SQL-Scorer

İki SQL sorgusunu çalıştırıp çalışma süresi, plan kalitesi ve okunaklılık açısından puanlayan, sonuçları grafik + tablo şeklinde veren Python / PostgreSQL tabanlı bir araç.

1. Amaç & Kapsam

- Gerçek çalıştırma + EXPLAIN (ANALYZE) ile performans ölçümü
- Plan analiziyle query-optimization puanı
- SQLFluff ile okunaklılık / kod kalitesi denetimi
- Tek komutla .sql dosyalarını seç → grafik rapor ve özet tablo üret

2. Mimari Tasarım

```
flowchart LR
    UI[CLI & Tkinter Dialog] --> Compare
    Compare((compare.py)) --> Runner
    Runner -->|SQLAlchemy| Postgres[(PostgreSQL 16)]
    Runner --> Analyzer
    Runner --> Linter
    Analyzer --> Scorer
    Linter --> Scorer
    Scorer --> Reporter
    Reporter -->|PNG+Rich| UI
```

Katman	Dosya	Sorumluluk
Runner	`src/runner.py`	EXPLAIN + gerçek çalıştırma + ham metrikler
Analyzer	`src/analyzer.py`	Plan JSON'dan `seq_scans`, `total_cost` vb. çıkarır
Linter	`src/linter.py`	SQLFluff ile `lint_score`
Scorer	`src/scorer.py`	Min-max normalize + ağırlıklar (`weights.yaml`)
Reporter	`src/reporter.py`	Rich tablo + matplotlib grafiği
CLI / Dialog	`run_scorer.py` / `compare.py`	Dosya seç, sonuçları göster

3. Teknoloji Seçimleri

Katman	Tercih	Gerekçe
Dil	Python 3.12	Hızlı prototipleme, zengin ekosistem
DB	PostgreSQL 16 (Docker)	EXPLAIN JSON desteği, kolay kurulum
ORM	SQLAlchemy 2.0	DB-agnostic, pool yönetimi
Plan Analizi	Native JSON + Python	Harici eklentiye gerek yok
Kod Kalitesi	SQLFluff	Dialect-aware linting
Görselleştirme	Matplotlib + Rich	Terminal + PNG raporu
Container	Docker Compose	Tek komutla ortam kurulumu

4. Kurulum

```
git clone https://github.com/…/sql-scorer.git
cd sql-scorer
python -m venv venv && venv\Scripts\activate        # Windows
pip install -r requirements.txt
docker compose up -d                                # Postgres servisi
# (isteğe bağlı demo veri)
Get-Content samples/init_demo.sql | docker exec -i scorer-postgres psql -U scorer -d scorertest
```

5. Kullanım

Senaryo	Komut
GUI dialog ile iki sorgu seç	`python run_scorer.py`
Parametreyle	`python run_scorer.py samples/a.sql samples/b.sql`
Sadece karşılaştırma (CLI)	`python compare.py a.sql b.sql --verbose`

Üretilen dosya:

- report.png → Kategori bar grafiği + alt-metrik özet + kazanan

6. Puanlama Metodolojisi

Kategori	Alt Metrikler	Ağırlık (%)
Computational	`elapsed_ms`, `cpu_sec`, `mem_mb`	40
Optimization	`seq_scans`, `index_scans`, `total_cost`	30
Readability	`lint_score`	30

- Min-max normalizasyon (düşük = iyi metriklerde terslenir)
- Toplam puan 0-100 ölçeğine çevrilir
- Katkı dağılımı grafikte üç renkli bar olarak gösterilir

7. Performans Ölçüm Ayrıntısı

elapsed_ms == PerfCounter stopwatch (ms) cpu_sec == psutil.Process().cpu_times().user mem_mb == ΔRSS / 1_048_576 seq_scans == #Plan nodes with Node Type='Seq Scan' total_cost == Σ Total Cost of all plan nodes

8. Containerization

```
# docker-compose.yml
services:
  db:
    image: postgres:16
    environment:
      POSTGRES_USER: scorer
      POSTGRES_PASSWORD: scorer
      POSTGRES_DB: scorertest
    ports: ["5432:5432"]
    volumes:
      - pgdata:/var/lib/postgresql/data
volumes:
  pgdata:
```

9. Sonuç & Gelecek İşler

Başarılar

- Tek komutla iki sorgu karşılaştırma
- Grafik + terminal tablo
- Kategori-bazlı puanlama

Geliştirilecekler

- Otomatik indeks önerisi (pg-index-adviser)
- Web UI (FastAPI + React)
- Farklı veritabanı sürücüleri (MySQL)
- Daha zengin plan metrikleri (disk I/O blokları)