1. En Zor K?s?m ve Neden?

Veritaban?ndan gelen iki farkl? tabloyu (tbl_daily_campaigns ve tbl_daily_scores) optimum ?ekilde birle?tirmek ve filtreleme i?lemlerini verimli hale getirmek en zor k?s?md?.

Büyük veri setleriyle çal???rken join i?lemlerinin ve filtreleme süreçlerinin performans?n? optimize etmek gerekti.

2. En Uzun Süren ve En Çok Bellek Tüketen K?s?m?

Veri birle?tirme ve filtreleme i?lemleri en uzun süren ve bellek tüketen k?s?md?.

Pandas kullanarak büyük veri kümeleriyle yap?lan merge i?lemleri, bellek kullan?m? aç?s?ndan dikkat gerektiriyor.

3. Olas? Güvenlik Aç?klar? Nelerdir?

- SQL Injection Riski: Veritaban? sorgular?nda kullan?c? giri?lerine kar?? do?rudan SQL ifadeleri kullan?lmad?, SQLAlchemy kullan?larak güvenli sorgular olu?turuldu.

- Veritaban? ?ifre Güvenli?i: ?ifreler aç?kça kodda yer al?yor. Daha güvenli olmas? için .env dosyas? kullan?larak gizlenmeli.

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

4. AWS Üzerinde Nas?l Tasarlan?r?

E?er bu sistem AWS üzerinde bar?nd?r?lacak olsayd?:

- EC2 Üzerinde FastAPI Sunucu: Uvicorn ile çal??t?r?lan FastAPI, bir EC2 instance üzerinde çal??t?r?l?r.

- RDS MySQL Kullan?m?: Veritaban? için Amazon RDS MySQL kullan?larak, verilerin yönetimi optimize edilir.

- Load Balancer & Auto Scaling: Yo?un trafi?e kar?? AWS ALB (Application Load Balancer) ve Auto Scaling Group eklenebilir.

- S3 & CloudFront (Opsiyonel): E?er statik dosya kullan?m? olursa S3 ve CloudFront kullan?labilir.

5. Daha Fazla Zaman?m Olsayd? Neleri Daha ?yi Yapard?m?

- Unit Testler Eklerdim: API endpoint?lerini test etmek için pytest kullanarak test senaryolar? olu?turulabilirdi.

- Veritaban? Optimizasyonu Yapard?m: SQL sorgular?n?n daha h?zl? çal??mas? için indexleme ve denormalizasyon teknikleri uygulanabilirdi.