNullOrEmpty(sonuc))
        {
            sonuc += " virgöl " + ondalikYazi.Trim();
        }
        else
        {
            sonuc = "sıfır virgöl " + ondalikYazi.Trim();
        }
    }

    return $"{sonuc.Trim()} {birim}";
}

private async void button1_Click(object? sender, EventArgs e)
{
    try
    {
        if (!await TestConnectionAsync())
        {
            return;
        }

        string? hayvanTuru = GetHayvanTuru();

        if (string.IsNullOrEmpty(hayvanTuru))
        {

```

```
        MessageBox.Show("Lütfen bir hayvan türü seçin!", "Hata",  
        MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        return;
```

```
    }
```

```
        if (!double.TryParse(textBox1.Text, out double agirlik) ||  
        !double.TryParse(textBox2.Text, out double aylıkArtis))
```

```
        {
```

```
            MessageBox.Show("Ağırlık ve aylık kilo artışı sayısal değer  
            olmalıdır!", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            return;
```

```
        }
```

```
        double günlükArtis = aylıkArtis / 30;
```

```
        günlükYemMiktari = günlükArtis * 10;
```

```
        var ihtiyac = await IhtiyaciYukleAsync(hayvanTuru, agirlik, günlükArtis);
```

```
        if (ihtiyac == null)
```

```
        {
```

```
            return;
```

```
        }
```

```
        if (!mevcutYemler.Any())
```

```
        {
```

```
            MessageBox.Show("Hesaplama yapmak için en az bir yem ekleyin!", "Hata",  
            MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            return;
```

```
        }
```

```
        var hesaplamaYemler = mevcutYemler.Where(y => y.Miktar > 0).ToList();
```

```
        var yemDagilim = YemDagit(hesaplamaYemler, ihtiyac, günlükYemMiktari);
```

```

        if (yemDagilim.Count == 0)
        {
            MessageBox.Show("İhtiyacı karşılayacak yeterli yem bulunamadı! Daha fazla yem eklemeyi deneyin.",
                "Hesaplama Sonucu", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);

            sonHesaplamaMesaji = null;

            return;
        }

        double toplamKullanilacakYem = yemDagilim.Sum(x => x.Value);

        StringBuilder finalMesaj = new StringBuilder();

        finalMesaj.AppendLine("=====");
        finalMesaj.AppendLine("          BESLEME PROGRAMI RAPORU          ");
        finalMesaj.AppendLine("=====");
        finalMesaj.AppendLine($"Rapor Tarihi: {DateTime.Now:dd.MM.yyyy HH:mm:ss}");
        finalMesaj.AppendLine($"Hayvan Türü: {hayvanTuru}");
        finalMesaj.AppendLine($"Ağırlık: {agirlik:F2} kg ({SayiyiYaziyaCevir(agirlik)})");
        finalMesaj.AppendLine($"Aylık Kilo Artışı: {aylikArtis:F2} kg ({SayiyiYaziyaCevir(aylikArtis)}) (Günlük: {gunlukArtis:F2} kg)");
        finalMesaj.AppendLine($"Günlük Toplam Yem Miktarı: {gunlukYemMiktari:F2} kg ({SayiyiYaziyaCevir(gunlukYemMiktari)})");
        finalMesaj.AppendLine();

        finalMesaj.AppendLine("-----");
        finalMesaj.AppendLine("Günlük Yem Dağılımı");
        finalMesaj.AppendLine("-----");
        finalMesaj.AppendLine("Yem Adı".PadRight(20) + "Miktar".PadRight(20) +
"Okunuş");
        finalMesaj.AppendLine(new string('-', 60));
        foreach (var item in yemDagilim)
        {

```

```

        double gunlukYem = item.Value;

        string miktarStr = gunlukYem >= 1 ? $"{gunlukYem:F2} kg" : $"{gunlukYem
* 1000:F2} g";

        string okunus = gunlukYem >= 1 ? SayiyiYaziyaCevir(gunlukYem) :
SayiyiYaziyaCevir(gunlukYem * 1000, true);

finalMesaj.AppendLine($"{item.Key.PadRight(20)}{miktarStr.PadRight(20)}{okunus}");

    }

    double toplamKM = 0, toplamME = 0, toplamHP = 0, toplamCa = 0, toplamP = 0;

    foreach (var item in yemDagilim)
    {
        var yem = hesaplamaYemler.FirstOrDefault(y => y.Ad == item.Key);

        if (yem == null) continue;

        double miktar = item.Value;

        toplamKM += miktar * (yem.KM / 100);

        toplamME += miktar * yem.ME;

        toplamHP += (miktar * (yem.HP / 100)) * 1000;

        toplamCa += (miktar * (yem.Ca / 100)) * 1000;

        toplamP += (miktar * (yem.P / 100)) * 1000;

    }

    double kmYuzde = toplamKullanilacakYem > 0 ? (toplamKM /
toplamKullanilacakYem) * 100 : 0;

    double meMcalKg = toplamKullanilacakYem > 0 ? toplamME /
toplamKullanilacakYem : 0;

    double hpYuzde = toplamKullanilacakYem > 0 ? (toplamHP / 1000 /
toplamKullanilacakYem) * 100 : 0;

    double caYuzde = toplamKullanilacakYem > 0 ? (toplamCa / 1000 /
toplamKullanilacakYem) * 100 : 0;

    double pYuzde = toplamKullanilacakYem > 0 ? (toplamP / 1000 /
toplamKullanilacakYem) * 100 : 0;

```

```

finalMesaj.AppendLine();

finalMesaj.AppendLine("-----");

finalMesaj.AppendLine("Besin Yüzdeleri");

finalMesaj.AppendLine("-----");

finalMesaj.AppendLine($"KM (%): {kmYuzde:F2}");

finalMesaj.AppendLine($"ME (Mcal/kg): {meMcalKg:F2}");

finalMesaj.AppendLine($"HP (%): {hpYuzde:F2}");

finalMesaj.AppendLine($"Ca (%): {caYuzde:F2}");

finalMesaj.AppendLine($"P (%): {pYuzde:F2}");


finalMesaj.AppendLine();

finalMesaj.AppendLine("-----");

finalMesaj.AppendLine("Besin İhtiyaçları ve Karşılanma Durumu");

finalMesaj.AppendLine("-----");

finalMesaj.AppendLine("Besin".PadRight(10) + "İhtiyaç".PadRight(15) +
"Kullanılan".PadRight(15) + "Kalan".PadRight(15) + "Durum");

finalMesaj.AppendLine(new string('-', 60));

finalMesaj.AppendLine($"KM (kg)".PadRight(10) +
${ihtiyac.KM:F2}".PadRight(15) + ${toplamKM:F2}".PadRight(15) +
${eksikKM:F2}".PadRight(15) + (eksikKM > 0 ? "Eksik" : "Karşılandı"));

finalMesaj.AppendLine($"ME (Mcal)".PadRight(10) +
${ihtiyac.ME:F2}".PadRight(15) + ${toplamME:F2}".PadRight(15) +
${eksikME:F2}".PadRight(15) + (eksikME > 0 ? "Eksik" : "Karşılandı"));

finalMesaj.AppendLine($"HP (g)".PadRight(10) +
${ihtiyac.HP:F2}".PadRight(15) + ${toplamHP:F2}".PadRight(15) +
${eksikHP:F2}".PadRight(15) + (eksikHP > 0 ? "Eksik" : "Karşılandı"));

finalMesaj.AppendLine($"Ca (g)".PadRight(10) +
${ihtiyac.Ca:F2}".PadRight(15) + ${toplamCa:F2}".PadRight(15) +
${eksikCa:F2}".PadRight(15) + (eksikCa > 0 ? "Eksik" : "Karşılandı"));

finalMesaj.AppendLine($"P (g)".PadRight(10) +
${ihtiyac.P:F2}".PadRight(15) + ${toplamP:F2}".PadRight(15) + ${eksikP:F2}".PadRight(15)
+ (eksikP > 0 ? "Eksik" : "Karşılandı"));


if (eksikKM > 0 || eksikME > 0 || eksikHP > 0 || eksikCa > 0 || eksikP > 0)
{

    finalMesaj.AppendLine();

```



```

finalMesaj.AppendLine("-----");

finalMesaj.AppendLine("Eksik Besin Önerileri");

finalMesaj.AppendLine("-----");

finalMesaj.AppendLine("Yem Adı".PadRight(20) + "Besin".PadRight(15) +
"Miktar".PadRight(15) + "Okunuş");

finalMesaj.AppendLine(new string('-', 60));

if (eksikKM > 0)
{
    var bestYemKM = hesaplamaYemler.Where(y => y.KM >
0).OrderByDescending(y => y.KM).FirstOrDefault();

    if (bestYemKM != null)
    {
        double eksikMiktar = eksikKM / (bestYemKM.KM / 100);

        string miktarStr = eksikMiktar >= 1 ? $"{eksikMiktar:F2} kg" :
$"{(eksikMiktar * 1000):F2} g";

        string okunus = eksikMiktar >= 1 ?
SayiyiYaziyaCevir(eksikMiktar) : SayiyiYaziyaCevir(eksikMiktar * 1000, true);

finalMesaj.AppendLine($"{bestYemKM.Ad.PadRight(20)}KM".PadRight(15) +
$"{miktarStr.PadRight(15)}{okunus}");

    }
}

if (eksikME > 0)
{
    var bestYemME = hesaplamaYemler.Where(y => y.ME >
0).OrderByDescending(y => y.ME).FirstOrDefault();

    if (bestYemME != null)
    {
        double eksikMiktar = eksikME / bestYemME.ME;

        string miktarStr = eksikMiktar >= 1 ? $"{eksikMiktar:F2} kg" :
$"{(eksikMiktar * 1000):F2} g";

        string okunus = eksikMiktar >= 1 ?
SayiyiYaziyaCevir(eksikMiktar) : SayiyiYaziyaCevir(eksikMiktar * 1000, true);

```

```

finalMesaj.AppendLine($"{bestYemME.Ad.PadRight(20)}ME".PadRight(15) +
    $"{miktarStr.PadRight(15)}{okunus}");

    }

    }

    if (eksikHP > 0)

    {

        var bestYemHP = hesaplamaYemler.Where(y => y.HP >
0).OrderByDescending(y => y.HP).FirstOrDefault();

        if (bestYemHP != null)

        {

            double eksikMiktar = eksikHP / (bestYemHP.HP * 10);

            string miktarStr = eksikMiktar >= 1 ? $"{eksikMiktar:F2} kg" :
$"{(eksikMiktar * 1000):F2} g";

            string okunus = eksikMiktar >= 1 ?
SayiyiYaziyaCevir(eksikMiktar) : SayiyiYaziyaCevir(eksikMiktar * 1000, true);

finalMesaj.AppendLine($"{bestYemHP.Ad.PadRight(20)}HP".PadRight(15) +
    $"{miktarStr.PadRight(15)}{okunus}");

        }

    }

    if (eksikCa > 0)

    {

        var bestYemCa = hesaplamaYemler.Where(y => y.Ca >
0).OrderByDescending(y => y.Ca).FirstOrDefault();

        if (bestYemCa != null)

        {

            double eksikMiktar = eksikCa / (bestYemCa.Ca * 10);

            string miktarStr = eksikMiktar >= 1 ? $"{eksikMiktar:F2} kg" :
$"{(eksikMiktar * 1000):F2} g";

            string okunus = eksikMiktar >= 1 ?
SayiyiYaziyaCevir(eksikMiktar) : SayiyiYaziyaCevir(eksikMiktar * 1000, true);

finalMesaj.AppendLine($"{bestYemCa.Ad.PadRight(20)}Ca".PadRight(15) +
    $"{miktarStr.PadRight(15)}{okunus}");

        }

    }

```

```

        if (eksikP > 0)
        {
            var bestYemP = hesaplamaYemler.Where(y => y.P >
0).OrderByDescending(y => y.P).FirstOrDefault();

            if (bestYemP != null)
            {
                double eksikMiktar = eksikP / (bestYemP.P * 10);

                string miktarStr = eksikMiktar >= 1 ? $"{eksikMiktar:F2} kg" :
$"{(eksikMiktar * 1000):F2} g";

                string okunus = eksikMiktar >= 1 ?
SayiyiYaziyaCevir(eksikMiktar) : SayiyiYaziyaCevir(eksikMiktar * 1000, true);

finalMesaj.AppendLine($"{bestYemP.Ad.PadRight(20)}P".PadRight(15) +
$"{miktarStr.PadRight(15)}{okunus}");

            }
        }

        finalMesaj.AppendLine();

        finalMesaj.AppendLine("=====");

        finalMesaj.AppendLine("Veteriner Otomasyon Ekibi");

        finalMesaj.AppendLine("=====");

        MessageBox.Show(finalMesaj.ToString(), "Besleme Programı Raporu",
MessageBoxButtons.OK, MessageBoxIcon.Information);

        sonHesaplamaMesaji = finalMesaj.ToString();
    }

    catch (Exception ex)
    {
        MessageBox.Show($"Hesaplama sırasında hata: {ex.Message}", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);

        sonHesaplamaMesaji = null;
    }
}

```

```

private void UpdateLabel6()
{
    if (!mevcutYemler.Any())
    {
        label6.Text = "Yem stoğu boş. Lütfen yem ekleyin.";
        return;
    }

    StringBuilder yemListesi = new StringBuilder();

    yemListesi.AppendLine("=====");
    yemListesi.AppendLine("            YEM STOK RAPORU            ");
    yemListesi.AppendLine("=====");
    yemListesi.AppendLine("Yem Adı".PadRight(20) + "Miktar".PadRight(15) +
"Okunuş");
    yemListesi.AppendLine(new string('-', 50));

    double toplamKM = 0, toplamME = 0, toplamHP = 0, toplamCa = 0, toplamP = 0;

    foreach (var yem in mevcutYemler)
    {
        string miktarStr = yem.Miktar >= 1 ? $"{yem.Miktar:F2} kg" : $"{(yem.Miktar
* 1000):F2} g";

        string okunus = yem.Miktar >= 1 ? SayiyiYaziyaCevir(yem.Miktar) :
SayiyiYaziyaCevir(yem.Miktar * 1000, true);

        yemListesi.AppendLine($"{yem.Ad.PadRight(20)}{miktarStr.PadRight(15)}{okunus}");

        toplamKM += yem.Miktar * (yem.KM / 100);
        toplamME += yem.Miktar * yem.ME;
        toplamHP += (yem.Miktar * (yem.HP / 100)) * 1000;
        toplamCa += (yem.Miktar * (yem.Ca / 100)) * 1000;
    }
}

```

```

        toplamP += (yem.Miktar * (yem.P / 100)) * 1000;
    }

    yemListesi.AppendLine();

    yemListesi.AppendLine("-----");

    yemListesi.AppendLine("Toplam Besin Değerleri");

    yemListesi.AppendLine("-----");

    yemListesi.AppendLine($"KM: {toplamKM:F2} kg".PadRight(20) +
        $"({SayiyiYaziyaCevir(toplamKM)})");

    yemListesi.AppendLine($"ME: {toplamME:F2} Mcal".PadRight(20) +
        $"({SayiyiYaziyaCevir(toplamME)} Mcal)");

    yemListesi.AppendLine($"HP: {toplamHP:F2} g".PadRight(20) +
        $"({SayiyiYaziyaCevir(toplamHP, true)})");

    yemListesi.AppendLine($"Ca: {toplamCa:F2} g".PadRight(20) +
        $"({SayiyiYaziyaCevir(toplamCa, true)})");

    yemListesi.AppendLine($"P: {toplamP:F2} g".PadRight(20) +
        $"({SayiyiYaziyaCevir(toplamP, true)})");

    yemListesi.AppendLine("=====");

    yemListesi.AppendLine("Veteriner Otomasyon Sistemi");

    yemListesi.AppendLine("=====");

    label6.Text = yemListesi.ToString();
}

private async void button2_Click(object? sender, EventArgs e)
{
    try
    {
        if (!await TestConnectionAsync())
        {
            return;
        }
    }
}

```

```

        string? yemAdi = textBox3.Text?.Trim();

        if (string.IsNullOrEmpty(yemAdi) || !double.TryParse(textBox4.Text, out
double miktar))
        {
            MessageBox.Show("Yem türü ve miktarı geçerli olmalıdır!", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }

        var yemler = await YemiYukleAsync();

        if (!yemler.Any())
        {
            MessageBox.Show("Veritabanında yem verisi bulunamadı!", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }

        var secilenYem = yemler.FirstOrDefault(y => y.Ad.Equals(yemAdi,
StringComparison.OrdinalIgnoreCase));

        if (secilenYem == null)
        {
            MessageBox.Show($"Belirtilen yem türü ({yemAdi}) veritabanında
bulunamadı!", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }

        var mevcutYem = mevcutYemler.FirstOrDefault(y => y.Ad == secilenYem.Ad);

        if (mevcutYem == null)
        {
            mevcutYemler.Add(new Yem
            {
                Ad = secilenYem.Ad,

```

```

        KM = secilenYem.KM,

        ME = secilenYem.ME,

        HP = secilenYem.HP,

        Ca = secilenYem.Ca,

        P = secilenYem.P,

        Miktar = miktar

    });

}

else

{

    mevcutYem.Miktar += miktar;

}

UpdateLabel6();

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox4.Clear();

}

catch (Exception ex)

{

    MessageBox.Show($"Yem ekleme sırasında hata: {ex.Message}", "Hata",
    MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button3_Click(object? sender, EventArgs e)

{

    try

```

```

    {

        Form1 anaSayfa = new Form1();

        anaSayfa.Show();

        this.Close();

    }

    catch (Exception ex)

    {

        MessageBox.Show($"Ana sayfaya dönerken hata oluştu: {ex.Message}", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);

    }

}

private async void button4_Click(object? sender, EventArgs e)

{

    try

    {

        if (string.IsNullOrEmpty(sonHesaplamaMesaji))

        {

            MessageBox.Show("Önce bir hesaplama yapmalısınız! Lütfen Button1 ile
hesaplama yapın.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;

        }

        string? aliciEmail = textBox5.Text?.Trim();

        if (string.IsNullOrEmpty(aliciEmail))

        {

            MessageBox.Show("Lütfen alıcı e-posta adresini girin (textBox5).",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;

        }

    }

}

```



```
string gondericiEmail = string.Empty;

string uygulamaSifresi = string.Empty;
```

```
using (var conn = new SQLiteConnection(connectionString))
{
```

```
    await conn.OpenAsync();
```

```
    string query = "SELECT email, password FROM mail LIMIT 1";
```

```
    using (var cmd = new SQLiteCommand(query, conn))
```

```
    using (var reader = await cmd.ExecuteReaderAsync())
```

```
    {
```

```
        if (await reader.ReadAsync())
```

```
        {
```

```
            gondericiEmail = reader["email"] as string ?? string.Empty;
```

```
            uygulamaSifresi = reader["password"] as string ?? string.Empty;
```

```
        }
```

```
    }
```

```
}
```

```
        if (string.IsNullOrEmpty(gondericiEmail) ||
string.IsNullOrEmpty(uygulamaSifresi))
```

```
        {
```

```
            MessageBox.Show("Gönderici e-posta veya şifre veritabanından  
çekilemedi.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            return;
```

```
        }
```

```
StringBuilder emailBody = new StringBuilder();
```

```
emailBody.AppendLine("Merhaba,");
```

```
emailBody.AppendLine();
```

```
        emailBody.AppendLine("Bu e-posta, veteriner otomasyon sisteminden  
hesaplanan 1 aylık besleme programını içermektedir.");
```

```

        emailBody.AppendLine($"Rapor Tarihi: {DateTime.Now:dd.MM.yyyy HH:mm:ss}");

        emailBody.AppendLine();

        emailBody.AppendLine("Aşağıda, seçtiğiniz hayvan türü için hesaplanan 1 aylık besleme programı detaylı bir şekilde listelenmiştir. Bu program, sistemdeki güncel verilere dayanmaktadır ve raporun oluşturulduğu tarih ve saat yukarıda belirtilmiştir.");

        emailBody.AppendLine();

        emailBody.AppendLine("Besleme Programı:");

        emailBody.AppendLine(new string('-', 50));

        emailBody.AppendLine(sonHesaplamaMesaji);

        emailBody.AppendLine(new string('-', 50));

        emailBody.AppendLine();

        emailBody.AppendLine("İyi günler,");

        emailBody.AppendLine("Veteriner Otomasyon Ekibi");


        using (SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587))
        {

            smtpClient.Credentials = new
System.Net.NetworkCredential(gondericiEmail, uygulamaSifresi);

            smtpClient.EnableSsl = true;


            using (MailMessage mailMessage = new MailMessage())
            {

                mailMessage.From = new MailAddress(gondericiEmail);

                mailMessage.To.Add(aliciEmail);

                mailMessage.Subject = $"Veteriner Otomasyon - 1 Aylık Besleme Programı - {DateTime.Now:dd.MM.yyyy HH:mm:ss}";

                mailMessage.Body = emailBody.ToString();

                mailMessage.IsBodyHtml = false;


                smtpClient.Send(mailMessage);

            }

        }

```

```

        MessageBox.Show("E-posta başarıyla gönderildi!", "Bilgi",
        MessageBoxButtons.OK, MessageBoxIcon.Information);

    }

    catch (Exception ex)

    {

        MessageBox.Show($"E-posta gönderilirken hata oluştu:
        {ex.Message}\nStackTrace: {ex.StackTrace}", "Hata", MessageBoxButtons.OK,
        MessageBoxIcon.Error);

    }

}

private void pictureBox1_Click(object sender, EventArgs e)

{

    this.Hide();

    Form14 form14 = Application.OpenForms["Form14"] as Form14;

    if (form14 != null)

    {

        form14.Show();

    }

    else

    {

        form14 = new Form14();

        form14.Show();

    }

}

}

}

```

Form16 kodunuz, bir veteriner otomasyon sisteminde hayvanlar için günlük yem ihtiyaçlarını hesaplayan ve besleme programları oluşturan bir Windows Forms uygulamasını temsil ediyor. Kod, SQLite veritabanından yem ve hayvan türü verilerini çekiyor, kullanıcı girdilerine (hayvan türü, ağırlık, aylık kilo artışı) dayalı olarak yem dağılımını hesaplıyor ve eksik besinler için öneriler sunuyor. Aşağıda, kodda kullanılan matematiksel formüllerin ve hesaplamaların detaylı bir açıklamasını sunuyorum.

---

## 1. Genel Matematiksel Çerçeve

Kod, aşağıdaki temel adımları izleyerek besleme programını hesaplar:

1. Hayvan İhtiyaçlarının Belirlenmesi: Hayvan türü, ağırlık ve günlük kilo artışı baz alınarak günlük kuru madde (KM), metabolik enerji (ME), ham protein (HP), kalsiyum (Ca) ve fosfor (P) ihtiyaçları belirlenir.
  2. Yem Dağılımı: Mevcut yem stoğundan, hayvanın ihtiyaçlarını karşılayacak şekilde yem miktarı dağıtılır.
  3. Eksik Besin Hesaplama: İhtiyaçlar ile kullanılan yemlerin sağladığı besinler karşılaştırılır ve eksik kalan besinler hesaplanır.
  4. Öneriler: Eksik besinleri tamamlamak için uygun yemler önerilir.
- 

## 2. Matematiksel Formüller ve Hesaplamalar

### 2.1. Günlük Yem Miktarı Hesaplama

Formül:

$$\text{Günlük Yem Miktarı (kg)} = \text{Günlük Kilo Artışı (kg)} \times 10$$

Açıklama: Günlük kilo artışı, aylık kilo artışının 30'a bölünmesiyle hesaplanır:

$$\text{Günlük Kilo Artışı (kg)} = \frac{\text{Aylık Kilo Artışı (kg)}}{30}$$

Kod Referansı (button1\_Click):

```
csharp
```

```
double gunlukArtis = aylıkArtis / 30;  
gunlukYemMiktari = gunlukArtis * 10;
```

Örnek:

Aylık kilo artışı = 30 kg

Günlük kilo artışı =

$$\frac{30}{30} = 1$$

kg

Günlük yem miktarı =

$$1 \times 10 = 10$$

kg

### 2.2. İhtiyaçların Belirlenmesi

Veritabanı Sorgusu: Hayvan türüne, ağırlığa (CA\_kg) ve günlük kilo artışına (GCAA\_g) göre ihtiyaçlar (KM\_kg, ME\_Mcal, HP\_g, Ca\_g, P\_g) veritabanından çekilir.

Yuvarlama: Girilen ağırlık ve günlük artış, veritabanındaki en yakın değerlere yuvarlanır:

$\text{En Yakın Ağırlık} = \arg\min_{x \in \text{Ağırlıklar}} |\text{Ağırlık} - x|$

$\text{En Yakın Günlük Artış} = \arg\min_{x \in \text{Günlük Artışlar}} |\text{Günlük Artış} - x|$

Varsayılan Değerler: Eğer veritabanında veri yoksa, varsayılan ihtiyaçlar kullanılır:

KM = 10 kg

ME = 25 Mcal

HP = 500 g

Ca = 40 g

P = 20 g

Kod Referansı (IhtiyaciYukleAsync):

csharp

```
double enYakinAgirlik = agirliklar.OrderBy(x => Math.Abs(x - agirlik)).First();
```

```
double enYakinGunlukArtis = gunlukArtislar.OrderBy(x => Math.Abs(x - gunlukArtis)).First();
```

### 2.3. Yem Dağılımı (YemDagit)

Yem dağılımı, hayvanın ihtiyaçlarını karşılamak için mevcut yemlerin optimal şekilde dağıtılmasını sağlar. Bu işlem, bir puanlama sistemiyle yapılır.

Puanlama Formülü: Her yem için bir puan hesaplanır, bu puan yemin ihtiyaçlara katkısını değerlendirir:

$\text{Puan}_{\text{yem}} = w_{\text{HP}} \cdot$

$\frac{\text{HP}_{\text{yem}}/100}{\text{HP}_{\text{ihtiyaç}}} \cdot \text{Kalan HP} +$

$w_{\text{P}} \cdot \frac{\text{P}_{\text{yem}}/100}{\text{P}_{\text{ihtiyaç}}} \cdot$

$\text{Kalan P} +$

$w_{\text{KM}} \cdot \frac{\text{KM}_{\text{yem}}/100}{\text{KM}_{\text{ihtiyaç}}} \cdot$

$\text{Kalan KM} +$

$w_{\text{ME}} \cdot \frac{\text{ME}_{\text{yem}}}{\text{ME}_{\text{ihtiyaç}}} \cdot$

$\text{Kalan ME} +$

$w_{\text{Ca}} \cdot \frac{\text{Ca}_{\text{yem}}/100}{\text{Ca}_{\text{ihtiyaç}}} \cdot$

$\text{Kalan Ca}$

$w_{\text{besin}}$

: Ağırlık faktörü (kodda eşit kabul edilir, yani

$w = 1$

).

$\text{Kalan Besin}$

: Henüz karşılanmamış besin miktarı.

$\text{Besin}_{\text{yem}}$

: Yemin birim başına sağladığı besin miktarı (örneğin,

$\text{KM}_{\text{yem}}$

% cinsinden).

$\text{Besin}_{\text{ihtiyaç}}$

: Hayvanın toplam besin ihtiyacı.

Toplam Puan:

$$\text{Toplam Puan} = \sum_{\text{yem}} \text{Puan}_{\text{yem}}$$

Yem Miktarı Dağılımı: Her yem için kullanılacak miktar, puan oranına göre hesaplanır:

$$\text{Kullanılan Miktar}_{\text{yem}} = \min(\text{Mevcut Miktar}_{\text{yem}},$$

$$\text{Günlük Yem Miktarı}) \cdot \frac{\text{Puan}_{\text{yem}}}{\text{Toplam Puan}}$$

Eğer Puanlama Başarısız Olursa: Eğer toplam puan sıfır veya yemlerin puanları hesaplanamazsa, yemler mevcut miktarlarına oranla dağıtılır:

$$\text{Oran}_{\text{yem}} = \frac{\text{Mevcut Miktar}_{\text{yem}}}{\sum_{\text{yem}} \text{Mevcut Miktar}_{\text{yem}}}$$

$$\text{Kullanılan Miktar}_{\text{yem}} = \min(\text{Mevcut Miktar}_{\text{yem}},$$

$$\text{Günlük Yem Miktarı}) \cdot \text{Oran}_{\text{yem}}$$

Kalan Miktar Dağılımı: Eğer toplam dağıtılan yem miktarı günlük yem miktarından azsa, kalan miktar en yüksek puanlı yemlere dağıtılır:

$$\text{Kalan Miktar} = \text{Günlük Yem Miktarı} - \sum_{\text{yem}} \text{Kullanılan Miktar}_{\text{yem}}$$

$$\text{Ek Miktar}_{\text{yem}} = \min(\text{Mevcut Miktar}_{\text{yem}} -$$

$$\text{Kullanılan Miktar}_{\text{yem}}, \text{Kalan Miktar})$$

Besin Karşılama Hesaplaması: Her yem için kullanılan miktar, ilgili besin katkılarını hesaplar:

$$\text{Kalan KM} = \text{KM}_{\text{ihtiyaç}} - \sum_{\text{yem}} (\text{Kullanılan Miktar}_{\text{yem}} \cdot \frac{\text{KM}_{\text{yem}}}{100})$$

$$\text{Kalan ME} = \text{ME}_{\text{ihtiyaç}} - \sum_{\text{yem}} (\text{Kullanılan Miktar}_{\text{yem}} \cdot \text{ME}_{\text{yem}})$$

$$\text{Kalan HP} = \text{HP}_{\text{ihtiyaç}} - \sum_{\text{yem}} (\text{Kullanılan Miktar}_{\text{yem}} \cdot \frac{\text{HP}_{\text{yem}}}{100} \cdot 1000)$$

$$\text{Kalan Ca} = \text{Ca}_{\text{ihtiyaç}} - \sum_{\text{yem}} (\text{Kullanılan Miktar}_{\text{yem}} \cdot \frac{\text{Ca}_{\text{yem}}}{100} \cdot 1000)$$

$$\text{Kalan P} = \text{P}_{\text{ihtiyaç}} - \sum_{\text{yem}} (\text{Kullanılan Miktar}_{\text{yem}} \cdot \frac{\text{P}_{\text{yem}}}{100} \cdot 1000)$$

Eksik Besinler:

$$\text{Eksik Besin} = \max(0, \text{Kalan Besin})$$

Kod Referansı (YemDagit):

csharp

```
double puan = 0;
```

```
if (kalanHP > 0 && yem.HP > 0) puan += (yem.HP / 100) * (kalanHP / ihtiyac.HP);
```

```
if (kalanP > 0 && yem.P > 0) puan += (yem.P / 100) * (kalanP / ihtiyac.P);
```

```
if (kalanKM > 0 && yem.KM > 0) puan += (yem.KM / 100) * (kalanKM / ihtiyac.KM);
```

```
if (kalanME > 0 && yem.ME > 0) puan += yem.ME * (kalanME / ihtiyac.ME);
```

```
if (kalanCa > 0 && yem.Ca > 0) puan += (yem.Ca / 100) * (kalanCa / ihtiyac.Ca);
```

## 2.4. Eksik Besin Önerileri

Eksik kalan besinler için en uygun yem önerilir. Her besin için en yüksek içeriğe sahip yem seçilir ve eksik miktarı tamamlamak için gerekli yem miktarı hesaplanır.

Formüller:

Kuru Madde (KM):

$$\text{Eksik KM Miktarı (kg)} = \frac{\text{Eksik KM (kg)}}{\text{KM}_{\text{yem}}/100}$$

Metabolik Enerji (ME):

$$\text{Eksik ME Miktarı (kg)} = \frac{\text{Eksik ME (Mcal)}}{\text{ME}_{\text{yem}}}$$

Ham Protein (HP):

$$\text{Eksik HP Miktarı (kg)} = \frac{\text{Eksik HP (g)}}{\text{HP}_{\text{yem}}} \cdot 10$$

Kalsiyum (Ca):

$$\text{Eksik Ca Miktarı (kg)} = \frac{\text{Eksik Ca (g)}}{\text{Ca}_{\text{yem}}} \cdot 10$$

Fosfor (P):

$$\text{Eksik P Miktarı (kg)} = \frac{\text{Eksik P (g)}}{\text{P}_{\text{yem}}} \cdot 10$$

Kod Referansı (button1\_Click):

csharp

```
if (eksikKM > 0)
{
    var bestYemKM = hesaplamaYemler.Where(y => y.KM > 0).OrderByDescending(y => y.KM).FirstOrDefault();
    if (bestYemKM != null)
    {
        double eksikMiktar = eksikKM / (bestYemKM.KM / 100);
        ...
    }
}
```

## 2.5. Besin Yüzdeleri

Dağıtılan yemlerin toplam besin katkıları, toplam yem miktarına bölünerek yüzdelik oranlar hesaplanır:

$$\text{KM Yüzdesi (\%)} = \frac{\text{Toplam KM (kg)}}{\text{Toplam Kullanılan Yem (kg)}} \cdot 100$$

$$\text{ME (Mcal/kg)} = \frac{\text{Toplam ME (Mcal)}}{\text{Toplam Kullanılan Yem (kg)}}$$

$$\text{HP Yüzdesi (\%)} = \frac{\text{Toplam HP (g)}}{\text{Toplam Kullanılan Yem (kg)}} \cdot 100$$

$$\text{Ca Yüzdesi (\%)} = \frac{\text{Toplam Ca (g)}}{\text{Toplam Kullanılan Yem (kg)}} \cdot 100$$

$$\text{P Yüzdesi (\%)} = \frac{\text{Toplam P (g)}}{\text{Toplam Kullanılan Yem (kg)}} \cdot 100$$

Kod Referansı (button1\_Click):

csharp

```
double kmYuzde = toplamKullanilacakYem > 0 ? (toplamlKM / toplamKullanilacakYem) * 100 : 0;
double meMcalKg = toplamKullanilacakYem > 0 ? toplamME / toplamKullanilacakYem : 0;
double hpYuzde = toplamKullanilacakYem > 0 ? (toplamlHP / 1000 / toplamKullanilacakYem) * 100 : 0;
```

```
double caYuzde = toplamKullanilacakYem > 0 ? (toplamlCa / 1000 / toplamKullanilacakYem) * 100 : 0;  
double pYuzde = toplamKullanilacakYem > 0 ? (toplamlP / 1000 / toplamKullanilacakYem) * 100 : 0;
```

## 2.6. Sayıyı Yazıya Çevirme

Yem miktarlarını ve besin değerlerini Türkçe olarak yazıya çevirmek için bir algoritma kullanılır:

Tam Kısım:

Sayı 1000'lik gruplara bölünür (birler, binler, milyonlar, milyarlar).

Her grup için yüzler, onlar ve birler basamakları ayrı ayrı yazılır.

Örnek: 1234 → "bin iki yüz otuz dört"

Ondalık Kısım:

Ondalık kısım 100 ile çarpılarak iki basamaklı bir sayı elde edilir.

Bu sayı, onlar ve birler basamaklarına ayrılarak yazılır.

Örnek: 0.45 → "virgül kırk beş"

Birim:

Miktar 1 kg'dan büyükse "kilogram", küçükse "gram" kullanılır.

Kod Referansı (SayıYazıyaCevir):

csharp

```
string[] birler = { "", "bir", "iki", "üç", "dört", "beş", "altı", "yedi", "sekiz", "dokuz" };  
string[] onlar = { "", "on", "yirmi", "otuz", "kırk", "elli", "altmış", "yetmiş", "seksen", "doksan" };  
string[] binler = { "", "bin", "milyon", "milyar" };
```

---

## 3. Örnek Hesaplama

Girdi:

Hayvan Türü: Yerli Irk Erkek Dana

Ağırlık: 300 kg

Aylık Kilo Artışı: 30 kg

Yem Stoğu:

Yem A: KM=70%, ME=2.5 Mcal/kg, HP=15%, Ca=1%, P=0.5%, Miktar=5 kg

Yem B: KM=60%, ME=2.0 Mcal/kg, HP=10%, Ca=0.8%, P=0.4%, Miktar=3 kg

İhtiyaçlar (veritabanından):

KM: 8 kg

ME: 20 Mcal

HP: 400 g

Ca: 30 g

P: 15 g

Hesaplama Adımları:

1. Günlük Yem Miktarı:

$\text{Günlük Artış} = \frac{30}{30} = 1 \text{ , kg}$

$\text{Günlük Yem Miktarı} = 1 \times 10 = 10 \text{ , kg}$



## 2. Puanlama:

Yem A için:

$$\begin{aligned}\text{Puan}_A &= \left( \frac{15}{100} \cdot 400 \right) + \left( \frac{0.5}{100} \cdot 15 \right) + \left( \frac{70}{100} \cdot 8 \right) + \left( \frac{2.5}{20} \cdot 20 \right) + \left( \frac{1}{100} \cdot 30 \right) \\ &= 0.15 + 0.005 + 0.7 + 2.5 + 0.01 = 3.365\end{aligned}$$

Yem B için:

$$\begin{aligned}\text{Puan}_B &= \left( \frac{10}{100} \cdot 400 \right) + \left( \frac{0.4}{100} \cdot 15 \right) + \left( \frac{60}{100} \cdot 8 \right) + 8.0 \\ &= 0.1 + 0.004 + 0.6 + 2.0 + 0.008 = 2.712\end{aligned}$$

Toplam Puan:

$$3.365 + 2.712 = 6.077$$

## 3. Yem Dağılımı:

Yem A:

$$\text{Miktar}_A = \min(5, 10 \cdot \frac{3.365}{6.077}) \approx \min(5, 5.54) = 5 \text{ kg}$$

Yem B:

$$\text{Miktar}_B = \min(3, 10 \cdot \frac{2.712}{6.077}) \approx \min(3, 4.46) = 3 \text{ kg}$$

Toplam:

$$5 + 3 = 8 \text{ kg}$$

## 4. Kalan Besinler:

KM:

$$\text{KM}_{\text{kullanılan}} = 5 \cdot \frac{70}{100} + 3 \cdot \frac{60}{100} = 3.5 + 1.8 = 5.3 \text{ kg}$$

$$\text{Eksik KM} = \max(0, 8 - 5.3) = 2.7 \text{ kg}$$

ME:

$$\text{ME}_{\text{kullanılan}} = 5 \cdot 2.5 + 3 \cdot 2.0 = 12.5 + 6.0 = 18.5 \text{ Mcal}$$

$$\text{Eksik ME} = \max(0, 20 - 18.5) = 1.5 \text{ Mcal}$$

HP:

$$\text{HP}_{\text{kullanılan}} = (5 \cdot \frac{15}{100} \cdot 1000) + (3 \cdot \frac{10}{100} \cdot 1000) = 750 + 300 = 1050 \text{ g}$$

$$\text{Eksik HP} = \max(0, 400 - 1050) = 0 \text{ g}$$

Ca:

$$\text{Ca}_{\text{kullanılan}} = (5 \cdot \frac{1}{100} \cdot 1000) + (3 \cdot \frac{0.8}{100} \cdot 1000) = 50 + 24 = 74 \text{ g}$$

$$\text{Eksik Ca} = \max(0, 30 - 74) = 0 \text{ g}$$

P:

$$\text{P}_{\text{kullanılan}} = (5 \cdot \frac{0.5}{100} \cdot 1000) + (3 \cdot \frac{0.4}{100} \cdot 1000) = 25 + 12 = 37 \text{ g}$$

$$\text{Eksik P} = \max(0, 15 - 37) = 0 \text{ g}$$

## 5. Öneriler:

Eksik KM için (Yem A seçilir, çünkü  $\text{KM}=70\% > 60\%$ ):

$\text{Miktar} = \frac{2.7}{70/100} \approx 3.86 \text{ kg}$   
Eksik ME için (Yem A seçilir, çünkü ME=2.5 > 2.0):  
 $\text{Miktar} = \frac{1.5}{2.5} = 0.6 \text{ kg}$

#### 4. Özet

Günlük Yem Miktarı: Günlük kilo artışı  $\times$  10

Yem Dağılımı: Puanlama sistemiyle yemler oransal olarak dağıtılır.

Besin Karşılama: Kullanılan yemlerin besin katkıları, ihtiyaçlarla karşılaştırılır.

Eksik Besin Önerileri: Eksik besinler için en yüksek içerikli yemler önerilir.

Yüzdelik Oranlar: Toplam besin miktarları, kullanılan yem miktarına bölünerek yüzdelik

oranlar hesaplanır.

KM : kuru madde (numunenin nemsiz içeriği ,rasyon her zaman kuru madde temelinde dengelemek ve değerlendirmek önemlidir.)

**ME,Mcal/kg :** (metabolik enerjidir , ineğinin enerji ihtiyacı hesaplandıktan sonra , verilecek yem miktarı , yemleri kuru madde ağırlığındaki metabolik enerjiye göre belirlenir)

**HP, %:**(ham protein yemlerdeki toplam azot içeriğini temsil eder ve genellikle protein miktarını belirlemek için kullanılır )

Ca, %:(yemlerdeki kalsiyumdur hayvanların kemik gelişimi , diş sağlığı , kas fonksiyonları ve sinir iletimi için gereklidir ayrıca yemlerin mineral dengesini sağlamak için kullanılır)

P, %:(yemlerdeki fosfor oranıdır , fosfor ; yemlerdeki enerji ve protein ile birlikte hayvanların büyüme ve verimliliğini destekler)

ÖRNEK KULLANIM:

#### Bilgi



Girilen ağırlık (200 kg) ve günlük artış (0,16666666666666666 kg) değerleri yuvarlandı.

Kullanılan değerler: Ağırlık = 182 kg, Günlük Artış = 200 kg.

Tamam



## =====

## BESLEME PROGRAMI RAPORU

=====

Rapor Tarihi: 27.04.2025 19:49:39  
Hayvan Türü: YERLİ İRK ERKEK DANA  
Ağırlık: 200,00 kg (iki yüz kilogram)  
Aylık Kilo Artışı: 5,00 kg (beş kilogram) (Günlük: 0,17 kg)  
Günlük Toplam Yem Miktarı: 1,67 kg (bir virgöl altmış altı kilogram)

## -----

## Günlük Yem Dağılımı

Yem Adı	Miktar	Okunuş
ARPA	1,67 kg	bir virgöl altmış altı kilogram

## -----

## Besin Yüzdeleri

-----

KM (%): 89,00  
ME (Mcal/kg): 3,13  
HP (%): 12,00  
Ca (%): 0,06  
P (%): 0,38

## -----

## Besin İhtiyaçları ve Karşılama Durumu

Besin	İhtiyaç	Kullanılan	Kalan	Durum
KM (kg)	4,40	1,48	2,92	Eksik
ME (Mcal)	8,54	5,22	3,32	Eksik
HP (g)	396,00	200,00	196,00	Eksik
Ca (g)	12,32	1,00	11,32	Eksik
P (g)	7,92	6,33	1,59	Eksik

## -----

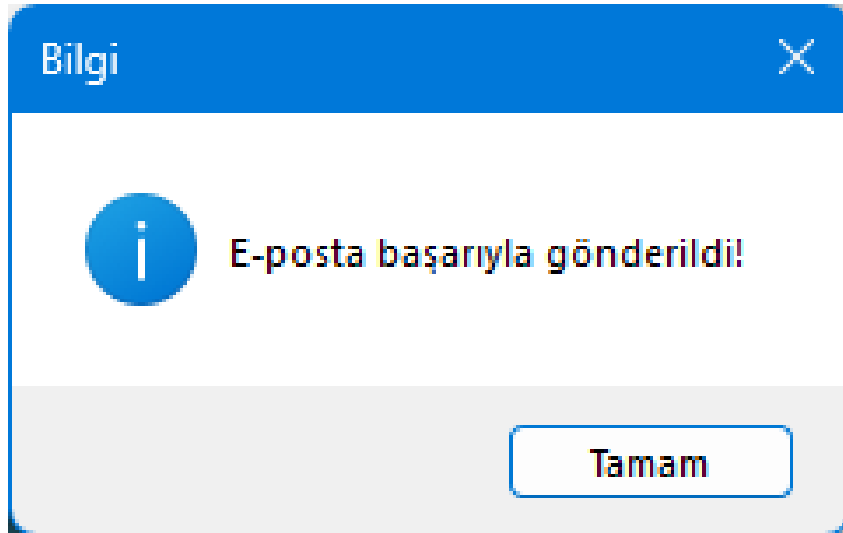
## Eksik Besin Önerileri

Yem Adı	Besin	Miktar	Okunuş
ARPA	KM3,28 kg	üç virgöl yirmi yedi kilogram	
ARPA	ME1,06 kg	bir virgöl altı kilogram	
ARPA	HP1,63 kg	bir virgöl altmış üç kilogram	
ARPA	Ca18,87 kg	on sekiz virgöl seksen altı kilogram	
ARPA	P417,54 g	dört yüz on yedi virgöl elli dört gram	

=====

Veteriner Otomasyon Ekibi

=====



```
CREATE TABLE "Yem" (  
    "Kod" INTEGER,  
    "Yemler" TEXT,  
    "KM_Yuzde" DECIMAL(5, 2),  
    "ME_Mcal_kg" DECIMAL(5, 2),  
    "NEyp_Mcal_kg" DECIMAL(5, 2),  
    "NEb_Mcal_kg" DECIMAL(5, 2),  
    "HP_Yuzde" DECIMAL(5, 2),  
    "Fiyat_TL" DECIMAL(10, 2),  
    "Ca_Yuzde" DECIMAL(5, 2),  
    "P_Yuzde" DECIMAL(5, 2)  
);
```

```
CREATE TABLE "Hayvan_Turleri" (  
    "id" INTEGER,
```

```

"Hayvan_Turu"      TEXT,

"CA_kg"            REAL,

"GCAA_g"           REAL,

"KM_kg"            REAL,

"ME_Mcal"          REAL,

"NEb_Mcal"         REAL,

"NEyp_Mcal"        REAL,

"HP_g" REAL,

"Ca_g" REAL,

"P_g" REAL,

PRIMARY KEY("id" AUTOINCREMENT)

```

```
);
```

## FORM 18: KEDİ AŞI TAKİP

KEDİ AŞI TAKİP

İSİM SOYİSİM

TELEFON NO:

CİNSİ

AŞI ADI:

aşı süresi

doz sayısı

Kaydet

Yaş (Hafta)

PARAZİT DURUMU

☐ İÇ PARAZİT
☐ DIŞ PARAZİT

☐ YAPILDI
☐ YAPILMADI

☐ 6-8 HAFTA
☐ 9-11 HAFTA
☐ 12-14 HAFTA
☐ 13-15 HAFTA
☐ 4- 6 AYLIK
☐ 12 AYLIK (1 YIL)

İNEK(gen)

KÖPEK

→

X

yyyy-MM-dd (örneğin: 2025-04-...)

KupeNo	Telefon	Cinsi	AşıAdı	Durum	YasAraligi	Periyot	Doz Numarası	Digerdoz	Sure	HayvanSahibi	Adres	AsiSo
»												

```

using System;

using System.Data;

```

```

using System.Drawing;

using System.Windows.Forms;

using System.Data.SQLite;

namespace WinFormsApp8
{
    public partial class Form18 : Form
    {
        private readonly string connectionString = @"Data
Source=C:\Users\ridva\Desktop\SQLITE3_VETERİNEROTOMASYON\veteriner_otomasyon.db;Version=3;"
;

        private System.Windows.Forms.Timer? timer;

        public Form18()
        {
            InitializeComponent();

            InitializeCheckBoxes();

            InitializeTimer();

            LoadDataGridView();

            StyleDataGridView();

            dataGridView1.SelectionChanged += DataGridView1_SelectionChanged;
        }

        private void InitializeCheckBoxes()
        {
            checkBox1.CheckedChanged += (s, e) =>
            {
                if (checkBox1.Checked) checkBox2.Checked = false;
            };

            checkBox2.CheckedChanged += (s, e) =>
            {

```

```

        if (checkBox2.Checked) checkBox1.Checked = false;

    };

}

private void InitializeTimer()
{
    timer = new System.Windows.Forms.Timer { Interval = 1000 };

    timer.Tick += (s, e) => UpdateDogumaKalanSure();

    timer.Start();
}

private void StyleDataGridView()
{
    dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI", 11,
FontStyle.Bold);

    dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.White;

    dataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.FromArgb(33, 150,
243);

    dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;

    dataGridView1.EnableHeadersVisualStyles = false;

    dataGridView1.DefaultCellStyle.Font = new Font("Segoe UI Semibold", 10,
FontStyle.Regular);

    dataGridView1.DefaultCellStyle.ForeColor = Color.Black;

    dataGridView1.DefaultCellStyle.SelectionBackColor = Color.LightBlue;

    dataGridView1.DefaultCellStyle.SelectionForeColor = Color.Black;

    dataGridView1.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;

    dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.None;

    if (dataGridView1.Columns.Count > 0)

```

```

        {
            if (dataGridView1.Columns.Contains("ID")) dataGridView1.Columns["ID"].Width
= 80;

            if (dataGridView1.Columns.Contains("Telefon"))
dataGridView1.Columns["Telefon"].Width = 100;

            if (dataGridView1.Columns.Contains("Cinsi"))
dataGridView1.Columns["Cinsi"].Width = 100;

            if (dataGridView1.Columns.Contains("HayvanSahibi"))
dataGridView1.Columns["HayvanSahibi"].Width = 150;

            if (dataGridView1.Columns.Contains("GebelikDurumu"))
dataGridView1.Columns["GebelikDurumu"].Width = 100;

            if (dataGridView1.Columns.Contains("KayitTarihi"))
dataGridView1.Columns["KayitTarihi"].Width = 150;

            if (dataGridView1.Columns.Contains("DogumTarihi"))
dataGridView1.Columns["DogumTarihi"].Width = 150;

            if (dataGridView1.Columns.Contains("DogumaKalanSure"))
dataGridView1.Columns["DogumaKalanSure"].Width = 150;

        }

        dataGridView1.Visible = true;
    }

    private void DataGridView1_SelectionChanged(object? sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

            textBox1.Text = selectedRow.Cells["Telefon"].Value?.ToString() ?? "";

            textBox2.Text = selectedRow.Cells["Cinsi"].Value?.ToString() ?? "";

            textBox3.Text = selectedRow.Cells["HayvanSahibi"].Value?.ToString() ?? "";

            string gebelikDurumu = selectedRow.Cells["GebelikDurumu"].Value?.ToString()
?? "";

            checkBox1.Checked = gebelikDurumu == "Gebe";

            checkBox2.Checked = gebelikDurumu == "Gebe Değil";
        }
    }

```



```

    }
}

private bool PhoneNumberExists(string telefon)
{
    try
    {
        using (var conn = new SQLiteConnection(connectionString))
        {
            conn.Open();

            string query = "SELECT COUNT(*) FROM kedi_gebelik WHERE Telefon =
@Telefon";

            using (var cmd = new SQLiteCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@Telefon", telefon);

                return (long)cmd.ExecuteScalar() > 0;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Telefon numarası kontrolü sırasında hata: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return false;
    }
}

private void button1_Click(object sender, EventArgs e)
{
    try

```

```

{

    string telefon = textBox1.Text.Trim();

    string cinsi = textBox2.Text.Trim();

    string adSoyad = textBox3.Text.Trim();


    if (string.IsNullOrEmpty(telefon) || string.IsNullOrEmpty(cinsi)
|| string.IsNullOrEmpty(adSoyad))

    {

        MessageBox.Show("Lütfen tüm alanları doldurun! (Telefon, Cinsi, Hayvan Sahibi)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;

    }


    if (PhoneNumberExists(telefon))

    {

        MessageBox.Show("Bu telefon numarası zaten kayıtlı!", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;

    }


    string gebelikDurumu = checkBox1.Checked ? "Gebe" : checkBox2.Checked ?
"Gebe Değil" : "";

    DateTime? dogumTarihi = checkBox1.Checked ? DateTime.Now.AddDays(65) :
null;


    using (var conn = new SQLiteConnection(connectionString))

    {

        conn.Open();


        string query = @"

            INSERT INTO kedi_gebelik (Telefon, Cinsi, HayvanSahibi,
GebelikDurumu, KayitTarihi, DogumTarihi)

            VALUES (@Telefon, @Cinsi, @HayvanSahibi, @GebelikDurumu,
@KayitTarihi, @DogumTarihi)";

```

```

        using (var cmd = new SQLiteCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@Telefon", telefon);
            cmd.Parameters.AddWithValue("@Cinsi", cinsi);
            cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);
            cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);
            cmd.Parameters.AddWithValue("@KayitTarihi", DateTime.Now);
            cmd.Parameters.AddWithValue("@DogumTarihi", (object?)dogumTarihi ??
DBNull.Value);

            cmd.ExecuteNonQuery();
        }
    }

    MessageBox.Show("Kayıt başarıyla eklendi!", "Başarılı",
    MessageBoxButtons.OK, MessageBoxIcon.Information);

    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    checkBox1.Checked = false;
    checkBox2.Checked = false;

    LoadDataGridView();
}

catch (Exception ex)
{
    MessageBox.Show($"Hata oluştu: {ex.Message}", "Hata", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}

```

```
private void button2_Click(object sender, EventArgs e)
{
    // Placeholder for button2_Click
}

private void button3_Click_1(object sender, EventArgs e)
{
    try
    {
        // Validate textboxes

        string telefon = textBox1.Text.Trim();

        string cinsi = textBox2.Text.Trim();

        string adSoyad = textBox3.Text.Trim();

        if (string.IsNullOrEmpty(telefon) || string.IsNullOrEmpty(cinsi)
|| string.IsNullOrEmpty(adSoyad))
        {
            MessageBox.Show("Lütfen tüm alanları doldurun! (Telefon, Cinsi, Hayvan Sahibi)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }

        // Ensure a checkbox is selected

        if (!checkBox1.Checked && !checkBox2.Checked)
        {
            MessageBox.Show("Lütfen gebelik durumunu seçin! (Gebe veya Gebe Değil)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }

        // Determine GebelikDurumu based on selected checkbox
```

```

string gebelikDurumu = checkBox1.Checked ? "Gebe" : "Gebe Değil";

DateTime? dogumTarihi = checkBox1.Checked ? DateTime.Now.AddDays(65) :
null;

using (var conn = new SQLiteConnection(connectionString))
{
    conn.Open();

    // Check if a record with this phone number exists
    if (PhoneNumberExists(telefon))
    {
        // Update the existing record based on the phone number
        string query = @"
            UPDATE kedi_gebelik
            SET Cinsi = @Cinsi,
                HayvanSahibi = @HayvanSahibi,
                GebelikDurumu = @GebelikDurumu,
                DogumTarihi = @DogumTarihi
            WHERE Telefon = @Telefon";

        using (var cmd = new SQLiteCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@Cinsi", cinsi);
            cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);
            cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);
            cmd.Parameters.AddWithValue("@DogumTarihi",
(object?)dogumTarihi ?? DBNull.Value);
            cmd.Parameters.AddWithValue("@Telefon", telefon);

            int rowsAffected = cmd.ExecuteNonQuery();

            if (rowsAffected > 0)

```

```

        {
            MessageBox.Show("Kayıt başarıyla güncellendi!", "Başarılı",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("Güncelleme sırasında bir hata oluştu!",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }
    }
}
else
{
    // Optionally insert a new record if no record with this phone
number exists

    string query = @"
        INSERT INTO kedi_gebelik (Telefon, Cinsi, HayvanSahibi,
GebelikDurumu, KayitTarihi, DogumTarihi)
        VALUES (@Telefon, @Cinsi, @HayvanSahibi, @GebelikDurumu,
@KayitTarihi, @DogumTarihi)";

    using (var cmd = new SQLiteCommand(query, conn))
    {
        cmd.Parameters.AddWithValue("@Telefon", telefon);
        cmd.Parameters.AddWithValue("@Cinsi", cinsi);
        cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);
        cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);
        cmd.Parameters.AddWithValue("@KayitTarihi", DateTime.Now);
        cmd.Parameters.AddWithValue("@DogumTarihi",
(Object?)dogumTarihi ?? DBNull.Value);

        cmd.ExecuteNonQuery();
    }
}

```

```
        MessageBox.Show("Telefon numarası bulunamadı, yeni kayıt eklendi!",  
"Başarılı", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
    }
```

```
}
```

```
// Clear form
```

```
textBox1.Clear();
```

```
textBox2.Clear();
```

```
textBox3.Clear();
```

```
checkBox1.Checked = false;
```

```
checkBox2.Checked = false;
```

```
// Refresh DataGridView
```

```
LoadDataGridView();
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
        MessageBox.Show($"Hata oluştu: {ex.Message}", "Hata", MessageBoxButtons.OK,  
MessageBoxIcon.Error);
```

```
}
```

```
}
```

```
private void LoadDataGridView()
```

```
{
```

```
    try
```

```
    {
```

```
        using (var conn = new SQLiteConnection(connectionString))
```

```
        {
```

```
            conn.Open();
```

```

        string query = "SELECT ID, Cinsi, GebelikDurumu, KayitTarihi,
DogumTarihi, Telefon, HayvanSahibi FROM kedi_gebelik";

        using (var adapter = new SQLiteDataAdapter(query, conn))
        {

            DataTable dt = new DataTable();

            adapter.Fill(dt);

            if (!dt.Columns.Contains("DogumaKalanSure"))
            {

                dt.Columns.Add("DogumaKalanSure", typeof(string));

            }

            dataGridView1.DataSource = dt;

            UpdateDogumaKalanSure(dt);

        }

    }

    StyleDataGridView();

}

catch (Exception ex)

{

    MessageBox.Show($"Veri yükleme hatası: {ex.Message}", "Hata",
    MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void UpdateDogumaKalanSure()

{

    if (dataGridView1.DataSource is DataTable dt)

    {

        UpdateDogumaKalanSure(dt);

    }

}

```



```

        dataGridView1.Refresh();
    }
}

private void UpdateDogumaKalanSure(DataTable dt)
{
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        DataRow row = dt.Rows[i];

        if (i < dataGridView1.Rows.Count && !dataGridView1.Rows[i].IsNewRow)
        {
            DataGridViewRow dgvRow = dataGridView1.Rows[i];

            string gebelikDurumu = row["GebelikDurumu"]?.ToString() ?? "";

            if (string.IsNullOrEmpty(gebelikDurumu) || gebelikDurumu == "Gebe
Değil")
            {
                row["DogumaKalanSure"] = "Bilinmiyor";

                dgvRow.DefaultCellStyle.BackColor = Color.White;
            }

            else if (row["DogumTarihi"] != DBNull.Value && row["DogumTarihi"] is
DateTime dogumTarihi)
            {
                TimeSpan kalanSure = dogumTarihi - DateTime.Now;

                row["DogumaKalanSure"] = kalanSure.TotalSeconds > 0

                    ? $"{kalanSure.Days} gün {kalanSure.Hours} saat
{kalanSure.Minutes} dakika {kalanSure.Seconds} saniye"

                    : "Doğum zamanı geldi!";

                if (kalanSure.TotalSeconds <= 0)

                    dgvRow.DefaultCellStyle.BackColor = Color.White;
            }
        }
    }
}

```

```

        else if (kalanSure.TotalDays <= 15)

            dgvRow.DefaultCellStyle.BackColor = Color.Red;

        else

            dgvRow.DefaultCellStyle.BackColor = Color.Green;

    }

    else

    {

        row["DogumaKalanSure"] = "Bilinmiyor";

        dgvRow.DefaultCellStyle.BackColor = Color.White;

    }

}

}

```

```

protected override void OnFormClosing(FormClosingEventArgs e)
{

    base.OnFormClosing(e);

    timer?.Stop();

    timer?.Dispose();

}

```

```

private void pictureBox6_Click(object sender, EventArgs e)
{

    this.Hide();

    Form19 form19 = Application.OpenForms["Form19"] as Form19;

    if (form19 != null)

    {

        form19.Show();

    }

    else

```

```

        {
            form19 = new Form19();

            form19.Show();
        }
    }
}

CREATE TABLE "kedi_asi" (

    "ID"    INTEGER,

    "Telefon"    TEXT NOT NULL,

    "Cins" TEXT,

    "AsiAdi"    TEXT,

    "Durum"    TEXT CHECK("Durum" IN ('Yapildi', 'Yapilmadi')),

    "Yas" TEXT CHECK("Yas" IN ('6-8 HAFTA', '9-11 HAFTA', '12-14 HAFTA', '13-15 HAFTA',
'4-6 AYLIK', '12 AYLIK (1 YIL)')),

    "AsiSuresi" TEXT,

    "Doz" INTEGER,

    "KalanDoz" INTEGER,

    "Sure" TEXT,

    "LastDoseDate" TEXT,

    PRIMARY KEY("ID" AUTOINCREMENT),

    FOREIGN KEY("Telefon") REFERENCES "HayvanKayit"("Telefon")

);

```

---


FORM 18 : KEDİ GEBELİK

Kedi Gebelik

KÖPEK GEBELİK

Anasavfa

Telefon no göre filtreleme



ad soyad:

Telefon no:

Cinsi:

☐ GEBE  
☐ BOŞ

ID	Cinsi	gebelikDurum	KayıtTarihi	DogumTarihi	Telefon
1	a	Gebe	1.05.2025 15:41	6.07.2025 17:26	0535300
2	aa	Bilinmiyor	1.05.2025 19:19		05414445
3	a	Gebe	1.05.2025 19:20	6.07.2025 17:26	0535300
4	a		1.05.2025 19:27		0534729
5	a	Gebe Değil	1.05.2025 19:38		05417420
6	a		1.05.2025 19:48		05417420
7	Van Kedisi	Gebe	1.05.2025 23:30	6.07.2025 17:27	05321234
*					

```
using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using System.Data.SQLite;

namespace WinFormsApp8
{
    public partial class Form18 : Form
    {
        private readonly string connectionString = @"Data
Source=C:\Users\ridva\Desktop\SQLITE3_VETERİNEROTOMASYON\veteriner_otomasyon.db;Version=3;"
;

        private System.Windows.Forms.Timer? timer;

        public Form18()
        {
            InitializeComponent();

            InitializeCheckBoxes();
        }
    }
}
```

```
InitializeTimer();

LoadDataGridView();

StyleDataGridView();

dataGridView1.SelectionChanged += DataGridView1_SelectionChanged;

}
```

```
private void InitializeCheckBoxes()

{

    checkBox1.CheckedChanged += (s, e) =>

    {

        if (checkBox1.Checked) checkBox2.Checked = false;

    };

    checkBox2.CheckedChanged += (s, e) =>

    {

        if (checkBox2.Checked) checkBox1.Checked = false;

    };

}
```

```
private void InitializeTimer()

{

    timer = new System.Windows.Forms.Timer { Interval = 1000 };

    timer.Tick += (s, e) => UpdateDogumaKalanSure();

    timer.Start();

}
```

```
private void StyleDataGridView()

{

    dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI", 11,
FontStyle.Bold);

    dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.White;
```

```

dataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.FromArgb(33, 150,
243);

dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;

dataGridView1.EnableHeadersVisualStyles = false;

dataGridView1.DefaultCellStyle.Font = new Font("Segoe UI Semibold", 10,
FontStyle.Regular);

dataGridView1.DefaultCellStyle.ForeColor = Color.Black;

dataGridView1.DefaultCellStyle.SelectionBackColor = Color.LightBlue;

dataGridView1.DefaultCellStyle.SelectionForeColor = Color.Black;

dataGridView1.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.None;

if (dataGridView1.Columns.Count > 0)
{
    if (dataGridView1.Columns.Contains("ID")) dataGridView1.Columns["ID"].Width
= 80;

    if (dataGridView1.Columns.Contains("Telefon"))
dataGridView1.Columns["Telefon"].Width = 100;

    if (dataGridView1.Columns.Contains("Cinsi"))
dataGridView1.Columns["Cinsi"].Width = 100;

    if (dataGridView1.Columns.Contains("HayvanSahibi"))
dataGridView1.Columns["HayvanSahibi"].Width = 150;

    if (dataGridView1.Columns.Contains("GebelikDurumu"))
dataGridView1.Columns["GebelikDurumu"].Width = 100;

    if (dataGridView1.Columns.Contains("KayitTarihi"))
dataGridView1.Columns["KayitTarihi"].Width = 150;

    if (dataGridView1.Columns.Contains("DogumTarihi"))
dataGridView1.Columns["DogumTarihi"].Width = 150;

    if (dataGridView1.Columns.Contains("DogumaKalanSure"))
dataGridView1.Columns["DogumaKalanSure"].Width = 150;
}

dataGridView1.Visible = true;

```

```
}
```

```
private void DataGridView1_SelectionChanged(object? sender, EventArgs e)
```

```
{
```

```
    if (dataGridView1.SelectedRows.Count > 0)
```

```
    {
```

```
        DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];
```

```
        textBox1.Text = selectedRow.Cells["Telefon"].Value?.ToString() ?? "";
```

```
        textBox2.Text = selectedRow.Cells["Cinsi"].Value?.ToString() ?? "";
```

```
        textBox3.Text = selectedRow.Cells["HayvanSahibi"].Value?.ToString() ?? "";
```

```
        string gebelikDurumu = selectedRow.Cells["GebelikDurumu"].Value?.ToString() ?? "";
```

```
        checkBox1.Checked = gebelikDurumu == "Gebe";
```

```
        checkBox2.Checked = gebelikDurumu == "Gebe Değil";
```

```
    }
```

```
}
```

```
private bool PhoneNumberExists(string telefon)
```

```
{
```

```
    try
```

```
    {
```

```
        using (var conn = new SQLiteConnection(connectionString))
```

```
        {
```

```
            conn.Open();
```

```
            string query = "SELECT COUNT(*) FROM kedi_gebelik WHERE Telefon = @Telefon";
```

```
            using (var cmd = new SQLiteCommand(query, conn))
```

```
            {
```

```
                cmd.Parameters.AddWithValue("@Telefon", telefon);
```

```

        return (long)cmd.ExecuteScalar() > 0;

    }

}

catch (Exception ex)

{

    MessageBox.Show($"Telefon numarası kontrolü sırasında hata: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

    return false;

}

}

private void button1_Click(object sender, EventArgs e)

{

    try

    {

        string telefon = textBox1.Text.Trim();

        string cinsi = textBox2.Text.Trim();

        string adSoyad = textBox3.Text.Trim();

        if (string.IsNullOrEmpty(telefon) || string.IsNullOrEmpty(cinsi)
|| string.IsNullOrEmpty(adSoyad))

        {

            MessageBox.Show("Lütfen tüm alanları doldurun! (Telefon, Cinsi, Hayvan
Sahibi)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;

        }

        if (PhoneNumberExists(telefon))

        {

            MessageBox.Show("Bu telefon numarası zaten kayıtlı!", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);

```



```

        return;
    }

    string gebelikDurumu = checkBox1.Checked ? "Gebe" : checkBox2.Checked ?
"Gebe Değil" : "";

    DateTime? dogumTarihi = checkBox1.Checked ? DateTime.Now.AddDays(65) :
null;

    using (var conn = new SQLiteConnection(connectionString))
    {
        conn.Open();

        string query = @"
            INSERT INTO kedi_gebelik (Telefon, Cinsi, HayvanSahibi,
GebelikDurumu, KayitTarihi, DogumTarihi)
            VALUES (@Telefon, @Cinsi, @HayvanSahibi, @GebelikDurumu,
@KayitTarihi, @DogumTarihi)";

        using (var cmd = new SQLiteCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@Telefon", telefon);

            cmd.Parameters.AddWithValue("@Cinsi", cinsi);

            cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);

            cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);

            cmd.Parameters.AddWithValue("@KayitTarihi", DateTime.Now);

            cmd.Parameters.AddWithValue("@DogumTarihi", (object?)dogumTarihi ??
DBNull.Value);

            cmd.ExecuteNonQuery();
        }
    }

    MessageBox.Show("Kayıt başarıyla eklendi!", "Başarılı",
MessageBoxButtons.OK, MessageBoxIcon.Information);

```

```

        textBox1.Clear();

        textBox2.Clear();

        textBox3.Clear();

        checkBox1.Checked = false;

        checkBox2.Checked = false;


        LoadDataGridView();

    }

    catch (Exception ex)

    {

        MessageBox.Show($"Hata oluřtu: {ex.Message}", "Hata", MessageBoxButtons.OK,
        MessageBoxIcon.Error);

    }

}

private void button2_Click(object sender, EventArgs e)

{

    // Placeholder for button2_Click

}

private void button3_Click_1(object sender, EventArgs e)

{

    try

    {

        // Validate textboxes

        string telefon = textBox1.Text.Trim();

        string cinsi = textBox2.Text.Trim();

        string adSoyad = textBox3.Text.Trim();


        if (string.IsNullOrEmpty(telefon) || string.IsNullOrEmpty(cinsi)
        || string.IsNullOrEmpty(adSoyad))

```

```

    {
        MessageBox.Show("Lütfen tüm alanları doldurun! (Telefon, Cinsi, Hayvan Sahibi)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;
    }

    // Ensure a checkbox is selected

    if (!checkBox1.Checked && !checkBox2.Checked)
    {
        MessageBox.Show("Lütfen gebelik durumunu seçin! (Gebe veya Gebe Değil)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;
    }

    // Determine GebelikDurumu based on selected checkbox

    string gebelikDurumu = checkBox1.Checked ? "Gebe" : "Gebe Değil";

    DateTime? dogumTarihi = checkBox1.Checked ? DateTime.Now.AddDays(65) :
null;

    using (var conn = new SQLiteConnection(connectionString))
    {
        conn.Open();

        // Check if a record with this phone number exists

        if (PhoneNumberExists(telefon))
        {
            // Update the existing record based on the phone number

            string query = @"

                UPDATE kedi_gebelik

                SET Cinsi = @Cinsi,

                    HayvanSahibi = @HayvanSahibi,

```

```

        GebelikDurumu = @GebelikDurumu,

        DogumTarihi = @DogumTarihi

WHERE Telefon = @Telefon";

using (var cmd = new SQLiteCommand(query, conn))
{

    cmd.Parameters.AddWithValue("@Cinsi", cinsi);

    cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);

    cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);

    cmd.Parameters.AddWithValue("@DogumTarihi",
(object?)dogumTarihi ?? DBNull.Value);

    cmd.Parameters.AddWithValue("@Telefon", telefon);

    int rowsAffected = cmd.ExecuteNonQuery();

    if (rowsAffected > 0)
    {

        MessageBox.Show("Kayıt başarıyla güncellendi!", "Başarılı",
MessageBoxButtons.OK, MessageBoxIcon.Information);

    }

    else

    {

        MessageBox.Show("Güncelleme sırasında bir hata oluştu!",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;

    }

}

else

{

    // Optionally insert a new record if no record with this phone
number exists

    string query = @"

```

```
INSERT INTO kedi_gebelik (Telefon, Cinsi, HayvanSahibi,
GebelikDurumu, KayitTarihi, DogumTarihi)
VALUES (@Telefon, @Cinsi, @HayvanSahibi, @GebelikDurumu,
@KayitTarihi, @DogumTarihi)";
```

```
using (var cmd = new SQLiteCommand(query, conn))
{
    cmd.Parameters.AddWithValue("@Telefon", telefon);
    cmd.Parameters.AddWithValue("@Cinsi", cinsi);
    cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);
    cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);
    cmd.Parameters.AddWithValue("@KayitTarihi", DateTime.Now);
    cmd.Parameters.AddWithValue("@DogumTarihi",
(object?)dogumTarihi ?? DBNull.Value);
    cmd.ExecuteNonQuery();
}
```

```
MessageBox.Show("Telefon numarası bulunamadı, yeni kayıt eklendi!",
"Başarılı", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
}
```

```
}
```

```
// Clear form
```

```
textBox1.Clear();
```

```
textBox2.Clear();
```

```
textBox3.Clear();
```

```
checkBox1.Checked = false;
```

```
checkBox2.Checked = false;
```

```
// Refresh DataGridView
```

```
LoadDataGridView();
```

```
}
```

```

        catch (Exception ex)
        {
            MessageBox.Show($"Hata oluştu: {ex.Message}", "Hata", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        }
    }

    private void LoadDataGridView()
    {
        try
        {
            using (var conn = new SQLiteConnection(connectionString))
            {
                conn.Open();

                string query = "SELECT ID, Cinsi, GebelikDurumu, KayitTarihi,
                DogumTarihi, Telefon, HayvanSahibi FROM kedi_gebelik";

                using (var adapter = new SQLiteDataAdapter(query, conn))
                {
                    DataTable dt = new DataTable();

                    adapter.Fill(dt);

                    if (!dt.Columns.Contains("DogumaKalanSure"))
                    {
                        dt.Columns.Add("DogumaKalanSure", typeof(string));
                    }

                    dataGridView1.DataSource = dt;

                    UpdateDogumaKalanSure(dt);
                }
            }
        }
    }

```

```

        StyleDataGridView();
    }

    catch (Exception ex)
    {
        MessageBox.Show($"Veri yükleme hatası: {ex.Message}", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void UpdateDogumaKalanSure()
{
    if (dataGridView1.DataSource is DataTable dt)
    {
        UpdateDogumaKalanSure(dt);
        dataGridView1.Refresh();
    }
}

private void UpdateDogumaKalanSure(DataTable dt)
{
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        DataRow row = dt.Rows[i];

        if (i < dataGridView1.Rows.Count && !dataGridView1.Rows[i].IsNewRow)
        {
            DataGridViewRow dgvRow = dataGridView1.Rows[i];

            string gebelikDurumu = row["GebelikDurumu"]?.ToString() ?? "";

            if (string.IsNullOrEmpty(gebelikDurumu) || gebelikDurumu == "Gebe
Değil")

```

```

        {
            row["DogumaKalanSure"] = "Bilinmiyor";
            dgvRow.DefaultCellStyle.BackColor = Color.White;
        }

        else if (row["DogumTarihi"] != DBNull.Value && row["DogumTarihi"] is
DateTime dogumTarihi)
        {
            TimeSpan kalanSure = dogumTarihi - DateTime.Now;

            row["DogumaKalanSure"] = kalanSure.TotalSeconds > 0
                ? $"{kalanSure.Days} gün {kalanSure.Hours} saat
{kalanSure.Minutes} dakika {kalanSure.Seconds} saniye"
                : "Doğum zamanı geldi!";

            if (kalanSure.TotalSeconds <= 0)
                dgvRow.DefaultCellStyle.BackColor = Color.White;
            else if (kalanSure.TotalDays <= 15)
                dgvRow.DefaultCellStyle.BackColor = Color.Red;
            else
                dgvRow.DefaultCellStyle.BackColor = Color.Green;
        }
    }
    else
    {
        row["DogumaKalanSure"] = "Bilinmiyor";
        dgvRow.DefaultCellStyle.BackColor = Color.White;
    }
}
}
}
}

```

```

protected override void OnFormClosing(FormClosingEventArgs e)
{

```



```

        base.OnFormClosing(e);

        timer?.Stop();

        timer?.Dispose();
    }

    private void pictureBox6_Click(object sender, EventArgs e)
    {
        this.Hide();

        Form19 form19 = Application.OpenForms["Form19"] as Form19;

        if (form19 != null)
        {
            form19.Show();
        }
        else
        {
            form19 = new Form19();

            form19.Show();
        }
    }
}

```

```

CREATE TABLE "kedi_gebelik" (
    "ID"    INTEGER,
    "Cinsi"    TEXT NOT NULL,
    "GebelikDurumu"    TEXT NOT NULL,
    "KayitTarihi" DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "DogumTarihi" DATETIME,
    "DogumaKalanSure"    TEXT,
    "Telefon"    TEXT,
    "HayvanSahibi"    TEXT,

```

PRIMARY KEY("ID" AUTOINCREMENT)

);

## FORM 9 : KÖPEK GEBELİK

ID	Cinsi	GebelikDurum	KayıtTarihi	DogumTarihi	Telefon
1	Golden Retri...	Gebe Değil	1.05.2025 23:34		05551234
2	a	Gebe	2.05.2025 18:11	4.07.2025 18:11	0535300

```
using System;
```

```
using System.Data;
```

```
using System.Drawing;
```

```
using System.Windows.Forms;
```

```
using System.Data.SQLite;
```

```
namespace WinFormsApp8
```

```
{
```

```
    public partial class Form19 : Form
```

```
    {
```

```
        private readonly string connectionString = @"Data
Source=C:\Users\ridva\Desktop\SQLITE3_VETERİNEROTOMASYON\veteriner_otomasyon.db;Version=3;"
        ;
```

```
        private System.Windows.Forms.Timer? timer;
```

```
public Form19()
{
    InitializeComponent();

    InitializeCheckBoxes();

    InitializeTimer();

    LoadDataGridView();

    StyleDataGridView();

    dataGridView1.SelectionChanged += DataGridView1_SelectionChanged;
}
```

```
private void InitializeCheckBoxes()
{
    checkBox1.CheckedChanged += (s, e) =>
    {
        if (checkBox1.Checked) checkBox2.Checked = false;
    };

    checkBox2.CheckedChanged += (s, e) =>
    {
        if (checkBox2.Checked) checkBox1.Checked = false;
    };
}
```

```
private void InitializeTimer()
{
    timer = new System.Windows.Forms.Timer { Interval = 1000 };

    timer.Tick += (s, e) => UpdateDogumaKalanSure();

    timer.Start();
}
```

```
private void StyleDataGridView()
```

```

{

    dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI", 11,
FontStyle.Bold);

    dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.White;

    dataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.FromArgb(33, 150,
243);

    dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;

    dataGridView1.EnableHeadersVisualStyles = false;

    dataGridView1.DefaultCellStyle.Font = new Font("Segoe UI Semibold", 10,
FontStyle.Regular);

    dataGridView1.DefaultCellStyle.ForeColor = Color.Black;

    dataGridView1.DefaultCellStyle.SelectionBackColor = Color.LightBlue;

    dataGridView1.DefaultCellStyle.SelectionForeColor = Color.Black;

    dataGridView1.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;

    dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.None;

    if (dataGridView1.Columns.Count > 0)
    {

        if (dataGridView1.Columns.Contains("ID")) dataGridView1.Columns["ID"].Width
= 80;

        if (dataGridView1.Columns.Contains("Telefon"))
dataGridView1.Columns["Telefon"].Width = 100;

        if (dataGridView1.Columns.Contains("Cinsi"))
dataGridView1.Columns["Cinsi"].Width = 100;

        if (dataGridView1.Columns.Contains("HayvanSahibi"))
dataGridView1.Columns["HayvanSahibi"].Width = 150;

        if (dataGridView1.Columns.Contains("GebelikDurumu"))
dataGridView1.Columns["GebelikDurumu"].Width = 100;

        if (dataGridView1.Columns.Contains("KayitTarihi"))
dataGridView1.Columns["KayitTarihi"].Width = 150;

        if (dataGridView1.Columns.Contains("DogumTarihi"))
dataGridView1.Columns["DogumTarihi"].Width = 150;
    }
}

```

```

        if (dataGridView1.Columns.Contains("DogumaKalanSure"))
dataGridView1.Columns["DogumaKalanSure"].Width = 150;

    }

    dataGridView1.Visible = true;
}

private void DataGridView1_SelectionChanged(object? sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

        textBox1.Text = selectedRow.Cells["Telefon"].Value?.ToString() ?? "";
        textBox2.Text = selectedRow.Cells["Cinsi"].Value?.ToString() ?? "";
        textBox3.Text = selectedRow.Cells["HayvanSahibi"].Value?.ToString() ?? "";

        string gebelikDurumu = selectedRow.Cells["GebelikDurumu"].Value?.ToString()
?? "";

        checkBox1.Checked = gebelikDurumu == "Gebe";
        checkBox2.Checked = gebelikDurumu == "Gebe Değil";
    }
}

private bool PhoneNumberExists(string telefon)
{
    try
    {
        using (var conn = new SQLiteConnection(connectionString))
        {
            conn.Open();

```

```

        string query = "SELECT COUNT(*) FROM kopek_gebelik WHERE Telefon =
@Telefon";

        using (var cmd = new SQLiteCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@Telefon", telefon);

            return (long)cmd.ExecuteScalar() > 0;
        }
    }

    catch (Exception ex)
    {
        MessageBox.Show($"Telefon numarası kontrolü sırasında hata: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return false;
    }
}

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        string telefon = textBox1.Text.Trim();

        string cinsi = textBox2.Text.Trim();

        string adSoyad = textBox3.Text.Trim();

        if (string.IsNullOrEmpty(telefon) || string.IsNullOrEmpty(cinsi)
|| string.IsNullOrEmpty(adSoyad))
        {
            MessageBox.Show("Lütfen tüm alanları doldurun! (Telefon, Cinsi, Hayvan
Sahibi)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }
    }
}

```

```

        if (PhoneNumberExists(telefon))
        {
            MessageBox.Show("Bu telefon numarası zaten kayıtlı!", "Hata",
                MessageBoxButtons.OK, MessageBoxIcon.Error);

            return;
        }

        string gebelikDurumu = checkBox1.Checked ? "Gebe" : checkBox2.Checked ?
"Gebe Değil" : "";

        DateTime? dogumTarihi = checkBox1.Checked ? DateTime.Now.AddDays(63) :
null;

        using (var conn = new SQLiteConnection(connectionString))
        {
            conn.Open();

            string query = @"
                INSERT INTO kopek_gebelik (Telefon, Cinsi, HayvanSahibi,
                GebelikDurumu, KayitTarihi, DogumTarihi)
                VALUES (@Telefon, @Cinsi, @HayvanSahibi, @GebelikDurumu,
                @KayitTarihi, @DogumTarihi)";

            using (var cmd = new SQLiteCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@Telefon", telefon);

                cmd.Parameters.AddWithValue("@Cinsi", cinsi);

                cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);

                cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);

                cmd.Parameters.AddWithValue("@KayitTarihi", DateTime.Now);

                cmd.Parameters.AddWithValue("@DogumTarihi", (object?)dogumTarihi ??
DBNull.Value);

                cmd.ExecuteNonQuery();
            }
        }

```

```
}
```

```
        MessageBox.Show("Kayıt başarıyla eklendi!", "Başarılı",  
        MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
        textBox1.Clear();
```

```
        textBox2.Clear();
```

```
        textBox3.Clear();
```

```
        checkBox1.Checked = false;
```

```
        checkBox2.Checked = false;
```

```
        LoadDataGridView();
```

```
    }
```

```
    catch (Exception ex)
```

```
    {
```

```
        MessageBox.Show($"Hata oluştu: {ex.Message}", "Hata", MessageBoxButtons.OK,  
        MessageBoxIcon.Error);
```

```
    }
```

```
}
```

```
private void button2_Click(object sender, EventArgs e)
```

```
{
```

```
    try
```

```
    {
```

```
        string telefon = textBox1.Text.Trim();
```

```
        string cinsi = textBox2.Text.Trim();
```

```
        string adSoyad = textBox3.Text.Trim();
```

```
        if (string.IsNullOrEmpty(telefon) || string.IsNullOrEmpty(cinsi)  
        || string.IsNullOrEmpty(adSoyad))
```

```
        {
```



```

        MessageBox.Show("Lütfen tüm alanları doldurun! (Telefon, Cinsi, Hayvan Sahibi)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;
    }

    if (!checkBox1.Checked && !checkBox2.Checked)
    {
        MessageBox.Show("Lütfen gebelik durumunu seçin! (Gebe veya Gebe Değil)", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;
    }

    string gebelikDurumu = checkBox1.Checked ? "Gebe" : "Gebe Değil";

    DateTime? dogumTarihi = checkBox1.Checked ? DateTime.Now.AddDays(63) :
null;

    using (var conn = new SQLiteConnection(connectionString))
    {
        conn.Open();

        if (PhoneNumberExists(telefon))
        {
            string query = @"
                UPDATE kopek_gebelik
                SET Cinsi = @Cinsi,
                    HayvanSahibi = @HayvanSahibi,
                    GebelikDurumu = @GebelikDurumu,
                    DogumTarihi = @DogumTarihi
                WHERE Telefon = @Telefon";

            using (var cmd = new SQLiteCommand(query, conn))

```

```

        {

            cmd.Parameters.AddWithValue("@Cinsi", cinsi);

            cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);

            cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);

            cmd.Parameters.AddWithValue("@DogumTarihi",
(object?)dogumTarihi ?? DBNull.Value);

            cmd.Parameters.AddWithValue("@Telefon", telefon);

            int rowsAffected = cmd.ExecuteNonQuery();

            if (rowsAffected > 0)

            {

                MessageBox.Show("Kayıt başarıyla güncellendi!", "Başarılı",
MessageBoxButtons.OK, MessageBoxIcon.Information);

            }

            else

            {

                MessageBox.Show("Güncelleme sırasında bir hata oluştu!",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);

                return;

            }

        }

    }

    else

    {

        string query = @"

            INSERT INTO kopek_gebelik (Telefon, Cinsi, HayvanSahibi,
GebelikDurumu, KayitTarihi, DogumTarihi)

            VALUES (@Telefon, @Cinsi, @HayvanSahibi, @GebelikDurumu,
@KayitTarihi, @DogumTarihi)";

        using (var cmd = new SQLiteCommand(query, conn))

        {

            cmd.Parameters.AddWithValue("@Telefon", telefon);

```

```

        cmd.Parameters.AddWithValue("@Cinsi", cinsi);

        cmd.Parameters.AddWithValue("@HayvanSahibi", adSoyad);

        cmd.Parameters.AddWithValue("@GebelikDurumu", gebelikDurumu);

        cmd.Parameters.AddWithValue("@KayitTarihi", DateTime.Now);

        cmd.Parameters.AddWithValue("@DogumTarihi",
(object?)dogumTarihi ?? DBNull.Value);

        cmd.ExecuteNonQuery();

    }

    MessageBox.Show("Telefon numarası bulunamadı, yeni kayıt eklendi!",
"Başarılı", MessageBoxButtons.OK, MessageBoxIcon.Information);

    }

}

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

checkBox1.Checked = false;

checkBox2.Checked = false;

LoadDataGridView();

}

catch (Exception ex)

{

    MessageBox.Show($"Hata oluştu: {ex.Message}", "Hata", MessageBoxButtons.OK,
MessageBoxIcon.Error);

}

}

private void LoadDataGridView()

{

```

```

try
{
    using (var conn = new SQLiteConnection(connectionString))
    {
        conn.Open();

        string query = "SELECT ID, Cinsi, GebelikDurumu, KayitTarihi,
DogumTarihi, Telefon, HayvanSahibi FROM kopek_gebelik";

        using (var adapter = new SQLiteDataAdapter(query, conn))
        {
            DataTable dt = new DataTable();

            adapter.Fill(dt);

            if (!dt.Columns.Contains("DogumaKalanSure"))
            {
                dt.Columns.Add("DogumaKalanSure", typeof(string));
            }

            dataGridView1.DataSource = dt;

            UpdateDogumaKalanSure(dt);
        }
    }

    StyleDataGridView();
}

catch (Exception ex)
{
    MessageBox.Show($"Veri yükleme hatası: {ex.Message}", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

```

```
private void UpdateDogumaKalanSure()
```

```
{
```

```
    if (dataGridView1.DataSource is DataTable dt)
```

```
    {
```

```
        UpdateDogumaKalanSure(dt);
```

```
        dataGridView1.Refresh();
```

```
    }
```

```
}
```

```
private void UpdateDogumaKalanSure(DataTable dt)
```

```
{
```

```
    for (int i = 0; i < dt.Rows.Count; i++)
```

```
    {
```

```
        DataRow row = dt.Rows[i];
```

```
        if (i < dataGridView1.Rows.Count && !dataGridView1.Rows[i].IsNewRow)
```

```
        {
```

```
            DataGridViewRow dgvRow = dataGridView1.Rows[i];
```

```
            string gebelikDurumu = row["GebelikDurumu"]?.ToString() ?? "";
```

```
            if (string.IsNullOrEmpty(gebelikDurumu) || gebelikDurumu == "Gebe  
Değil")
```

```
            {
```

```
                row["DogumaKalanSure"] = "Bilinmiyor";
```

```
                dgvRow.DefaultCellStyle.BackColor = Color.White;
```

```
            }
```

```
            else if (row["DogumTarihi"] != DBNull.Value && row["DogumTarihi"] is  
DateTime dogumTarihi)
```

```
            {
```

```
                TimeSpan kalanSure = dogumTarihi - DateTime.Now;
```

```
                row["DogumaKalanSure"] = kalanSure.TotalSeconds > 0
```

```
                ? $"{kalanSure.Days} gün {kalanSure.Hours} saat  
{kalanSure.Minutes} dakika {kalanSure.Seconds} saniye"
```

```
                : "Doğum zamanı geldi!";
```

```
            if (kalanSure.TotalSeconds <= 0)
```

```
                dgvRow.DefaultCellStyle.BackColor = Color.White;
```

```
            else if (kalanSure.TotalDays <= 15)
```

```
                dgvRow.DefaultCellStyle.BackColor = Color.Red;
```

```
            else
```

```
                dgvRow.DefaultCellStyle.BackColor = Color.Green;
```

```
        }
```

```
    else
```

```
    {
```

```
        row["DogumaKalanSure"] = "Bilinmiyor";
```

```
        dgvRow.DefaultCellStyle.BackColor = Color.White;
```

```
    }
```

```
    }
```

```
    }
```

```
}
```

```
protected override void OnFormClosing(FormClosingEventArgs e)
```

```
{
```

```
    base.OnFormClosing(e);
```

```
    timer?.Stop();
```

```
    timer?.Dispose();
```

```
}
```

```
}
```

```
}
```

```
CREATE TABLE "kopek_gebelik" (  
    "ID"    INTEGER,  
    "Cinsi"    TEXT NOT NULL,  
    "GebelikDurumu"    TEXT NOT NULL,  
    "KayitTarihi" DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    "DogumTarihi" DATETIME,  
    "DogumaKalanSure"    TEXT,  
    "Telefon"    TEXT,  
    "HayvanSahibi"    TEXT,  
    PRIMARY KEY("ID" AUTOINCREMENT)  
);
```

---

FORM:20 :HAYVAN HASTALIK TAHMİNİ

HAYVAN HASTALIK TAHMINI

hayvan secimi

☐ KEDI
☒ KÖPEK

Hastalik Adı:

Hastalik Belirtileri:

Hırıltı, nefes alma zorluğu, öksürük

Hastalik Tedavisi:

button1

BUL

id	kedi_hastalik_adi	kedi_belirtiler	kedi_tedavi	kopek_hastalik_adi	kopek_belirtiler	kopek_tedavi
13	Şeker Hastalığı...	Çok su içme, ç...	İnsülin enjeksiy...			
14	İç Parazitler (Sol...	Kusma, ishal, k...	Düzenli parazit l...			
15	Dış Parazitler (P...	Kapıntı, tüy dok...	Pire daniyası, an...			
16	Mantar Enfeksi...	Tüy dökülmesi...	Mantar kremi, ...			
17	Kulak Uyuşu	Kulakta siyah a...	Kulak temizliği, ...			
18	Alerjik Dermato...	Kapıntı, kızamık...	Alerjenin tespiti...			
19	Kedi Astımı	Hırıltı, nefes al...	Kortikosteroidle...			
20	Kalp Hastalıklar...	Hızlı solunum, ...	Kalp ilaçları, dü...			
21	Toksoplazmozis	İshal, ateş, kas e...	Antibiyotik ted...			

HASTALIK TESPİT RAPORU

Rapor Tarihi: 03.05.2025 01:26:47

Hayvan Türü: Köpek

Girilen Belirtiler: hırıltı, nefes alma zorluğu, öksürük

En Yüksek İhtimalle 3 Hastalık ve Tedaviler

Hastalık Adı

Olasılık (%)

Tedavi

Canine Bronchitis 60,0

Antibiyotikler, bronkodilatörle

Canine Bronchitis 60,0

Antibiyotikler, bronkodilatörle

Canine Bronchitis 60,0

Antibiyotikler, bronkodilatörle

Veteriner Otomasyon Sistemi

```

using System;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Microsoft.Data.Sqlite;

namespace WinFormsApp8
{
    public partial class Form20 : Form
    {
        private readonly string connectionString = @"Data
Source=C:\Users\ridva\Desktop\SQLITE3_VETERİNEROTOMASYON\veteriner_otomasyon.db;";

        private Panel? label5Panel;

        public Form20()
        {
            InitializeComponent();

```



```
CreateTableIfNotExists();
```

```
LoadDataGridView();
```

```
checkBox1.CheckedChanged += CheckBox1_CheckedChanged;
```

```
checkBox2.CheckedChanged += CheckBox2_CheckedChanged;
```

```
button1.Click += button1_Click;
```

```
button2.Click += button2_Click;
```

```
SetupLabel5Panel();
```

```
this.AutoScroll = true;
```

```
label5.Text = "Sonuçlar burada görüntülenecek. Lütfen belirtileri arayın.";
```

```
}
```

```
private void SetupLabel5Panel()
```

```
{
```

```
    Point label5Location = new Point(dataGridView1.Location.X,  
dataGridView1.Location.Y + dataGridView1.Height + 10);
```

```
    Size label5Size = new Size(500, 100);
```

```
    label5Panel = new Panel
```

```
    {
```

```
        Location = label5Location,
```

```
        Size = label5Size,
```

```
        AutoScroll = true,
```

```
        BorderStyle = BorderStyle.FixedSingle
```

```
    };
```

```
space    label5.Location = new Point(0, 0); // No padding on left or top to maximize
```

```
    label5.AutoSize = false;
```

```
        label5.Size = new Size(label5Panel.ClientSize.Width, 80); // Initially no  
scrollbar adjustment
```

```
        label5.BackColor = Color.LightGray;
```

```
        label5.ForeColor = Color.DarkBlue;
```

```
        label5.Font = new Font("Consolas", 12, FontStyle.Regular);
```

```
        label5.Padding = new Padding(5, 5, 0, 5); // Remove right padding
```

```
        label5.TextAlign = ContentAlignment.TopLeft;
```

```
        label5Panel.Paint += (sender, e) =>
```

```
{
```

```
    using (Pen pen = new Pen(Color.Black, 2))
```

```
{
```

```
        e.Graphics.DrawRectangle(pen, 0, 0, label5Panel.Width - 1,  
label5Panel.Height - 1);
```

```
}
```

```
};
```

```
        label5Panel.Controls.Add(label5);
```

```
        this.Controls.Add(label5Panel);
```

```
}
```

```
private void CheckBox1_CheckedChanged(object? sender, EventArgs e)
```

```
{
```

```
    if (checkBox1.Checked) checkBox2.Checked = false;
```

```
}
```

```
private void CheckBox2_CheckedChanged(object? sender, EventArgs e)
```

```
{
```

```
    if (checkBox2.Checked) checkBox1.Checked = false;
```

```
}
```

```

private void CreateTableIfNotExists()
{
    try
    {
        using (var conn = new SQLiteConnection(connectionString))
        {
            conn.Open();

            string query = @"

                CREATE TABLE IF NOT EXISTS HayvanHastaliklari (

                    id INTEGER PRIMARY KEY AUTOINCREMENT,

                    kedi_hastalik_adi TEXT,

                    kedi_belirtiler TEXT,

                    kedi_tedavi TEXT,

                    kopek_hastalik_adi TEXT,

                    kopek_belirtiler TEXT,

                    kopek_tedavi TEXT

                )";

            using (var cmd = new SQLiteCommand(query, conn))
            {
                cmd.ExecuteNonQuery();
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Tablo oluşturulamadı: {ex.Message}", "Hata",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

private void LoadDataGridView()
{
    try
    {
        using (var conn = new SqlConnection(connectionString))
        {
            conn.Open();

            string query = "SELECT * FROM HayvanHastaliklari";

            using (var cmd = new SqlCommand(query, conn))
            using (var reader = cmd.ExecuteReader())
            {
                DataTable dt = new DataTable();

                dt.Load(reader);

                dataGridView1.DataSource = dt;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Veri yüklenemedi: {ex.Message}", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void button1_Click(object? sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox1.Text) ||
        string.IsNullOrEmpty(richTextBox1.Text) ||
        string.IsNullOrEmpty(richTextBox2.Text))
    {

```

```
        MessageBox.Show("Lütfen tüm alanları doldurun!", "Hata",  
        MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        return;
```

```
    }
```

```
    if (!checkBox1.Checked && !checkBox2.Checked)
```

```
    {
```

```
        MessageBox.Show("Lütfen hayvan türünü seçin!", "Hata",  
        MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        return;
```

```
    }
```

```
    string query = checkBox1.Checked
```

```
        ? "INSERT INTO HayvanHastaliklari (kedi_hastalik_adi, kedi_belirtiler,  
        kedi_tedavi) VALUES (@adi, @belirti, @tedavi)"
```

```
        : "INSERT INTO HayvanHastaliklari (kopek_hastalik_adi, kopek_belirtiler,  
        kopek_tedavi) VALUES (@adi, @belirti, @tedavi)";
```

```
    try
```

```
    {
```

```
        using (var conn = new SqlConnection(connectionString))
```

```
        {
```

```
            conn.Open();
```

```
            using (var cmd = new SqlCommand(query, conn))
```

```
            {
```

```
                cmd.Parameters.AddWithValue("@adi", textBox1.Text.Trim());
```

```
                cmd.Parameters.AddWithValue("@belirti", richTextBox1.Text.Trim());
```

```
                cmd.Parameters.AddWithValue("@tedavi", richTextBox2.Text.Trim());
```

```
                cmd.ExecuteNonQuery();
```

```
            }
```

```
        }
```

```
        MessageBox.Show("Kayıt eklendi!", "Bilgi", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

        LoadDataGridView();

        textBox1.Clear();

        richTextBox1.Clear();

        richTextBox2.Clear();

        checkBox1.Checked = false;

        checkBox2.Checked = false;

    }

    catch (Exception ex)

    {

        MessageBox.Show($"Kayıt eklenemedi: {ex.Message}", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);

    }

}

private void button2_Click(object? sender, EventArgs e)

{

    if (!checkBox1.Checked && !checkBox2.Checked)

    {

        MessageBox.Show("Lütfen kedi veya köpek seçin!", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;

    }

    string girilenBelirti = richTextBox1.Text.Trim().ToLower();

    if (string.IsNullOrEmpty(girilenBelirti))

    {

        MessageBox.Show("Belirti giriniz!", "Hata", MessageBoxButtons.OK,
        MessageBoxIcon.Error);

        return;

    }

}
```

```
}
```

```
string hayvanTuru = checkBox1.Checked ? "Kedi" : "Köpek";
```

```
string query = "SELECT * FROM HayvanHastaliklari";
```

```
var sonucListesi = new System.Collections.Generic.List<(string Hastalik, double  
Yuzde, string? Tedavi)>();
```

```
try
```

```
{
```

```
    using (var conn = new SqlConnection(connectionString))
```

```
    {
```

```
        conn.Open();
```

```
        using (var cmd = new SqlCommand(query, conn))
```

```
        using (var reader = cmd.ExecuteReader())
```

```
        {
```

```
            while (reader.Read())
```

```
            {
```

```
                string? hastalikAdi = null;
```

```
                string? belirtiler = null;
```

```
                string? tedavi = null;
```

```
                if (checkBox1.Checked)
```

```
                {
```

```
                    hastalikAdi = reader["kedi_hastalik_adi"]?.ToString();
```

```
                    belirtiler = reader["kedi_belirtiler"]?.ToString();
```

```
                    tedavi = reader["kedi_tedavi"]?.ToString();
```

```
                }
```

```
            else
```

```
            {
```

```
                hastalikAdi = reader["kopek_hastalik_adi"]?.ToString();
```

```

        belirtiler = reader["kopek_belirtiler"]?.ToString();

        tedavi = reader["kopek_tedavi"]?.ToString();

    }

    if (!string.IsNullOrEmpty(hastalikAdi) &&
        !string.IsNullOrEmpty(belirtiler))
    {
        string[] girilenKelimeler = girilenBelirti.Split(new[] { ' '
        ', ', '.', '; ' }, StringSplitOptions.RemoveEmptyEntries);

        int ortak = girilenKelimeler.Count(k =>
        belirtiler.ToLower().Contains(k));

        double yuzde = (double)ortak / girilenKelimeler.Length *
        100;

        if (yuzde > 0)
        {
            sonucListesi.Add((hastalikAdi, yuzde, tedavi));
        }
    }
}

UpdateLabel5(hayvanTuru, girilenBelirti, sonucListesi);
}

catch (Exception ex)
{
    MessageBox.Show($"Arama sırasında hata: {ex.Message}", "Hata",
    MessageBoxButtons.OK, MessageBoxIcon.Error);

    label5.Text = "Arama sırasında hata oluştu.";
}
}

```



```

        private void UpdateLabel5(string hayvanTuru, string girilenBelirti,
System.Collections.Generic.List<(string Hastalik, double Yuzde, string? Tedavi)>
sonucListesi)

    {

        if (label5Panel == null) return;

        StringBuilder rapor = new StringBuilder();

        rapor.AppendLine("=====");

        rapor.AppendLine("      HASTALIK TESPİT RAPORU      ");

        rapor.AppendLine("=====");

        rapor.AppendLine($"Rapor Tarihi: {DateTime.Now:dd.MM.yyyy HH:mm:ss}");

        rapor.AppendLine($"Hayvan Türü: {hayvanTuru}");

        rapor.AppendLine($"Girilen Belirtiler: {girilenBelirti}");

        rapor.AppendLine();

        if (!sonucListesi.Any())

        {

            rapor.AppendLine("-----");

            rapor.AppendLine("SONUÇ: Benzer hastalık bulunamadı.");

            rapor.AppendLine("-----");

        }

        else

        {

            var top3Sonuclar = sonucListesi.OrderByDescending(x =>
x.Yuzde).Take(3).ToList();

            rapor.AppendLine("-----");

            rapor.AppendLine("En Yüksek İhtimalle 3 Hastalık ve Tedaviler");

            rapor.AppendLine("-----");

            // Calculate column widths based on content

            int maxHastalikLength = top3Sonuclar.Max(x => x.Hastalik.Length);

```

```

        int maxYuzdeLength = top3Sonuclar.Max(x => $"{x.Yuzde:F1}".Length);

        maxHastalikLength = Math.Max(maxHastalikLength, "Hastalik Adı".Length);

        maxYuzdeLength = Math.Max(maxYuzdeLength, "Olasılık (%)".Length);

        // Header

        rapor.AppendLine($"{("Hastalik Adı").PadRight(maxHastalikLength)}
{"Olasılık (%)".PadRight(maxYuzdeLength)} Tedavi");

        rapor.AppendLine(new string('-', maxHastalikLength + maxYuzdeLength + "
Tedavi".Length + 2));

        // Data rows

        foreach (var (hastalik, yuzde, tedavi) in top3Sonuclar)
        {

            string yuzdeStr = $"{yuzde:F1}";

            string tedaviStr = tedavi ?? "Tedavi bilgisi yok";

            rapor.AppendLine($"{hastalik.PadRight(maxHastalikLength)}
{yuzdeStr.PadRight(maxYuzdeLength)} {tedaviStr}");

        }

    }

    rapor.AppendLine();

    rapor.AppendLine("=====");

    rapor.AppendLine("Veteriner Otomasyon Sistemi");

    rapor.AppendLine("=====");

    // Calculate dimensions based on content

    int baseHeight = 100;

    int lineHeight = 20;

    int numberOfLines = rapor.ToString().Split('\n').Length;

    int newHeight = baseHeight + (numberOfLines * lineHeight);

    int maxHeight = 600;

```

```

newHeight = Math.Min(newHeight, maxHeight);

// Calculate the exact width needed for the text
string[] lines = rapor.ToString().Split('\n');
using (Graphics g = label5.CreateGraphics())
{
    int newWidth = 0;

    foreach (string line in lines)
    {
        if (string.IsNullOrEmpty(line)) continue;

        SizeF textSize = g.MeasureString(line, label5.Font);

        newWidth = Math.Max(newWidth, (int)textSize.Width);
    }

    newWidth += 10; // Small padding for safety

    newWidth = Math.Max(500, newWidth); // Ensure minimum width

    newWidth = Math.Min(newWidth, 800); // Cap maximum width

    label5Panel.Size = new Size(newWidth, newHeight);

    label5.Size = new Size(newWidth, newHeight); // No scrollbar adjustment
}

label5.Text = rapor.ToString();

label5Panel.Invalidate();
}
}
}

```

```

CREATE TABLE "HayvanHastaliklari" (

    "id"    INTEGER,

    "kedi_hastalik_adi" TEXT,

```

```
"kedi_belirtiler"    TEXT,  
  
"kedi_tedavi" TEXT,  
  
"kopek_hastalik_adi" TEXT,  
  
"kopek_belirtiler"   TEXT,  
  
"kopek_tedavi"       TEXT,  
  
PRIMARY KEY("id" AUTOINCREMENT)
```

```
);
```

---