1. En Zor Kısım ve Neden?

Veri tabanından gelen iki farklı tabloyu (tbl_daily_campaigns ve tbl_daily_scores) optimum şekilde birleştirmek ve filtreleme işlemlerini verimli hale getirmek en zor kısımdı.

Büyük veri setleriyle çalışırken join işlemlerinin ve filtreleme süreçlerinin performansını optimize etmek gerekti.

2. En Uzun Süren ve En Çok Bellek Tüketen Kısım?

Veri birleştirme ve filtreleme işlemleri en uzun süren ve bellek tüketen kısımdı.

Pandas kullanarak büyük veri kümeleriyle yapılan merge işlemleri, bellek kullanımı açısından dikkat gerektiriyor.

3. Olası Güvenlik Açıkları Nelerdir?

- SQL Injection Riski: Veritabanı sorgularında kullanıcı girişlerine karşı doğrudan SQL ifadeleri kullanılmadı, SQLAlchemy kullanılarak güvenli sorgular oluşturuldu.

- Veritabanı Şifre Güvenliği: Şifreler açıkça kodda yer alıyor. Daha güvenli olması için .env dosyası kullanılarak gizlenmeli.

- DDoS ve Rate Limiting: API'ye çok fazla istek gönderilmesi durumunda performans sorunları olabilir. Rate limiting (örn. FastAPI Throttling) eklenerek bu önlenebilir.

4. AWS Üzerinde Nasıl Tasarlanır?

Eğer bu sistem AWS üzerinde barındırılacak olsaydı:

- EC2 Üzerinde FastAPI Sunucu: Uvicorn ile çalıştırılan FastAPI, bir EC2 instance üzerinde çalıştırılır.

- RDS MySQL Kullanımı: Veritabanı için Amazon RDS MySQL kullanılarak, verilerin yönetimi optimize edilir.

- Load Balancer & Auto Scaling: Yoğun trafiğe karşı AWS ALB (Application Load Balancer) ve Auto Scaling Group eklenebilir.

- S3 & CloudFront (Opsiyonel): Eğer statik dosya kullanımı olursa S3 ve CloudFront kullanılabilir.

5. Daha Fazla Zamanım Olsaydı Neleri Daha İyi Yapardım?

- Unit Testler Eklerdim: API endpoint'lerini test etmek için pytest kullanarak test senaryoları oluşturulabilirdi.

- Veritabanı Optimizasyonu Yapardım: SQL sorgularının daha hızlı çalışması için indexleme ve denormalizasyon teknikleri uygulanabilirdi.