SQL Dijital Oyun Platformlarında Dinamik Listeleme ve Keşif Sistemi

1. Giriş

Bu proje, bir oyun platformu için tasarlanıp gerçekleştirilen bir SQL Veritabanı Yönetim Sistemidir. Projede, oyun satın alma, topluluklara üyelik, kullanıcı başarıları ve seviyeleri gibi temel sistem fonksiyonlarının yönetimi hedeflenmiştir. Veritabanı tasarımında ilişkisel model kullanılmış, SQL Server'da tablolar ve kısıtlamalar tanımlanmış, örnek veri eklenmiş ve saklı yordamlar (Stored Procedures) ile tetikleyiciler (Triggers) gerçekleştirilmiştir. Projenin E-R diyagramı, ilişkisel veri modeli ve SQL kodları en aşağıda belirtilmiştir.

Amaçlar:

- Kullanıcıların oyun satın alma ve topluluklara üyelik süreçlerini kaydetmek
- Kullanıcı seviyelerini (level) ve başarı kazanımlarını yönetmek.
- Tüm bu süreçlerin veri bütünlüğünü sağlamak.

2. Veritabanı Tasarımı

ER Diyagramı

Yukarıda verilen ER diyagramı, projenin veritabanı tasarımını göstermektedir. Bu diyagram üzerinde temel tablolar ve ilişkileri açıklanır.

Tabloların Açıklamaları:

1. Kullanıcılar: Sisteme kayıtlı kullanıcıların bilgilerini saklar. İçerdiği alanlar: Kullanıcı Adı, Şifre, E-posta, Bakiye, Kayıt Tarihi ve KullaniciLevel.
2. Oyunlar: Platformdaki oyunların temel bilgilerini saklar. İçerdiği alanlar: Oyun Adı, Açıklama, Fiyat, Geliştirici.
3. Kategoriler ve Kategori Atamaları: Oyunların türlerine göre kategorilendirilmesini sağlar.
4. Ödemeler: Kullanıcıların oyun satın alma işlemlerini kaydeder.
5. Satın Almalar: Kullanıcıların sahip olduğu oyunları saklar.
6. Topluluklar ve Üyelikler: Kullanıcıların topluluklara üyeliğini kaydeder.
7. Başarılar ve Kazanılan Başarılar: Kullanıcı başarılarını ve bu başarılardan elde edilen deneyimleri takip eder.
8. Bağlantı Kurduğu: Kullanıcılar arasında bağlantıları (arkadaşlık gibi) kaydeder.

3. Kod Açıklamaları

3.1. Tabloların Oluşturulması

Projede tanımlı tablolar yukarıdaki ER diyagramına uygun şekilde aşağıdaki kodlarla oluşturulmuştur.

Kullanıcılar Tablosu:

```
CREATE TABLE Kullanıcılar (
   KullanıcıID INT PRIMARY KEY IDENTITY(1,1),
```

```
    KullanıcıAd NVARCHAR(50) NOT NULL,
    KullanıcıŞifre NVARCHAR(50) NOT NULL,
    Eposta NVARCHAR(100) NOT NULL UNIQUE,
    Bakiye DECIMAL(10,2) NOT NULL DEFAULT 0.0,
    KayıtTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    KullaniciLevel INT NOT NULL DEFAULT 1
);
```

4. Test Senaryoları

Test 1: Ahmet'in (KullanıcıID: 1) Bulmaca Oyunu Satın Alması

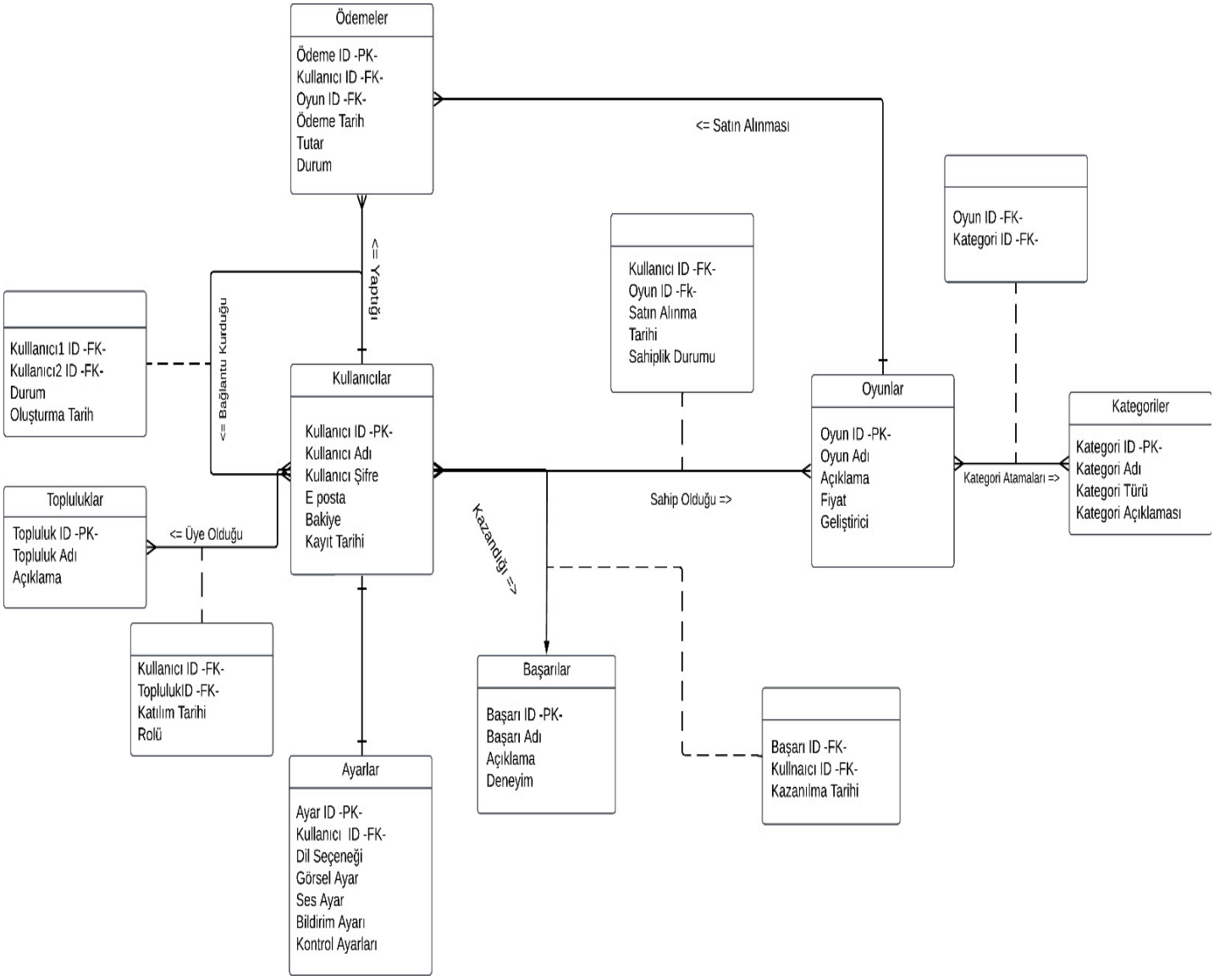
```
EXEC OyunSatınAl @KullanıcıID = 1, @OyunID = 2;
SELECT * FROM Ödemeler WHERE KullanıcıID = 1;
SELECT * FROM SatınAlmalar WHERE KullanıcıID = 1 AND OyunID = 2;
```

Beklenen Sonuç:
- Ödemeler tablosuna kayıt eklenir.
- Kullanıcı bakiyesi güncellenir.
- SahipOldugu tablosuna kayıt eklenir.

5. Sonuç ve Değerlendirme

- Proje hedefleri başarıyla tamamlanmıştır.
- Transaction yönetimi ile veri bütünlüğü sağlanmıştır.
- Tetikleyiciler ve saklı yordamlarla işlemler otomatikleştirilmiştir.

İlişkinin E-R Diyagramı



Projenin İlişkisel Veri Modeli

KULLANICILAR (Kullanıcı ID (PK), Kullanıcı Adı, Kullanıcı Şifre, E-posta, Bakiye, Kayıt Tarihi)

OYUNLAR (Oyun ID (PK), Oyun Adı, Açıklama, Fiyat, Geliştirici)

KATEGORİLER (Kategori ID (PK), Kategori Adı, Kategori Türü, Kategori Açıklaması)

KATEGORİ_ATAMALARI (Oyun ID (FK), Kategori ID (FK))

TOPLULUKLAR (Topluluk ID (PK), Topluluk Adı, Açıklama)

ÜYE_OLDUĞU (Kullanıcı ID (FK), Topluluk ID (FK), Katılım Tarihi, Rolü)

BAŞARILAR (Başarı ID (PK), Başarı Adı, Açıklama, Deneyim)

KAZANDIĞI_BAŞARILAR (Başarı ID (FK), Kullanıcı ID (FK), Kazanılma Tarihi)

ÖDEMELER (Ödeme ID (PK), Kullanıcı ID (FK), Oyun ID (FK), Ödeme Tarihi, Tutar, Durum)

SATIN_ALMALARI (Kullanıcı ID (FK), Oyun ID (FK), Satın Alınma Tarihi, Sahiplik Durumu)

BAĞLANTI_KURDUĞU (Kullanıcı1 ID (FK), Kullanıcı2 ID (FK), Oluşturma Tarihi, Durum)

Projenin SQL Server Kodları

```
-- Tablo: KULLANICILAR
CREATE TABLE Kullanıcılar (
    KullanıcıID INT PRIMARY KEY IDENTITY(1,1),
    KullanıcıAd NVARCHAR(50) NOT NULL,
    KullanıcıŞifre NVARCHAR(50) NOT NULL,
    Eposta NVARCHAR(100) NOT NULL UNIQUE,
    Bakiye DECIMAL(10,2) NOT NULL DEFAULT 0.0,
    KayıtTarihi DATETIME NOT NULL DEFAULT GETDATE(),
        KullaniciLevel INT NOT NULL DEFAULT 1 -- Level sütunu eklendi


);

-- Tablo: OYUNLAR
CREATE TABLE Oyunlar (
    OyunID INT PRIMARY KEY IDENTITY(1,1),
    OyunAd NVARCHAR(100) NOT NULL,
    Açıklama NVARCHAR(255),
    Fiyat DECIMAL(10,2) NOT NULL,
    Geliştirici NVARCHAR(100)
);

-- Tablo: KATEGORİLER
CREATE TABLE Kategoriler (
    KategoriID INT PRIMARY KEY IDENTITY(1,1),
```

```
    KategoriAd NVARCHAR(50) NOT NULL,
    KategoriTür NVARCHAR(50),
    KategoriAçıklama NVARCHAR(255)
);

-- Tablo: KATEGORİ_ATAMALARI
CREATE TABLE KategoriAtamalari (
    OyunID INT NOT NULL,
    KategoriID INT NOT NULL,
    PRIMARY KEY (OyunID, KategoriID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID),
    FOREIGN KEY (KategoriID) REFERENCES Kategoriler(KategoriID)
);

-- Tablo: ÖDEMELER
CREATE TABLE Ödemeler (
    ÖdemeID INT PRIMARY KEY IDENTITY(1,1),
    KullanıcıID INT NOT NULL,
    OyunID INT NOT NULL,
    ÖdemeTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    Tutar DECIMAL(10,2) NOT NULL,
    Durum NVARCHAR(50),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID)
);
```

```
-- Tablo: SATIN_ALMALARI
CREATE TABLE SatınAlmalar (
    KullanıcıID INT NOT NULL,
    OyunID INT NOT NULL,
    SatınAlınmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    SahiplikDurumu NVARCHAR(50),
    PRIMARY KEY (KullanıcıID, OyunID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID)
);

CREATE TABLE Topluluklar (
    ToplulukID INT PRIMARY KEY IDENTITY(1,1),
    ToplulukAd NVARCHAR(100) NOT NULL,
    Açıklama NVARCHAR(255)
);

-- Tablo: ÜYE_OLDUĞU
CREATE TABLE UyeOldugu (
    KullanıcıID INT NOT NULL,
    ToplulukID INT NOT NULL,
    KatılımTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    Rol NVARCHAR(50),
    PRIMARY KEY (KullanıcıID, ToplulukID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (ToplulukID) REFERENCES Topluluklar(ToplulukID)
```

```
);

-- Tablo: BAŞARILAR
CREATE TABLE Başarılar (
    BaşarıID INT PRIMARY KEY IDENTITY(1,1),
    BaşarıAd NVARCHAR(100) NOT NULL,
    Açıklama NVARCHAR(255),
    Deneyim INT NOT NULL
);

-- Tablo: KAZANDIĞI_BAŞARILAR
CREATE TABLE KazandigiBaşarılar (
    KullanıcıID INT NOT NULL,
    BaşarıID INT NOT NULL,
    KazanılmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    PRIMARY KEY (KullanıcıID, BaşarıID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (BaşarıID) REFERENCES Başarılar(BaşarıID)
);

-- Tablo: BAĞLANTI_KURDUĞU
CREATE TABLE BaglantiKuruldugu (
    Kullanıcı1ID INT NOT NULL,
    Kullanıcı2ID INT NOT NULL,
    Durum NVARCHAR(50),
    OlusturmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
```

```
    PRIMARY KEY (Kullanıcı1ID, Kullanıcı2ID),
    FOREIGN KEY (Kullanıcı1ID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (Kullanıcı2ID) REFERENCES Kullanıcılar(KullanıcıID)
);
-- Tablo: SAHİP_OLDUĞU
CREATE TABLE SahipOldugu (
    KullanıcıID INT NOT NULL,
    OyunID INT NOT NULL,
    SatınAlınmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    SahiplikDurumu NVARCHAR(50) NOT NULL,
    PRIMARY KEY (KullanıcıID, OyunID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID)
);

CREATE TABLE Ayarlar (
    AyarID INT PRIMARY KEY IDENTITY(1,1),
    KullanıcıID INT NOT NULL,
    DilSecenegi NVARCHAR(50) DEFAULT 'Türkçe',
    GorselAyar NVARCHAR(50) DEFAULT 'Varsayılan',
    SesAyar NVARCHAR(50) DEFAULT 'Orta',
    BildirimAyar NVARCHAR(50) DEFAULT 'Açık',
    KontrolAyarları NVARCHAR(50) DEFAULT 'Varsayılan',
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID)
);
```

```
-- Tablo: KULLANICILAR
CREATE TABLE Kullanıcılar (
    KullanıcıID INT PRIMARY KEY IDENTITY(1,1),
    KullanıcıAd NVARCHAR(50) NOT NULL,
    KullanıcıŞifre NVARCHAR(50) NOT NULL,
    Eposta NVARCHAR(100) NOT NULL UNIQUE,
    Bakiye DECIMAL(10,2) NOT NULL DEFAULT 0.0,
    KayıtTarihi DATETIME NOT NULL DEFAULT GETDATE(),
        KullaniciLevel INT NOT NULL DEFAULT 1 -- Level sütunu eklendi


);

-- Tablo: OYUNLAR
CREATE TABLE Oyunlar (
    OyunID INT PRIMARY KEY IDENTITY(1,1),
    OyunAd NVARCHAR(100) NOT NULL,
    Açıklama NVARCHAR(255),
    Fiyat DECIMAL(10,2) NOT NULL,
    Geliştirici NVARCHAR(100)
);

-- Tablo: KATEGORİLER
CREATE TABLE Kategoriler (
    KategoriID INT PRIMARY KEY IDENTITY(1,1),
    KategoriAd NVARCHAR(50) NOT NULL,
```

```
    KategoriTür NVARCHAR(50),
    KategoriAçıklama NVARCHAR(255)
);

-- Tablo: KATEGORİ_ATAMALARI
CREATE TABLE KategoriAtamalari (
    OyunID INT NOT NULL,
    KategoriID INT NOT NULL,
    PRIMARY KEY (OyunID, KategoriID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID),
    FOREIGN KEY (KategoriID) REFERENCES Kategoriler(KategoriID)
);

-- Tablo: ÖDEMELER
CREATE TABLE Ödemeler (
    ÖdemeID INT PRIMARY KEY IDENTITY(1,1),
    KullanıcıID INT NOT NULL,
    OyunID INT NOT NULL,
    ÖdemeTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    Tutar DECIMAL(10,2) NOT NULL,
    Durum NVARCHAR(50),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID)
);

-- Tablo: SATIN_ALMALARI
```

```
CREATE TABLE SatınAlmalar (
    KullanıcıID INT NOT NULL,
    OyunID INT NOT NULL,
    SatınAlınmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    SahiplikDurumu NVARCHAR(50),
    PRIMARY KEY (KullanıcıID, OyunID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID)
);

CREATE TABLE Topluluklar (
    ToplulukID INT PRIMARY KEY IDENTITY(1,1),
    ToplulukAd NVARCHAR(100) NOT NULL,
    Açıklama NVARCHAR(255)
);

-- Tablo: ÜYE_OLDUĞU
CREATE TABLE UyeOldugu (
    KullanıcıID INT NOT NULL,
    ToplulukID INT NOT NULL,
    KatılımTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    Rol NVARCHAR(50),
    PRIMARY KEY (KullanıcıID, ToplulukID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (ToplulukID) REFERENCES Topluluklar(ToplulukID)
);
```

```
-- Tablo: BAŞARILAR
CREATE TABLE Başarılar (
    BaşarıID INT PRIMARY KEY IDENTITY(1,1),
    BaşarıAd NVARCHAR(100) NOT NULL,
    Açıklama NVARCHAR(255),
    Deneyim INT NOT NULL
);

-- Tablo: KAZANDIĞI_BAŞARILAR
CREATE TABLE KazandigiBaşarılar (
    KullanıcıID INT NOT NULL,
    BaşarıID INT NOT NULL,
    KazanılmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    PRIMARY KEY (KullanıcıID, BaşarıID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (BaşarıID) REFERENCES Başarılar(BaşarıID)
);

-- Tablo: BAĞLANTI_KURDUĞU
CREATE TABLE BaglantiKuruldugu (
    Kullanıcı1ID INT NOT NULL,
    Kullanıcı2ID INT NOT NULL,
    Durum NVARCHAR(50),
    OlusturmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    PRIMARY KEY (Kullanıcı1ID, Kullanıcı2ID),
```

```
    FOREIGN KEY (Kullanıcı1ID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (Kullanıcı2ID) REFERENCES Kullanıcılar(KullanıcıID)
);
-- Tablo: SAHİP_OLDUĞU
CREATE TABLE SahipOldugu (
    KullanıcıID INT NOT NULL,
    OyunID INT NOT NULL,
    SatınAlınmaTarihi DATETIME NOT NULL DEFAULT GETDATE(),
    SahiplikDurumu NVARCHAR(50) NOT NULL,
    PRIMARY KEY (KullanıcıID, OyunID),
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID),
    FOREIGN KEY (OyunID) REFERENCES Oyunlar(OyunID)
);

CREATE TABLE Ayarlar (
    AyarID INT PRIMARY KEY IDENTITY(1,1),
    KullanıcıID INT NOT NULL,
    DilSecenegi NVARCHAR(50) DEFAULT 'Türkçe',
    GorselAyar NVARCHAR(50) DEFAULT 'Varsayılan',
    SesAyar NVARCHAR(50) DEFAULT 'Orta',
    BildirimAyar NVARCHAR(50) DEFAULT 'Açık',
    KontrolAyarları NVARCHAR(50) DEFAULT 'Varsayılan',
    FOREIGN KEY (KullanıcıID) REFERENCES Kullanıcılar(KullanıcıID)
);
```

```
--Eklemeler

-- KULLANICILAR
INSERT INTO Kullanıcılar (KullanıcıAd, KullanıcıŞifre, Eposta, Bakiye)
VALUES ('Ahmet', '12345', 'ahmet@gmail.com', 150.00),
    ('Ayşe', '54321', 'ayse@gmail.com', 200.00),
    ('Mehmet', '67890', 'mehmet@gmail.com', 100.00);

-- OYUNLAR
INSERT INTO Oyunlar (OyunAd, Açıklama, Fiyat, Geliştirici)
VALUES ('Aksiyon Oyunu', 'Hızlı bir aksiyon oyunu', 50.00, 'OyunStudio'),
    ('Bulmaca Oyunu', 'Zeka geliştiren bir bulmaca oyunu', 20.00, 'PuzzleGames'),
    ('Macera Oyunu', 'Efsanevi bir macera', 70.00, 'AdventureSoft');

-- KATEGORİLER
INSERT INTO Kategoriler (KategoriAd, KategoriTür, KategoriAçıklama)
VALUES ('Aksiyon', 'Tür', 'Adrenalin dolu oyunlar'),
    ('Bulmaca', 'Tür', 'Zeka geliştiren oyunlar'),
    ('Macera', 'Tür', 'Keşfetmeyi sevenler için');

-- KATEGORİ_ATAMALARI
INSERT INTO KategoriAtamalari (OyunID, KategoriID)
VALUES (1, 1), -- Aksiyon Oyunu -> Aksiyon
    (2, 2), -- Bulmaca Oyunu -> Bulmaca
    (3, 3); -- Macera Oyunu -> Macera
```

```
-- ÖDEMELER
INSERT INTO Ödemeler (KullanıcıID, OyunID, Tutar, Durum)
VALUES (1, 1, 50.00, 'Tamamlandı'),
    (2, 2, 20.00, 'Tamamlandı'),
    (3, 3, 70.00, 'Beklemede');

-- SATIN_ALMALARI
INSERT INTO SatınAlmalar (KullanıcıID, OyunID, SahiplikDurumu)
VALUES (1, 1, 'Sahip'),
    (2, 2, 'Sahip');

-- TOPLULUKLAR
INSERT INTO Topluluklar (ToplulukAd, Açıklama)
VALUES ('Geliştiriciler Topluluğu', 'Oyun geliştirme topluluğu'),
    ('Oyuncular Kulübü', 'Tüm oyuncular için');

-- ÜYE_OLDUĞU
INSERT INTO UyeOldugu (KullanıcıID, ToplulukID, Rol)
VALUES (1, 1, 'Üye'),
    (2, 2, 'Moderatör');

-- BAŞARILAR
INSERT INTO Başarılar (BaşarıAd, Açıklama, Deneyim)
VALUES ('İlk Oyun', 'İlk oyununuzu satın aldınız!', 50),
    ('50 Saat Oynama', 'Toplamda 50 saat oyun oynadınız', 100);
```

```
-- KAZANDIĞI_BAŞARILAR
INSERT INTO KazandigiBaşarılar (KullanıcıID, BaşarıID, KazanılmaTarihi)
VALUES (1, 1, GETDATE()),
    (2, 2, GETDATE());

-- BAĞLANTI_KURDUĞU
INSERT INTO BaglantiKuruldugu (Kullanıcı1ID, Kullanıcı2ID, Durum, OlusturmaTarihi)
VALUES (1, 2, 'Arkadaşlık Bekliyor', GETDATE()),
    (2, 3, 'Arkadaşlık Kabul Edildi', GETDATE());


INSERT INTO Kullanıcılar (KullanıcıAd, KullanıcıŞifre, Eposta, Bakiye, KayıtTarihi)
VALUES
('Fatma', 'password123', 'fatma@gmail.com', 300.00, GETDATE()),
('Ali', 'ali123', 'ali@gmail.com', 500.00, GETDATE()),
('Elif', 'elif123', 'elif@gmail.com', 400.00, GETDATE());
INSERT INTO Oyunlar (OyunAd, Açıklama, Fiyat, Geliştirici)
VALUES
('Savaş Oyunu', 'Çok oyunculu savaş oyunu', 60.00, 'BattleCorp'),
('Korku Oyunu', 'Korkutucu bir macera oyunu', 40.00, 'HorrorHouse'),
('Spor Oyunu', 'Futbol ve basketbol simülasyonu', 30.00, 'SportsStudio'),
('Bilimkurgu Oyunu', 'Uzay temalı bilimkurgu oyunu', 70.00, 'SciFiSoft');

INSERT INTO Kategoriler (KategoriAd, KategoriTür, KategoriAçıklama)
VALUES
('Macera', 'Tür', 'Keşif odaklı oyunlar'),
```

('Savaş', 'Tür', 'Çok oyunculu savaş oyunları'),

('Korku', 'Tür', 'Korkutucu hikayeler'),

('Spor', 'Tür', 'Spor simülasyonları');

INSERT INTO KategoriAtamalari (OyunID, KategoriID)

VALUES

(1, 2), -- Savaş Oyunu -> Savaş

(2, 3), -- Korku Oyunu -> Korku

(3, 4), -- Spor Oyunu -> Spor

(4, 1); -- Bilimkurgu Oyunu -> Macera

INSERT INTO Topluluklar (ToplulukAd, Açıklama)

VALUES

('Gamer Kulübü', 'Tüm oyuncular için bir topluluk'),

('Geliştiriciler Grubu', 'Oyun geliştiricilerinin buluştuğu topluluk'),

('Savaş Oyunları Fanları', 'Savaş oyunlarına özel bir topluluk');

INSERT INTO Ödemeler (KullanıcıID, OyunID, Tutar, Durum, ÖdemeTarihi)

VALUES

(1, 1, 60.00, 'Tamamlandı', GETDATE()),

(2, 2, 40.00, 'Tamamlandı', GETDATE()),

(3, 3, 30.00, 'Tamamlandı', GETDATE()),

(4, 4, 70.00, 'Beklemede', GETDATE());

```
INSERT INTO Ayarlar (KullanıcıID, DilSecenegi, GorselAyar, SesAyar, BildirimAyar, KontrolAyarları)
VALUES
(1, 'Türkçe', 'Karanlık', 'Orta', 'Açık', 'Varsayılan'),
(2, 'İngilizce', 'Aydınlık', 'Yüksek', 'Kapalı', 'Özel'),
(3, 'Türkçe', 'Karanlık', 'Düşük', 'Açık', 'Varsayılan'),
(4, 'İngilizce', 'Karanlık', 'Orta', 'Açık', 'Özel');

-- Saklı Yordam

CREATE PROCEDURE OyunSatınAl (
   @KullanıcıID INT,
   @OyunID INT
)
AS
BEGIN
   BEGIN TRANSACTION;
   BEGIN TRY
      -- Ödeme Kaydı Ekleme
      INSERT INTO Ödemeler (KullanıcıID, OyunID, Tutar, Durum)
      SELECT @KullanıcıID, @OyunID, Fiyat, 'Tamamlandı'
      FROM Oyunlar
      WHERE OyunID = @OyunID;

      -- Sahiplik Güncelleme
      INSERT INTO SahipOldugu (KullanıcıID, OyunID, SahiplikDurumu)
      VALUES (@KullanıcıID, @OyunID, 'Sahip');
```

```
        -- Commit İşlemi
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Hata Durumunda Rollback
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;

-- Ahmet'in Bulmaca Oyunu satın alması
EXEC OyunSatınAl @KullanıcıID = 1, @OyunID = 2;

-- Sonuçları kontrol et
SELECT * FROM Ödemeler WHERE KullanıcıID = 1;
SELECT * FROM SahipOldugu WHERE KullanıcıID = 1 AND OyunID = 2;



-- Ayşe'nin Macera Oyunu satın alması
EXEC OyunSatınAl @KullanıcıID = 2, @OyunID = 3;

-- Sonuçları kontrol et
SELECT * FROM Ödemeler WHERE KullanıcıID = 2;
SELECT * FROM SahipOldugu WHERE KullanıcıID = 2 AND OyunID = 3;
```

```
-- Mehmet'in Aksiyon Oyunu satın alması
EXEC OyunSatınAl @KullanıcıID = 3, @OyunID = 1;

-- Sonuçları kontrol et
SELECT * FROM Ödemeler WHERE KullanıcıID = 3;
SELECT * FROM SahipOldugu WHERE KullanıcıID = 3 AND OyunID = 1;



CREATE PROCEDURE BasariEkleVeKullaniciLevelArtir (
    @KullanıcıID INT,
    @BaşarıID INT
)
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        -- Başarı Daha Önce Eklenmiş mi Kontrol Et
        IF NOT EXISTS (
            SELECT 1
            FROM KazandigiBaşarılar
            WHERE KullanıcıID = @KullanıcıID AND BaşarıID = @BaşarıID
        )
```

```
    BEGIN
        -- Başarı Ekleme
        INSERT INTO KazandigiBaşarılar (KullanıcıID, BaşarıID, KazanılmaTarihi)
        VALUES (@KullanıcıID, @BaşarıID, GETDATE());
    END

    -- Kullanıcının Toplam Deneyimini Hesapla
    DECLARE @ToplamDeneyim INT;
    SELECT @ToplamDeneyim = SUM(b.Deneyim)
    FROM Başarılar b
    INNER JOIN KazandigiBaşarılar kb ON b.BaşarıID = kb.BaşarıID
    WHERE kb.KullanıcıID = @KullanıcıID;

    -- Kullanıcının Mevcut KullaniciLevel'ını Al
    DECLARE @MevcutKullaniciLevel INT;
    SELECT @MevcutKullaniciLevel = KullaniciLevel FROM Kullanıcılar WHERE KullanıcıID = @KullanıcıID;

    -- Eğer Toplam Deneyim, Bir Sonraki KullaniciLevel için Gerekli Deneyimden Büyükse KullaniciLevel Artır
    IF @ToplamDeneyim >= (@MevcutKullaniciLevel * 100)
    BEGIN
        UPDATE Kullanıcılar
        SET KullaniciLevel = KullaniciLevel + 1
        WHERE KullanıcıID = @KullanıcıID;
    END;
```

```
        -- Commit İşlemi
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Hata Durumunda Rollback
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;

-- Başarı Kazanımı ve Level Artışı
EXEC BasariEkleVeKullaniciLevelArtir @KullanıcıID = 1, @BaşarıID = 1;

-- Sonuçları Kontrol Et
SELECT KullanıcıID, KullaniciLevel FROM Kullanıcılar WHERE KullanıcıID = 1;

EXEC BasariEkleVeKullaniciLevelArtir @KullanıcıID = 3, @BaşarıID = 2;

-- Sonuçları Kontrol Et
SELECT KullanıcıID, KullaniciLevel FROM Kullanıcılar WHERE KullanıcıID = 3;



-- Başarı Kazanımı (Toplam Deneyim >= Level * 100 Olursa Level Artar)
EXEC BasariEkleVeKullaniciLevelArtir @KullanıcıID = 2, @BaşarıID = 2;
```

```
-- Sonuçları Kontrol Et
SELECT KullanıcıID, KullaniciLevel FROM Kullanıcılar WHERE KullanıcıID = 1;



--Trigger


CREATE TRIGGER SatışSonrasıGüncelleme
ON Ödemeler
AFTER INSERT
AS
BEGIN
  BEGIN TRY
    -- Kullanıcı Bakiyesini Güncelle
    UPDATE Kullanıcılar
    SET Bakiye = Bakiye - i.Tutar
    FROM Kullanıcılar k
    INNER JOIN Inserted i ON k.KullanıcıID = i.KullanıcıID;

    -- Oyunun Sahiplik Tablosuna Eklenmesi
    INSERT INTO SahipOldugu (KullanıcıID, OyunID, SatınAlınmaTarihi, SahiplikDurumu)
    SELECT i.KullanıcıID, i.OyunID, GETDATE(), 'Sahip'
    FROM Inserted i;
  END TRY
  BEGIN CATCH
    -- Hata Durumunda Hata Mesajını Döndür
    PRINT 'Hata oluştu. İşlem durduruldu.';
```

```
    END CATCH
END;

CREATE PROCEDURE SiparisTamamla (
    @KullanıcıID INT,
    @OyunID INT
)
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        -- Oyunun fiyatını kontrol et
        DECLARE @Fiyat DECIMAL(10, 2);
        SELECT @Fiyat = Fiyat FROM Oyunlar WHERE OyunID = @OyunID;

        -- Kullanıcının bakiyesini kontrol et
        DECLARE @Bakiye DECIMAL(10, 2);
        SELECT @Bakiye = Bakiye FROM Kullanıcılar WHERE KullanıcıID = @KullanıcıID;

        IF @Bakiye < @Fiyat
        BEGIN
            THROW 50000, 'Yetersiz bakiye. İşlem iptal edildi.', 1;
        END

        -- Ödeme İşlemini Gerçekleştir
        INSERT INTO Ödemeler (KullanıcıID, OyunID, Tutar, Durum, ÖdemeTarihi)
```

```
        VALUES (@KullanıcıID, @OyunID, @Fiyat, 'Tamamlandı', GETDATE());

        -- İşlemleri Commit Et
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Hata Durumunda Rollback
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;

EXEC SiparisTamamla @KullanıcıID = 1, @OyunID = 2;

-- Test Sonucu Kontrolü
SELECT * FROM Ödemeler WHERE KullanıcıID = 1;
SELECT * FROM SahipOldugu WHERE KullanıcıID = 1 AND OyunID = 2;
SELECT * FROM Kullanıcılar WHERE KullanıcıID = 1;

-- Kullanıcı bakiyesini düşük bir tutara ayarla
UPDATE Kullanıcılar
SET Bakiye = 100.00
WHERE KullanıcıID = 3;

-- Sipariş Tamamlama (Beklenen: Hata ve Rollback)
EXEC SiparisTamamla @KullanıcıID = 2, @OyunID = 2;
```

```
-- Test Sonucu Kontrolü
SELECT * FROM Ödemeler WHERE KullanıcıID = 2;
SELECT * FROM SahipOldugu WHERE KullanıcıID = 1 AND OyunID = 2;
SELECT * FROM Kullanıcılar WHERE KullanıcıID = 1;

 SELECT * FROM Kullanıcılar WHERE KullanıcıID=1

 EXEC SiparisTamamla @KullanıcıID = 3, @OyunID = 1;
```