1. SQLite, sunucuya ihtiyaç duymayan, tek dosyada veri saklayan, kurulumu kolay ve taşınabilir bir veritabanıdır. Özellikle küçük ve orta ölçekli uygulamalarda, sunucu altyapısına ihtiyaç duymaması nedeniyle tercih edilir.

2. Projeye SQLite desteği eklemek için "System.Data.SQLite" paketi yüklenir.

3. Örnek bir veritabanına veri ekleme:

```
private void btnAddBook_Click(object sender, EventArgs e)
{
    // SQLite veritabanı dosyamızın bağlantı dizesi (örneğin: "Data Source=Kitaplar.db;Version=3;"
    string connectionString = "Data Source=Kitaplar.db;Version=3;";

    using (SQLiteConnection con = new SQLiteConnection(connectionString))
    {
        con.Open();
        // INSERT sorgusu: Kitaplar tablosuna yeni kayıt ekleniyor.
        string query = "INSERT INTO Kitaplar (KitapAdi, Yazar, Yayinevi, SayfaSayisi) " +
                       "VALUES (@KitapAdi, @Yazar, @Yayinevi, @SayfaSayisi)";
        using (SQLiteCommand cmd = new SQLiteCommand(query, con))
        {
            // Parametreleri ekliyoruz:
            cmd.Parameters.AddWithValue("@KitapAdi", txtKitapAdi.Text);
            cmd.Parameters.AddWithValue("@Yazar", txtYazar.Text);
            cmd.Parameters.AddWithValue("@Yayinevi", txtYayinevi.Text);
            cmd.Parameters.AddWithValue("@SayfaSayisi", int.Parse(txtSayfaSayisi.Text));

            // Sorguyu çalıştırıyoruz:
            cmd.ExecuteNonQuery();
        }
        MessageBox.Show("Kitap başarıyla eklendi!");
        // Eğer kitapları listeleyen bir metot varsa, burada çağırarak güncellemeyi sağlayabilirsi
        LoadBooks();
    }
```

Açıklamalar

- **SQLiteConnection:**
  Belirtilen bağlantı dizesiyle (örneğin, "Kitaplar.db" dosyası) veritabanına erişimi sağlar.
- **Open():**
  Bağlantıyı açmak için kullanılır. Bağlantı açılmadan sorgular çalıştırılamaz.
- **SQLiteCommand:**
  INSERT sorgusunu tanımlar. Parametreler (@KitapAdi, @Yazar, vb.) sayesinde SQL Injection gibi problemlere karşı güvenli bir yöntem sunar.
- **AddWithValue:**
  Parametreye, ilgili kontrol (örneğin, txtKitapAdi.Text) değerini atar.
- **ExecuteNonQuery:**
  Sorgunun çalıştırılmasını sağlar. Bu metot, veri ekleme, güncelleme veya silme gibi sonuç döndürmeyen sorgularda kullanılır.
- **LoadBooks():**
  Bu metodun amacı, ekleme işleminden sonra DataGridView gibi bir kontrolü güncelleyerek, veritabanındaki kitapları tekrar listelemektir. (Ramazan örneğinde LoadChecklist() metodu kullanılmıştı.)

4. Örnek bir veritabanından veri listeleme:

```
private void LoadBooks()
{
    string connectionString = "Data Source=Kitaplar.db;Version=3;";
    using (SQLiteConnection con = new SQLiteConnection(connectionString))
    {
        con.Open();
        // Kitaplar tablosundaki tüm kayıtları çekmek için SELECT sorgusu
        string query = "SELECT * FROM Kitaplar";
        using (SQLiteDataAdapter da = new SQLiteDataAdapter(query, con))
        {
            DataTable dt = new DataTable();
            // Fill metodu, SELECT sorgusunun sonuçlarını DataTable'a doldurur.
            da.Fill(dt);
            // DataGridView'in DataSource özelliğine DataTable'ı atayarak veriler ekranda görüntül
            dgvBooks.DataSource = dt;
        }
    }
}
```

Açıklamalar

- **SQLiteConnection:**
  Belirtilen bağlantı dizesiyle veritabanı dosyasına erişim sağlar.
- **Open():**
  Bağlantıyı açmak için kullanılır.
- **SQLiteDataAdapter:**
  "SELECT * FROM Kitaplar" sorgusunu çalıştırır ve veritabanından çekilen verileri DataTable nesnesine doldurur.
- **DataTable:**
  Verilerin bellekte tablo şeklinde saklandığı nesnedir.
- **Fill():**
  DataAdapter'ın sorgu sonuçlarını DataTable'a aktarır.
- **DataSource:**
  DataGridView'in DataSource özelliğine atanan DataTable, verilerin ekranda görüntülenmesini sağlar.

Bu metodu, kitap ekleme işleminden sonra ya da "Listele" butonuna tıklanıldığında çağırarak güncel verilerin DataGridView üzerinde görünmesini sağlayabilirsiniz.