Ekibinizde yer alan Backend mobil Developer'a teslim etmek üzere SIGN UP sürecine ait analiz dokümanı oluşturunuz. Analiz dokümanında veritabanında tutulması gereken veri başlıklarına, veri tiplerine, foreign key'lere yer vermelisiniz. Oluşturacağınız tabloda yapılacak bir değişikliği nasıl yönetebileceğiniz hususunda alternatif çözüm yollarını Developer ile karşılaştırınız (Update, Flag veya Status Label vs kullanabilirsiniz) (ANALYSIS DOCUMENTATION - TABLE CREATION)

—----------

Değişiklik Yapma Konusunda Çözüm Yolları

1. Update (Güncelleme) Yaklaşımı

- **Açıklama:** Veritabanında mevcut kayıtları doğrudan güncelleyebilirsiniz. Örneğin, bir kullanıcının bilgilerini güncellemek için UPDATE SQL komutu kullanılabilir.
- **Avantajlar:**
    - **Basitlik:** Güncelleme işlemi basittir ve mevcut veritabanı şemasında minimal değişiklik gerektirir.
    - **Hızlı Uygulama:** Veri üzerinde anlık değişiklik gerektiren durumlarda hızla uygulanabilir.
- **Dezavantajlar:**
    - **Veri Kaybı Riski:** Eski veriler doğrudan üzerine yazılır, bu da geri dönüşü zor olan veri kayıplarına neden olabilir.
    - **Performans:** Büyük veri güncellemeleri performans sorunlarına yol açabilir.

2. Flag - Status Label Kullanımı

- **Açıklama:** Flag veya Status Label, veritabanlarındaki bir kayıt üzerinde belirli bir durumun veya özelliğin olup olmadığını belirtmek için kullanılan bir alanı ifade eder.
- **Flag Kullanımının Temel Özellikleri:**
    - **Boolean Flag:** Çoğunlukla true veya false değerlerine sahip bir alan olarak kullanılır. Örneğin, bir kullanıcının aktif olup olmadığını göstermek için bir bayrak.
    - **Enum Flag**: Birden fazla durumu veya durumu temsil eden belirli değerler içeren bir alan. Örneğin, bir iş sürecinin durumlarını izlemek için pending, approved, rejected gibi değerler.
- **Avantajlar:**
    - **Durum Yönetimi:** Kayıtların durumlarını kolayca takip edebilir ve yönetebilirsiniz.
    - **Geçmiş Veri Koruma:** Bayraklar sayesinde verilerin geçmiş durumlarını kaybetmeden yönetebilirsiniz.
- **Dezavantajlar:**
    - **Tablo Büyüklüğü:** Tablodaki kayıtların sayısı hızla artabilir ve yönetimi zorlaşabilir.
    - **Ek Kodlama Gereksinimi**: Uygulama seviyesinde bu statüleri yönetmek için ek kodlama gerektirebilir.
    - **Güvenlik ve Performans:** Büyük veri setlerinde performans sorunları ve güvenlik açıklarına yol açabilir.

FLAG - STATUS LABEL KULLANIM ÖRNEKLERİ

Status Label

```
CREATE TABLE sign_up (
    user_id             INT AUTO_INCREMENT PRIMARY KEY,
    email               VARCHAR(100) UNIQUE NOT NULL,
    password            VARCHAR(255),
    phone               VARCHAR(15) UNIQUE
    birth_date          DATE,
    sign_up_method      VARCHAR(50) NOT NULL,
    social_id           VARCHAR(100),
    created_at          DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at          DATETIME DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
    status ENUM('active','inactive','banned','verified','pending','approved','rejected','deleted') DEFAULT 'active'
  )
```

- **Kullanıcıyı Yasaklı Olarak İşaretleme:**

```
UPDATE sign_up
SET status = 'banned'
WHERE user_id = 1;
```

- **Doğrulanmış E-Posta Adreslerine Sahip Kullanıcıları Listeleme:**

```
SELECT * FROM sign_up
WHERE status = 'verified';
```

Boolean Flag

```
CREATE TABLE sign_up (
    user_id         INT AUTO_INCREMENT PRIMARY KEY,
    email           VARCHAR(100) UNIQUE NOT NULL,
    password        VARCHAR(255),
    phone           VARCHAR(15) UNIQUE
    birth_date      DATE,
    sign_up_method  VARCHAR(50) NOT NULL,
    social_id       VARCHAR(100),
    created_at      DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at      DATETIME DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
    is_2fa_enabled  BOOLEAN DEFAULT FALSE
  )
```

- **İki Faktörlü Kimlik Doğrulamayı Etkinleştirme:**

```
UPDATE sign_up
SET is_2fa_enabled = TRUE
WHERE user_id = 1;
```

- **İki Faktörlü Kimlik Doğrulama Aktif Kullanıcıları Listeleme:**

```
SELECT * FROM sign_up
WHERE is_2fa_enabled = TRUE;
```

3. Soft Delete

- **Açıklama:** Gerçekten veriyi silmek yerine, bir is_deleted flag kullanarak, kullanıcıları sistemde pasif hale getirme yöntemidir.
- **Avantajlar:**
    - Veri kaybı olmadan, kayıtları "silinebilir" hale getirirsiniz. Böylece ileride geri almak mümkün olur.
    - Kayıtların tamamını kaybetmeden temizleme işlemleri yapılabilir.
- **Dezavantajlar:**
    - Silinmiş kayıtların veritabanında kalması yer kaplamasına neden olabilir.
    - Her sorguda is_deleted durumu kontrol edilmelidir, bu da kodlama ve performans açısından yük getirebilir.

4. Audit Logs (Denetim Günlükleri)

- **Açıklama:** Her değişiklik veya güncelleme için ayrı bir denetim kaydı tutmak.
- **Avantajlar:**
    - Değişikliklerin kim tarafından yapıldığını ve ne zaman yapıldığını izleyebilirsiniz.
    - Güvenliği olumlu yönde geliştirir.
- **Dezavantajlar:**
    - Günlükler için ek veri depolama gereksinimleri artar.

5. Versiyonlama

- **Açıklama:** Her kaydın bir versiyon numarası ile takip edilmesini ifade eder.. Her güncelleme yeni bir versiyon oluşturur.
- **Avantajlar:**
    - Kayıtların geçmiş versiyonlarına erişim sağlar.
    - Veri değişikliklerinin ayrıntılı bir şekilde takip edilmesini sağlar.
- **Dezavantajlar:**
    - Veritabanında fazla yer kaplayabilir.

İlişkili Tablolar Durumunda Dikkat Edilmesi Gerekenler

- **Foreign Key Constraint'ler:** Tablolar arası ilişkilerde foreign key'ler kullanılırsa, yapılan değişikliklerin bu anahtarlara nasıl etki edeceği göz önünde bulundurulmalıdır.
- **Cascade Update/Delete:** Cascade özellikleri dikkatle ele alınmalı, çünkü bir tablodaki güncelleme veya silme işlemi diğer ilişkili tabloları da etkileyebilir.
- **Referential Integrity:** Veritabanında referential integrity (ilişki bütünlüğü) korunmalı, yani bir tablodaki değişiklik diğer ilişkili tabloları bozmamalıdır.