Öğrenci Adı Soyadı : EMİNE YİĞİT

Öğrenci Numarası: 100919015

T.C. MARMARA ÜNİVERSİTESİ TEKNOLOJİ FAKÜLTESİ
2024-2025 EĞİTİM ÖĞRETİM YILI
GÜZ DÖNEMİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
VERİTABANI YÖNETİM SİSTEMLERİ DERSİ
PROJE-FİNAL DÖKÜMANI

Tüm sorular soru metnini altındaki alanlara cevaplandırılacaktır. Bu belgenin formatını bozmayınız, sadece gerekli alanları doldurunuz.

1. **Gerçekleştirdiğiniz veri tabanı projesi için grup arkadaşlarınızın isimlerini yazınız ve projenize ait veri tabanı/diğer yazılım bileşenleri hakkında bilgi veriniz. (5 p)**

- Büşra AY
- Tankut Arca CAN

Projemizde, veri tabanı bileşeni olarak Microsoft SQL Server kullanılmıştır. Veritabanı, çiftlik yönetimi, hasat, ürün bilgisi, satışlar ve lojistik süreçlerini kapsayan tablolarla tasarlanmıştır. Çiftlikler, ürünler, müşteri bilgileri gibi temel tablolar arasında birincil anahtar-foreign key ilişkileri tanımlanarak veri bütünlüğü sağlanmıştır. Yazılım bileşenleri tarafında, Python dili kullanılarak CustomTkinter kütüphanesi ile modern bir kullanıcı arayüzü geliştirilmiştir. Uygulama, kullanıcı doğrulama (şifreleme ve giriş işlemleri), dinamik MSSQL sorgu desteği, CRUD işlemleri (ekleme, güncelleme, silme) ve raporlama yetenekleri sunmaktadır.

2. **Gerçekleştirdiğiniz veri tabanı projesi için proje dokümanınızı ve dosyalarınızı içeren herkese açık github bağlantılarınızı paylaşınız. (5 p)**

- https://github.com/coderemine/TarimTakipUygulamasi

3. **Gerçekleştirdiğiniz projenin amacını detaylı bir şekilde açıklayınız. (10 p)**

Bu projenin amacı, tarım sektöründe faaliyet gösteren bireyler veya kurumlar için dijital bir tarım ürünleri takip sistemi geliştirmektir. Tarımsal süreçlerin izlenmesi ve yönetimi, modern teknolojilerle daha verimli ve organize bir şekilde gerçekleştirilebilir. Bu proje, bu ihtiyacı karşılamak için kullanıcı dostu bir arayüz ve güçlü bir veri tabanı altyapısı sağlar.

1.Tarımsal Verilerin Merkezi Yönetimi:

- Çiftlik, ürün, hasat, satış ve lojistik süreçleri gibi tarımsal faaliyetlerle ilgili tüm verilerin tek bir sistemde toplanması.
- Tarımsal operasyonların verimli ve doğru bir şekilde yönetilmesi.

2.Kullanıcı Dostu Arayüz Sağlama: CustomTkinter kütüphanesi ile oluşturulan modern ve sezgisel arayüz sayesinde kullanıcılar, teknik bilgiye ihtiyaç duymadan sistemi kolayca kullanabilir.

3.Güvenli Kullanıcı Yönetimi:

- Kullanıcı giriş ve kayıt işlemlerinin güvenli bir şekilde gerçekleştirilmesi.
- Şifrelerin güvenli bir şekilde saklanması (SHA-256 hash kullanılarak).

4.Dinamik Veritabanı Yönetimi:

- MSSQL tabanlı veritabanında tarımsal süreçlerin detaylı şekilde kaydedilmesi ve yönetilmesi.
- Farklı tablolar (Çiftlikler, Ürün Bilgileri, Hasatlar, Satışlar ve Lojistik) arasında ilişki kurularak verilerin bütünlüğü sağlanır.

5.Veri Analizi ve Raporlama:

- Kullanıcıların tarımsal veriler üzerinde sorgular çalıştırarak analiz yapabilmesini sağlamak.
- Dinamik SQL sorguları ile kullanıcıların özel ihtiyaçlarına göre verileri görüntüleyebilmesi.

6.Kolay Entegrasyon:

- Farklı tarımsal süreçlerin izlenmesi ve yönetilmesi için ölçeklenebilir bir altyapı sağlanır.
- Sistem, ileride eklenebilecek yeni tablolar ve işlevler için genişletilebilir yapıdadır.
- Yanlış giriş yapan kullanıcılar için geçici hesap kilitleme sistemiyle güvenliği artırır.
- Kullanıcı hatalarına karşı detaylı uyarılar ve hata yönetimi mekanizmaları sunar.

Projenin İşlevsel Hedefleri:

1. Veri Yönetimi:
   - Çiftlik Yönetimi:Çiftliklerin adları, sahipleri gibi bilgilerin kayıt altına alınması.
   - Ürün Bilgisi Takibi:Tarımsal ürünlerin türleri, isimleri gibi bilgilerin takip edilmesi.
   - Hasat Yönetimi:Her çiftliğin hasat miktarları, tarihleri ve ürün bilgileri gibi detayların tutulması.
   - Satış Yönetimi:Satış tarihleri, müşteriler, miktarlar ve toplam tutar gibi bilgilerin yönetimi.
   - Lojistik Yönetimi:Satış sonrası lojistik süreçlerinin (kargo numarası, teslimat durumu) takibi.

2. Raporlama ve Veri Analizi:Kullanıcılar MSSQL sorguları yazarak çiftlikler, hasatlar veya satışlarla ilgili istedikleri verilere ulaşabilir.Örnek sorgular aracılığıyla kullanıcılar sisteme hızlı bir başlangıç yapabilir.

3. Kullanıcı Yönetimi:
   - Güvenli bir şekilde kullanıcı kaydı oluşturma ve giriş yapma.
   - Hatalı girişlerin ardından kullanıcı hesabını geçici olarak kilitleme mekanizması.
   - Şifrelerin güvenli bir şekilde hashlenerek saklanması.

Bu proje, modern bir tasarımla kullanıcıların rahatça çalışabileceği bir arayüz sunarak tarımsal süreçlerin yönetimini kolaylaştırmayı amaçlamaktadır. Her tablo için farklı renk temalarıyla görsel hiyerarşi oluşturulmuş ve sistem, kullanıcıların tüm operasyonları kolayca takip edebilmesine olanak tanır. Dijital ortamda süreçlerin yönetilmesi, manuel hataları ve zaman kaybını önlerken, verilerin doğru ve hızlı bir şekilde analiz edilmesi karar alma süreçlerini hızlandırır. Tüm bilgiler güvenli bir şekilde veritabanında saklanmakta, şifreleme ve kullanıcı doğrulama mekanizmalarıyla veri güvenliği sağlanmaktadır. Sistem, farklı tarım işletmelerinin ihtiyaçlarına göre özelleştirilebilir ve yeni özelliklerin kolayca eklenmesine uygun bir yapı sunar. Çiftlikten lojistik sürecine kadar tüm operasyonların tek bir platform üzerinden

yönetilebildiği bu proje, küçük ve orta ölçekli tarım işletmeleri, tarımsal kooperatifler ve tarımsal üretimi optimize etmek isteyen bireyler için ideal bir yazılım çözümüdür. Böylelikle, tarım sektörüne modern bir yaklaşım kazandırılarak süreçlerin daha düzenli, güvenli ve verimli hale getirilmesine katkı sağlanır.

4. **Tasarladığınız veri tabanı mimarisinde hangi tablo ve ilişkileri kullanıldığınızı açıklayınız. (10 p)**

1.Tablolar ve Sütunlar:

1.1 Ciftlikler (Çiftlikler Tablosu) :Tarımsal faaliyetlerin yapıldığı çiftliklerin temel bilgilerini saklar.

- Ciftlik_No: Çiftliklerin benzersiz kimliği (Primary Key).
- Ciftlik_Adi: Çiftlik adı.
- Ciftlik_Sahibi: Çiftlik sahibinin adı.

1.2 Konumlar (Çiftliklerin Konum Bilgisi) : Çiftliklerin bulunduğu şehir ve ülke bilgilerini saklar.

- Konum_ID: Konumların benzersiz kimliği (Primary Key).
- Konum_Sehir: Çiftliğin bulunduğu şehir.
- Konum_Ulke: Çiftliğin bulunduğu ülke.

1.3 Ciftlik_Konum (Çiftlik ve Konum Arasındaki İlişki) : Her çiftliği bir konumla ilişkilendiren ara tablo.

- Ciftlik_No: Çiftlikler tablosuna ait anahtar (Foreign Key).
- Konum_ID: Konumlar tablosuna ait anahtar (Foreign Key).

1.4 Urun (Ürün Bilgisi) :Çiftliklerde üretilen ürünleri tanımlamak için kullanılır.

- Ciftlik_No: Ürünün üretildiği çiftliği belirtir (Foreign Key).
- Urun_Adi: Ürünün adı

1.5 UrunBilgi (Ürün Türleri): Ürünlerin türü ve özellikleri hakkında bilgi sağlar.

- Urun_ID: Ürünlerin benzersiz kimliği (Primary Key).
- Urun_Adi: Ürünün adı.
- Urun_Turu: Ürünün türü (ör. Tahıl, Meyve, Sebze).

1.6 Fiyat (Ürün Fiyatları) : Her ürünün birim fiyatını saklar.

- Urun_ID: Fiyatı belirlenen ürün (Foreign Key).
- Birim_Fiyat: Ürünün birim fiyatı.

1.7 SaklamaKosullari (Ürün Saklama Koşulları): Her ürünün saklama koşullarını tanımlar.

- Urun_ID: Saklama koşulları tanımlanan ürün (Foreign Key).
- Saklama_Kosullari: Ürünün saklama koşulları.

1.8 Hasat (Ürün Hasat Bilgisi): Ürünlerin hasat bilgilerini kaydetmek için kullanılır.

- Hasat_No: Hasat işlemine ait benzersiz kimlik (Primary Key).
- Ciftlik_No: Ürünün hasat edildiği çiftlik (Foreign Key).
- Ciftlik_Adi: Çiftlik adı.
- Urun_ID: Hasat edilen ürünün kimliği (Foreign Key).
- Urun_Adi: Hasat edilen ürünün adı.

- Hasat_Tarihi: Hasat tarihi.
- Hasat_Miktari: Hasat edilen miktar.

1.9 Satislar (Satış Bilgisi): Satılan ürünlere ait detayları kaydetmek için kullanılır.

- Satis_No: Satışa ait benzersiz kimlik (Primary Key).
- Urun_ID: Satılan ürün (Foreign Key).
- Musteri_ID: Ürünü satın alan müşteri (Foreign Key).
- Satis_Tarihi: Satış tarihi.
- Satis_Miktari: Satılan miktar.
- Toplam_Tutar: Toplam satış tutarı.

1.10 Musteriler (Müşteri Bilgisi): Ürün satın alan müşterilerin bilgilerini saklar.

- Musteri_ID: Müşteriye ait benzersiz kimlik (Primary Key).
- Musteri_AdiSoyad: Müşterinin adı ve soyadı.
- Musteri_TelNo: Müşterinin telefon numarası.

1.11ojistik (Teslimat ve Kargo Bilgisi) : Satılan ürünlerin teslimat detaylarını saklar.

- Lojistik_ID: Teslimata ait benzersiz kimlik (Primary Key).
- Satis_No: Teslimatı yapılan satış (Foreign Key).
- Kargo_No: Kargo numarası.
- Teslimat_Tarihi: Teslimat tarihi.
- Teslimat_Durumu: Teslimat durumu.
- Tasima_ID: Teslimatı yapan taşıma şirketi (Foreign Key).

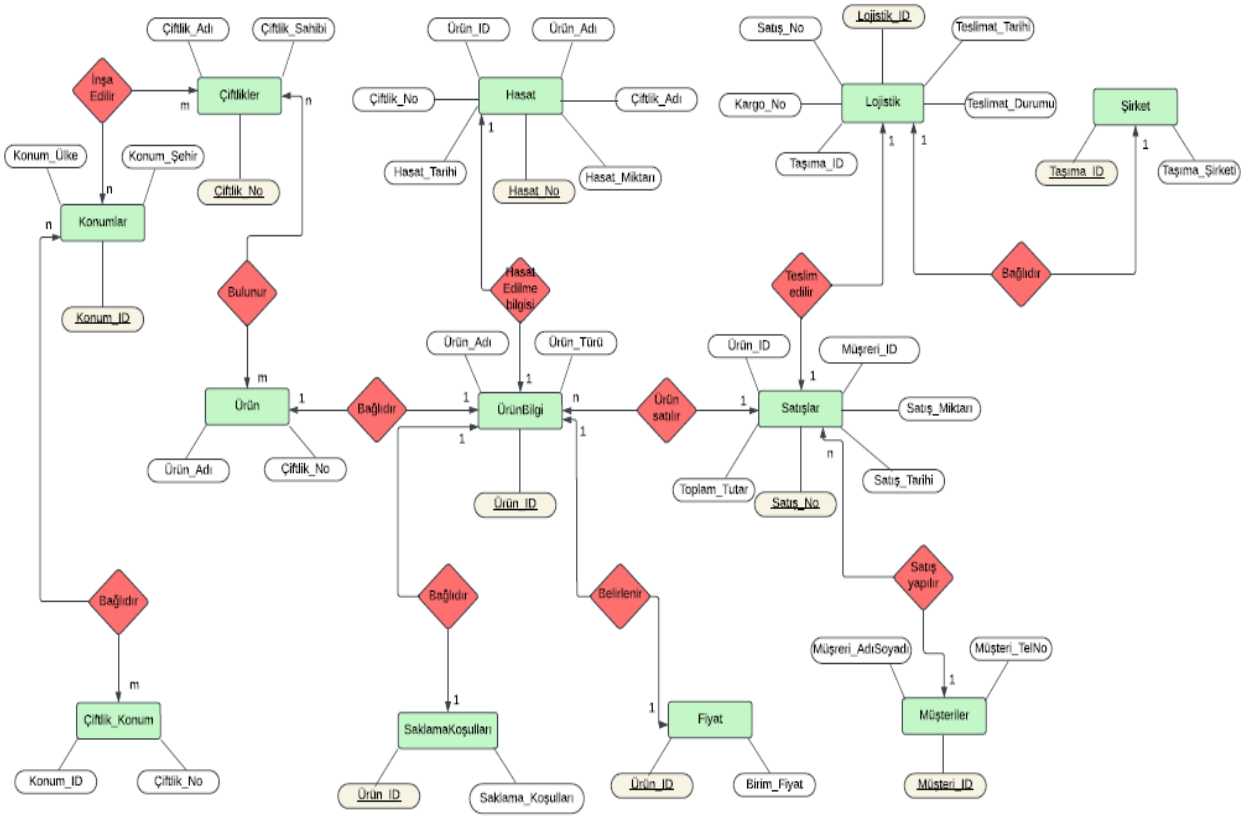
1.12 Sirket (Taşıma Şirketleri): Teslimat süreçlerinde kullanılan taşıma şirketlerinin bilgilerini saklar.

- Tasima_ID: Şirkete ait benzersiz kimlik (Primary Key).
- Tasima_Sirketi: Taşıma şirketinin adı.

2. Tablo İlişkileri

- Ciftlikler ve Konumlar:
  - Ciftlik_Konum tablosu aracılığıyla her çiftlik, bir konuma bağlanmıştır.
- Urun ve Ciftlikler:
  - Her çiftlikte birden fazla ürün üretilebilir (Ciftlik_No ile ilişki).
- Hasat ve Ciftlikler:
  - Her hasat, bir çiftlikle ilişkilidir (Ciftlik_No ile ilişki).
- Hasat ve UrunBilgi:
  - Her hasat edilen ürün, UrunBilgi tablosundaki bir ürünle ilişkilidir.
- Satislar ve Musteriler:
  - Her satış, bir müşteriyle ilişkilidir (Musteri_ID ile ilişki).
- Lojistik ve Satislar:
  - Her lojistik kaydı, bir satış kaydıyla ilişkilidir (Satis_No ile ilişki).
- Lojistik ve Sirket:
  - Her lojistik kaydı, bir taşıma şirketiyle ilişkilidir (Tasima_ID ile ilişki).

5. **Veri tabanı ER (Entity Relationship) diagramının bilgisayar ortamında çizilmiş halini paylaşınız. (Ara raporda eksik kısımlar bu raporda giderilmelidir ve ER çizme programlarından faydalanıbilir. Elle çizim, çizip fotoğrafını çekme vb. kabul edilmeyecektir.) (10 p)**



6. **Herhangi iki tablonuz için DDL (create) kodları yazılmalıdır. (10 p)**

- **Çiftlikler Tablosu (Ciftlikler)**

```
CREATE TABLE Ciftlikler (
   Ciftlik_No INT PRIMARY KEY IDENTITY(1,1),
   Ciftlik_Adi NVARCHAR(100) NOT NULL,
   Ciftlik_Sahibi NVARCHAR(100),
   Konum_Sehir NVARCHAR(100),
   Konum_Ulke NVARCHAR(100)
);
```

- **Hasat Tablosu (Hasat)**

```
CREATE TABLE Hasat (
   Hasat_No INT PRIMARY KEY IDENTITY(1,1),
   Ciftlik_No INT NOT NULL,
   Ciftlik_Adi NVARCHAR(100) NOT NULL,
   Urun_ID INT NOT NULL,
   Urun_Adi NVARCHAR(100) NOT NULL,
   Hasat_Tarihi DATE,
   Hasat_Miktari DECIMAL(10,2),
   FOREIGN KEY (Ciftlik_No) REFERENCES Ciftlikler(Ciftlik_No) ON DELETE
CASCADE,
```

```
    FOREIGN KEY (Urun_ID) REFERENCES UrunBilgi(Urun_ID) ON DELETE CASCADE
);
```

7. 5 adet DML (update, insert, delete) içeren kodları yazılmalıdır. (10 p)

```
INSERT INTO Ciftlikler (Ciftlik_Adi, Ciftlik_Sahibi, Konum_Sehir, Konum_Ulke)
VALUES ('Gökkuşağı Çiftliği', 'Hakan Demir', 'Mersin', 'Türkiye');

INSERT INTO Konumlar (Konum_Sehir, Konum_Ulke)
VALUES ('Mersin', 'Türkiye');

INSERT INTO Ciftlik_Konum (Ciftlik_No, Konum_ID)
VALUES (8, 6); -- Burada 8 ve 6, ilgili çiftlik ve konum ID'leridir.

INSERT INTO Hasat (Ciftlik_No, Ciftlik_Adi, Urun_ID, Urun_Adi, Hasat_Tarihi,
Hasat_Miktari)
VALUES (3, 'Verimli Topraklar', 2, 'Arpa', '2024-12-15', 1000.00);

UPDATE Fiyat
SET Birim_Fiyat = Birim_Fiyat * 1.10
WHERE Urun_ID IN (1, 2, 3);

UPDATE Lojistik
SET Teslimat_Durumu = 'Teslim Edildi', Teslimat_Tarihi = '2024-12-20'
WHERE Kargo_No = 'KARGO005';

DELETE FROM Satislar
WHERE Musteri_ID = 5;

DELETE FROM Musteriler
WHERE Musteri_ID = 5;
```

8. **Projenize ait kendi belirlediğiniz 10 adet SQL sorgusu yazınız, sorguların amacını ve sonuç çıktısını da lütfen ekleyiniz. (Açıklama: Sorgular Select deyimleri ve gruplama fonksiyonlarını HAVING deyimini (min, max, avg, count gibi) ve join deyimlerini (en az iki tablo ile birleştirme sorgusu) içerecek şekilde basitten karmaşığa doğru gitmelidir. Proje sunum anında veri tabanınıza ait sorular SQL ortamında gösterilecek ve açıklanacaktır. Raporunuzda ise sorgular, sorguların cevap ve sonuçlarının ekran görüntüsü olarak paylaşılması beklenmektedir. (10 p) Örnek:**

1)
SELECT Urun_Adi, SUM(Hasat_Miktari) AS Toplam_Hasat
FROM Hasat
GROUP BY Urun_Adi
ORDER BY Toplam_Hasat DESC;

- Bu SQL sorgusunun amacı, **"Hasat"** tablosunda yer alan ürünlerin toplam hasat miktarlarını hesaplamak ve sonuçları en yüksek hasat miktarından en düşük hasat miktarına doğru sıralamaktır.

2)

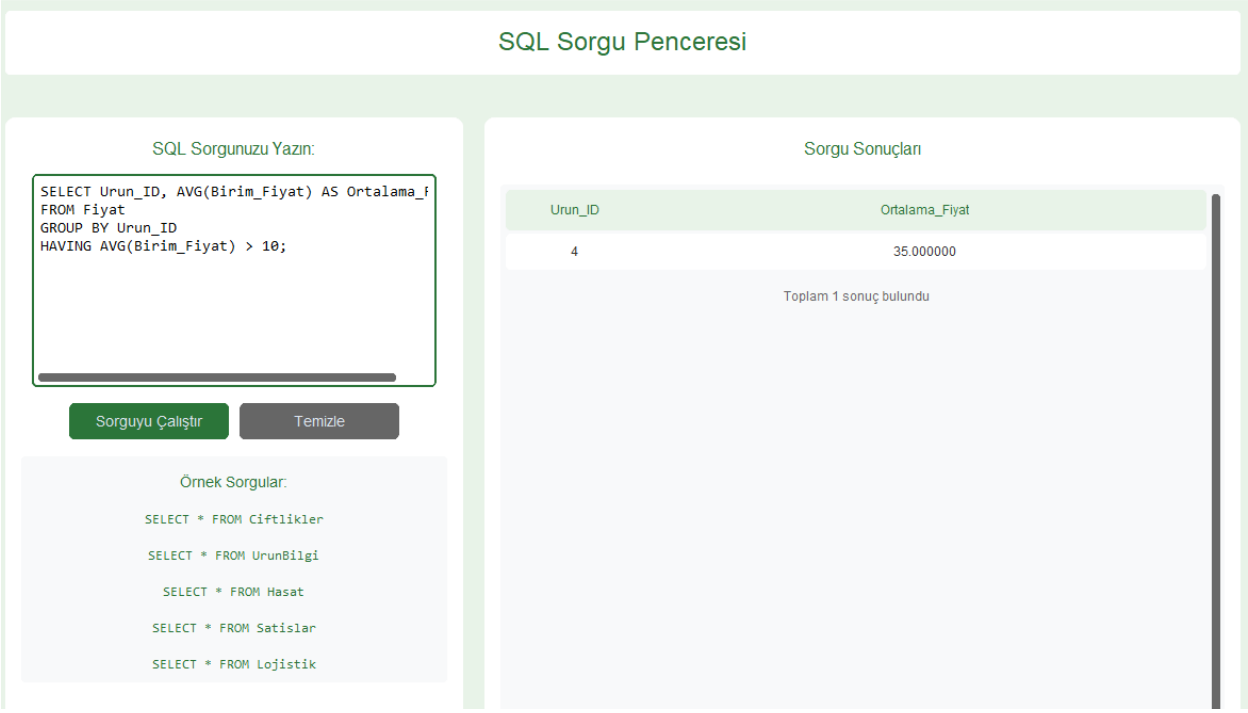
SELECT Urun_ID, AVG(Birim_Fiyat) AS Ortalama_Fiyat

FROM Fiyat

GROUP BY Urun_ID

HAVING AVG(Birim_Fiyat) > 10;

- Bu SQL sorgusunun amacı, **Fiyat** tablosundaki ürünlerin ortalama birim fiyatlarını hesaplamak ve yalnızca ortalama birim fiyatı 10'dan büyük olan ürünleri listelemektir.



3)

SELECT Ciftlik_Adi, Urun_Adi, Hasat_Miktari

FROM Hasat

WHERE Ciftlik_Adi = 'Yeşil Vadi Çiftliği';

- Bu SQL sorgusunun amacı, Hasat tablosundaki verilerden yalnızca "Yeşil Vadi Çiftliği" adlı çiftlikte yetiştirilen ürünlerin adlarını ve hasat miktarlarını listelemektir.

4)

SELECT Ciftlik_No, COUNT(DISTINCT Urun_ID) AS Urun_Sayisi

FROM Hasat

GROUP BY Ciftlik_No;

- Bu SQL sorgusunun amacı, **Hasat** tablosundaki verileri kullanarak her çiftlikte yetiştirilen farklı ürünlerin sayısını hesaplamaktır.

5)

SELECT C.Ciftlik_Sahibi, SUM(H.Hasat_Miktari) AS Toplam_Hasat

FROM Ciftlikler C

JOIN Hasat H ON C.Ciftlik_No = H.Ciftlik_No

GROUP BY C.Ciftlik_Sahibi;

- Bu SQL sorgusunun amacı, Hasat tablosundaki verileri kullanarak her çiftlikte yetiştirilen farklı ürünlerin sayısını hesaplamaktır.

6)

SELECT Urun_ID, MAX(Satis_Miktari) AS En_Yuksek_Satis

FROM Satislar

GROUP BY Urun_ID;

- Bu SQL sorgusunun amacı, Satislar tablosundaki her ürün için yapılan satışlar arasından en yüksek satış miktarını belirlemektir. Sorgu, her ürünün (belirli bir Urun_ID) satış miktarlarını analiz ederek en yüksek değeri döndürür.

7)

SELECT L.Kargo_No, S.Tasima_Sirketi, L.Teslimat_Tarihi

FROM Lojistik L

JOIN Sirket S ON L.Tasima_ID = S.Tasima_ID

WHERE L.Teslimat_Durumu = 'Teslim Edildi';

- Bu SQL sorgusunun amacı, Lojistik ve Sirket tablolarını birleştirerek yalnızca "Teslim Edildi" durumunda olan kargoların detaylarını döndürmektir. Her teslim edilmiş kargo için, kargo numarası, taşıma şirketi ve teslimat tarihi bilgilerini sağlar.

8)

SELECT H.Ciftlik_Adi, S.Urun_ID, MAX(S.Satis_Miktari) AS En_Fazla_Satis

FROM Hasat H

JOIN Satislar S ON H.Urun_ID = S.Urun_ID

GROUP BY H.Ciftlik_Adi, S.Urun_ID;

- Bu SQL sorgusunun amacı, her çiftlikteki ürünler için yapılan en yüksek satış miktarını belirlemektir. Sorgu, Hasat ve Satislar tablolarını birleştirerek, her çiftlik ve ürün bazında en fazla yapılan satış miktarını döndürür.

9)

SELECT S.Tasima_Sirketi, COUNT(L.Kargo_No) AS Toplam_Kargo

FROM Lojistik L

JOIN Sirket S ON L.Tasima_ID = S.Tasima_ID

WHERE L.Teslimat_Durumu = 'Teslim Edildi'

GROUP BY S.Tasima_Sirketi;

- Bu SQL sorgusunun amacı, Lojistik ve Sirket tablolarını birleştirerek, teslim edilmiş kargoların taşıma şirketlerine göre toplam sayısını hesaplamaktır. Her taşıma şirketi için, teslim edilen kargoların toplam sayısını döndürür.

10)

SELECT

C.Ciftlik_Adi,

H.Urun_Adi,

H.Hasat_Miktari,

L.Kargo_No,

S.Tasima_Sirketi,

```
    L.Teslimat_Tarihi
FROM Ciftlikler C
JOIN Hasat H ON C.Ciftlik_No = H.Ciftlik_No
JOIN Lojistik L ON H.Hasat_No = L.Satis_No
JOIN Sirket S ON L.Tasima_ID = S.Tasima_ID;
```

- Bu SQL sorgusunun amacı, Ciftlikler, Hasat, Lojistik, ve Sirket tablolarını birleştirerek bir çiftlikte üretilen ürünlerin hasat, lojistik ve teslimat detaylarını listelemektir. Sorgu, her çiftlikteki ürünlerin hasat miktarını, taşındığı kargo numarasını, taşıma şirketini ve teslimat tarihini bir arada sunar.

9. **Eğer gerçekleştirmiş iseniz, veri tabanı bağlama ve uygulama geliştirme aşamalarınızı kısaca açıklayarak, kullanıcı ara yüz ekranından bir örnek veriniz. Ve geliştirdiğiniz ara yüzü anlatınız. (10 p)**

1)Veri Tabanı Bağlama Aşamaları

- Kullanılan veritabanı: Microsoft SQL Server.
- Bağlantı için **pyodbc** kütüphanesi kullanıldı.
- Bağlantı kodu:

```
DB_CONFIG = {
    'driver': 'SQL Server',
    'server': 'DESKTOP-D11L7UA\\SQLEXPRESS',
    'database': 'TARIMTAKIP',
    'trusted_connection': 'yes'  # Windows Authentication
}

def get_db_connection():
    try:
        conn_str = (
            f"DRIVER={{SQL Server}};"
            f"SERVER={DB_CONFIG['server']};"
            f"DATABASE={DB_CONFIG['database']};"
            f"Trusted_Connection=yes;"
        )
        return pyodbc.connect(conn_str)
    except pyodbc.Error as e:
```

```
        print(f"Bağlantı hatası: {str(e)}")
        raise
```

2. Uygulama Geliştirme Aşamaları

a. Kullanıcı Ara Yüzü Tasarımı:

- CustomTkinter kullanılarak modern bir grafik kullanıcı ara yüzü (GUI) geliştirildi.
- Uygulama, veritabanı ile kullanıcı etkileşimini sağlayan formlar ve raporlama ekranları içerir.

b. CRUD İşlemleri (Ekleme, Güncelleme, Silme, Listeleme):

- Kullanıcılar çiftlik, ürün, hasat ve lojistik kayıtlarını ekleyebilir, güncelleyebilir ve silebilir.
- Örneğin, bir ürün kaydı eklemek için bir form ekranı oluşturuldu:

```
def add_record_window(self, table_name):
    dialog = ctk.CTkToplevel(self)
    dialog.title(f"{table_name} Kaydı Ekle")
```

3.Kullanıcı Ara Yüz Ekranı Örneği

Ana Menü:

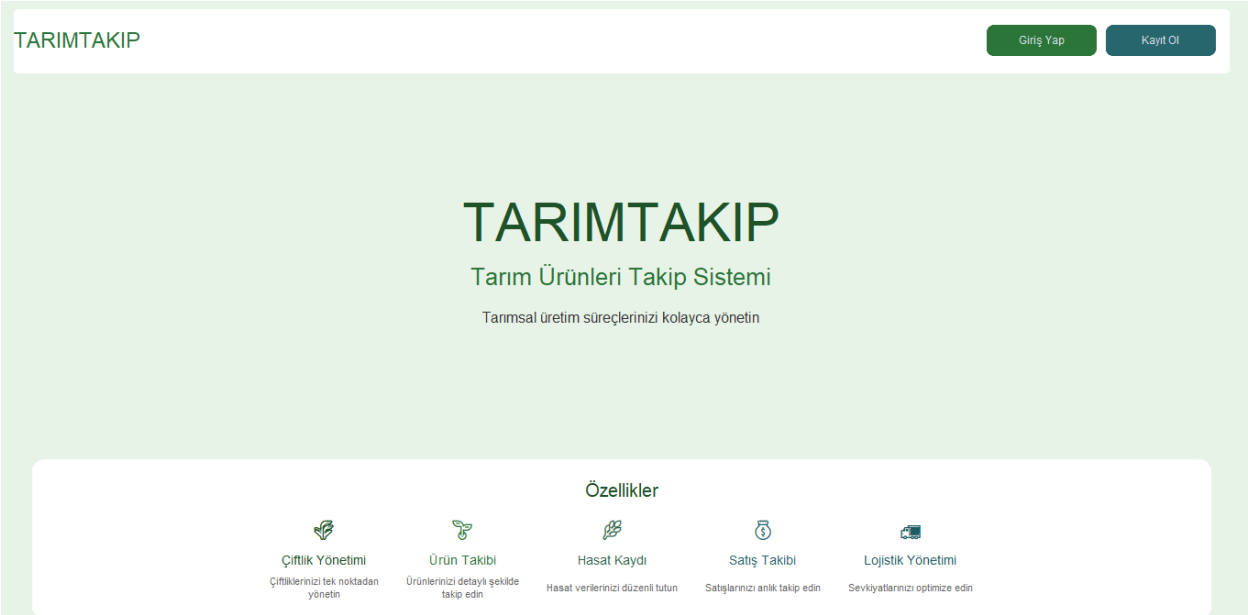
- Ana ekranda kullanıcıların yapabileceği işlemleri içeren bir menü bulunmaktadır.
- Örneğin: Çiftlikler, Ürünler, Hasat ve Lojistik tabloları için düğmeler ve MSSQL sorgusu çalıştırma seçeneği.

```
class MainApp(ctk.CTk):
    def __init__(self):
        super().__init__()
        self.title("Tarım Takip Sistemi")
        self.geometry("600x400")
        self.show_main_menu()

    def show_main_menu(self):
        self.clear_window()
        self.label = ctk.CTkLabel(self, text="İşlem yapmak istediğiniz bölümü seçin",
font=("Arial", 24, "bold"))
        self.label.pack(pady=20)

        self.buttons_frame = ctk.CTkFrame(self)
        self.buttons_frame.pack(fill="both", expand=True, padx=20, pady=20)

        for table in ["Ciftlikler", "Hasat", "Lojistik", "Ürünler"]:
            button = ctk.CTkButton(self.buttons_frame, text=table, command=lambda t=table:
self.open_table_window(t))
            button.pack(pady=10, padx=20, fill="x")
```

4.Ara Yüz Görünümü:

1. Ana Menü Ekranı:
    - İşlem seçimi için düğmeler (ör. Çiftlikler, Hasat, Lojistik).
    - SQL sorgusu çalıştırma düğmesi.

2. CRUD Ekranı:
    - Kayıt ekleme ve güncelleme için giriş alanları.
    - Tablo verilerini listeleme ve silme işlemleri.

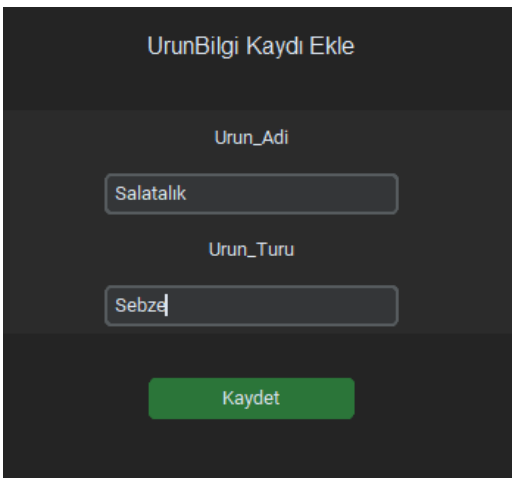


(TarımTakip sisteminin giriş ekranıdır. Kullanıcılar burada sisteme giriş yapabilir, kayıt olabilir ve sistemin özelliklerini görebilir.)

(Giriş Yap ve Kaydol sayfaları)

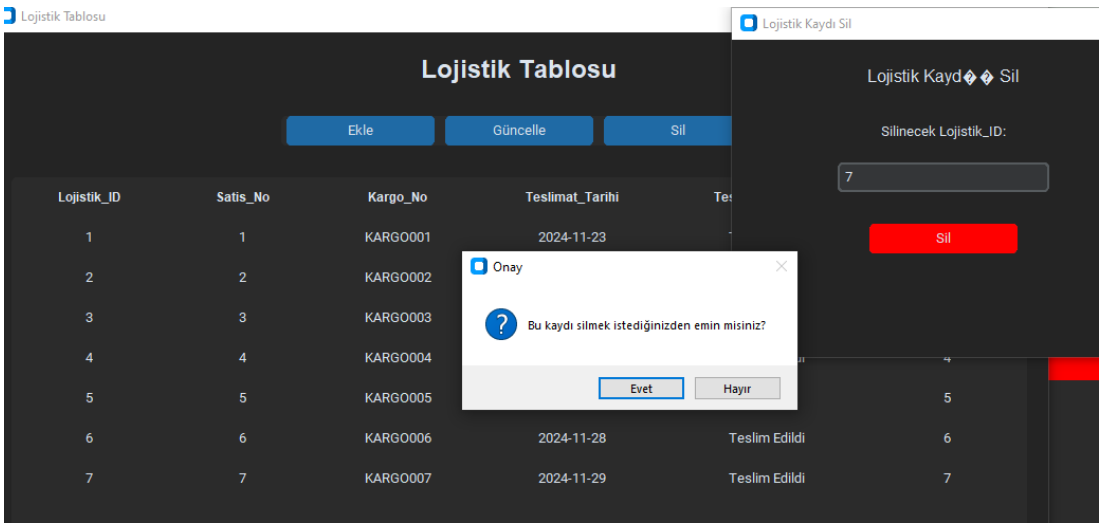
(Kullanıcıların Çiftlikler, Ürün Bilgisi, Hasat, Satışlar, Lojistik ve MSSQL sorgu girişi gibi modüller arasında işlem yapmak için seçim yapabileceği, ayrıca uygulamadan çıkış yapabileceği ana menüdür.)

(Tablolardan birinin örneğidir. Ekle, Güncelle , Sil butonlarını ve tablo içeriklerini içerir.)

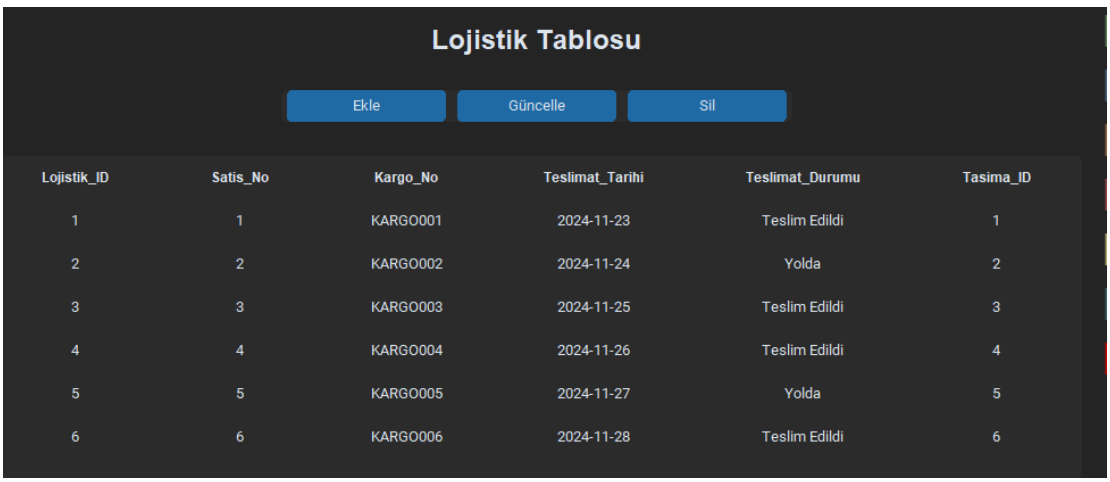


(Ürün ekleme sayfası)

(Eklenilen ürünler tabloyı günceller ve görünür)



(Silme işlemlerinin gerçekleşmesi)



(Silme işlemi sonrası tablonın güncellenmiş çıktısı)

(Sql Sorgularını manuel ya da halihazırdakilere tıklayarak , veritabanından bilgileri çeken ve ekrana yazdıran sayfa)

10. **Eğer veri tabanı bağlama işlemini gerçekleştirmemiş iseniz VTYS sistemlerinde Transaction nedir açıklayınız ve çalışmanızdan bir Transaction örneği veriniz. (10 p)**

11. **View nedir açıklayınız ve bir adet view, bir adet saklı yordam (Stored Procedute) ifadesine ait SQL deyimlerinin sorgusunu ve cevabını yazınız. (10 p)**

VİEW NEDİR ?

View (Görünüm), SQL'de bir veya birden fazla tablodan alınan verilerin belirli koşullarla filtrelenerek oluşturulan sanallaştırılmış bir tablodur. Veritabanında fiziksel olarak veri saklamaz; yalnızca bir sorgunun sonucunu döndüren bir nesnedir. View, karmaşık sorguları basitleştirmek ve tekrar kullanılabilir bir yapı oluşturmak için kullanılır. Dinamik yapısı sayesinde, altında yatan tablolar değiştiğinde view içeriği de otomatik olarak güncellenir. View'ler, kullanıcıların yalnızca belirli verilere erişmesini sağlayarak veri güvenliğini artırır ve hassas bilgilere doğrudan erişimi kısıtlamak için güçlü bir araçtır.

View'ler, genellikle veri güvenliği sağlamak, raporlamayı kolaylaştırmak ve karmaşık sorguları tekrar kullanıma uygun hale getirmek için tercih edilir. Örneğin, birden fazla tabloyu birleştiren ve filtreleme yapan bir sorgu, bir view olarak tanımlanabilir. Bu sayede, kullanıcılar sadece view'i çağırarak kolayca raporlama yapabilir. Ayrıca, karmaşık SQL sorgularını view içine gömerek, uygulama kodunu sadeleştirebilir ve bakım maliyetlerini düşürebilirsiniz. Kullanıcılar view'leri bir tablo gibi kullanarak sorgulamalarını gerçekleştirebilir, bu da kullanım kolaylığı sağlar.

- **View sorgusu örneği;**

```
CREATE VIEW CiftlikUrunHasat AS
SELECT
   C.Ciftlik_Adi AS Ciftlik_Adi,
   C.Ciftlik_Sahibi AS Sahibi,
   UB.Urun_Adi AS Urun_Adi,
   SUM(H.Hasat_Miktari) AS Toplam_Hasat
FROM Ciftlikler C
JOIN Hasat H ON C.Ciftlik_No = H.Ciftlik_No
JOIN UrunBilgi UB ON H.Urun_ID = UB.Urun_ID
GROUP BY C.Ciftlik_Adi, C.Ciftlik_Sahibi, UB.Urun_Adi;

SELECT * FROM CiftlikUrunHasat;
```

- **Saklı Yordam sorgusu örneği;**

```
CREATE PROCEDURE GetSatisDetayByUrun
   @UrunAdi NVARCHAR(100)  -- Parametre: Ürün adı
AS
BEGIN
   SELECT
      S.Satis_No,            -- Satış numarası
      M.Musteri_AdiSoyad AS Musteri, -- Müşteri adı
      M.Musteri_TelNo AS Telefon,    -- Müşteri telefon numarası
      S.Satis_Miktari AS Satis_Miktari, -- Satış miktarı
      S.Toplam_Tutar AS Toplam_Tutar,   -- Toplam tutar
      UB.Urun_Adi AS Urun            -- Ürün adı
   FROM Satislar S
   JOIN Musteriler M ON S.Musteri_ID = M.Musteri_ID
   JOIN UrunBilgi UB ON S.Urun_ID = UB.Urun_ID
   WHERE UB.Urun_Adi = @UrunAdi;  -- Belirtilen ürün adı için filtreleme
END;
```