2. Ders - SQL

3 Eylül 2025 Çarşamba 09:16

Dün öğrendiklerimizi tekrar ederek güne başladık.

```python
def filtrele(musteriler, min_yas=None, max_yas=None) :
    sonuc=[] #başlangıçta boş liste olarak tanımladık, sonrasında değerleri buraya tanımlayacağız
    for m in musteriler:
      yas = m.get("age")
      if yas is None:
        continue #yaş verisi bulunmadığında o veriyi atlamayı sağladı. Eksik veriye karşı önlem aldık
      if min_yas is not None and yas < min_yas:
        continue
      if max_yas is not None and yas > max_yas:
        continue
      sonuc.append(m)
    return sonuc
```

Bunu kolay bir şekilde switch case ile de işledik :

```python
def yas_kategorisi(yas: int) -> str: #tam sayı olarak alınan yaş bilgisini bir kategori olarak döndürmeye yarıyor
    match yas: #match yapısı ile yaşların kuralları sağlanıyor
    #işin kuralını case'de veriyoruz, match de bunu eşliyor
      case _ if yas< 18:
        return "Çocuk"
      case _ if  18 <= yas <= 25:
        return "Genç"
      case _ if 26 <= yas <= 35:
        return "Orta Yaşlı"
      case _ if 26<= yas <= 35:
        return "Genç Yetişkin"
      case _ if 36 <= yas <= 45:
        return "Orta Yaş"
      case _ if 46 <= yas <= 60:
        return "Yaşlı"
      case _: #60 üstü bütün yaşlar için
        return "Bilinmeyen"
```

- https://docs.python.org/3.13/library/index.html -> Python kütüphaneleri için bu dökümantasyondan inceme yapmamızı önerdi hoca.

Any Kütüphanesi, Python'un typing modülünde bulunan ve tip ipucu (type hint) sisteminde özel bir anlama sahip olan bir türdür.

- Kısaca, herhangi bir veri türünü temsil etmek için kullanılır.
- Bir değişken, parametre veya dönüş değeri Any olarak belirtildiğinde, bu o öğenin her türden veri kabul edebileceği veya döndürebileceği anlamına gelir.

```python
from typing import Any, Iterable, Mapping, Sequence

def bucket_customers_by_age(
    customers : Iterable[Mapping[str, Any]] #Müşteri listesi (Her müşteri bir sözlük)
    #her veri türü gelebilir. string türünde gelecek, sözlük yapısında olacak ve bunları iterasyonla işleyebilirm
    buckets : Sequence[tupple[int, int]] # =(18,25) şeklinde yaş aralıklarını işler

)
```

Anahtar-Değer İlişkisi örneği :

```python
{"isim" : "Rümeysa"}
```

Bu kod, müşterileri belirtilen yaş aralıklarına (bucket'lara) göre gruplandırır.

****

SQL

- Yazılan her sorgunun bir maliyeti vardır.
- Bazı şirketler sorgu maliyet tablosu çıkarmadan projeye başlamaz.
- Sorgu maliyeti, sorgunun işlenmesi için gereken kaynak (CPU, bellek, disk I/O) miktarıdır ve yürütme planına bakılarak tahmin edilen göreceli bir değerdir.
- Sorgu motorunda bilgisayardaki fiziksel diskler kullanıyor. Bu işlem için kurumsal çalışma ortamında yaptığımız bu sorgular maliyeti etkiliyor.
- (Cloud tarafında maliyet daha fazla artıyormuş.)

- **SELECT :** Veritabanından belirli verileri sorgulamak ve çekmek için kullanılan temel SQL komutudur. "Ne istiyorum?" sorusuna cevap verir.

- **AS :** Sorgu sonuçlarında sütunlara veya tablolara geçici isim (takma ad/alias) vermek için kullanılan SQL yan tümcesidir. Sonucu daha okunabilir yapar.

- **JOIN :** İki veya daha fazla tabloyu ortak bir sütun (alan) üzerinden birleştirerek tek bir sonuç tablosu oluşturan SQL operasyonudur.

- **GROUP BY :** Bir SQL sorgusunda, aynı değerlere sahip satırları gruplamak ve her grup için toplu hesaplamalar (COUNT, SUM, AVG, MAX, MIN vb.) yapmak için kullanılan bir yan tümcedir.

- **HAVING:** SQL'de GROUP BY ile oluşturulmuş gruplara koşul filtrelemesi uygulamak için kullanılan yan tümcedir. (WHERE satırları, HAVING grupları filtreler.)

Projection Push Down, bir veritabanı veya veri işleme sorgusunda yalnızca ihtiyaç duyulan sütunları mümkün olan **en erken aşamada** seçme (projeksiyon) işlemidir.

Amacı: Gereksiz verileri (sütunları) baştan eleyerek bellek kullanımını ve I/O işini azaltıp performansı artırmak.

Semantic Clarity (Anlamsal Netlik): Kodun yalnızca çalışması değil, aynı zamanda ne yaptığının da kolayca anlaşılabilir olması gerektiğini savunan yazılım prensibidir.

Amacı, kodu okuyan birinin niyeti ve mantığı hızla kavrayabilmesidir.

https://www.semanticclarity.com/ -> AS yapısı anlatılırken Caner hoca bu bağlantıyı önerdi.

https://www.learnaiso.com/guides/semantics/

- **JOIN ORDER OPTİMİZASYONU :** Bir SQL sorgusundaki JOIN işlemlerinin sırasını en verimli olacak şekilde (en az veriyi işleyecek şekilde) değiştiren bir sorgu optimizasyon tekniğidir.
  - Önce küçük, sonra da büyük tablo eşleştiriliyor bu işlemde
  - **Amacı**: Ara sonuç kümesinin boyutunu mümkün olduğunca küçük tutarak toplam işlem süresini ve bellek kullanımını azaltmak.

- **Shuffle Maliyeti**, dağıtık bir sistemde (örn. Spark, Hadoop) verilerin farklı makineler (node'lar) arasında yeniden düzenlenmesi ve taşınmasının yol açtığı performans bedelidir.
  - Neden Olur? JOIN, GROUP BY gibi operasyonlarda, aynı anahtara (key) sahip verilerin işlenmek üzere aynı makineye gönderilmesi gerekir. Bu ağ (network) IO ve disk IO yükü yaratır.
  - Etkisi: İşlemi yavaşlatır, ağ bant genişliği tüketir ve sistem kaynaklarını yoğun kullanır. Optimizasyonların amacı shuffle maliyetini en aza indirmektir.

- **Partition Aware Join**, dağıtık sistemlerde, join yapılacak her iki tablonun da aynı anahtara ve aynı partitioning yöntemine göre önceden bölümlenmiş (partitioned) olmasından yararlanan bir optimizasyondur.
  - Avantajı: Veriler zaten aynı kuralla dağıtıldığı için shuffle (verilerin makineler arasında yeniden karıştırılması) gerektirmez. Her makine kendi yerel verisi üzerinde işlem yapabilir, bu da performansı büyük ölçüde artırır.
  - Hocanın önerdiği kaynak -> https://www.guptaakashdeep.com/storage-partition-join-in-apache-spark-why-how-and-where/

- **Explain Plan**, bir SQL sorgusunun nasıl çalıştığını (hangi adımları izleyeceğini) önceden gösteren bir araçtır.
  - Sorguyu çalıştırmaz, sadece veritabanı optimizörünün seçtiği yürütme planını (execution plan) metinsel veya grafiksel olarak raporlar.

- Amacı: Sorgunun performansını analiz etmek, darboğazları (örn. full table scan) bulmak ve optimizasyon yapmaktır.
- Sorgunun hangi index'leri kullanacağını, join stratejilerini (ör. Hash Join, Nested Loop) ve işlem sırasını özetler. Böylece performans sorunları (tam tablo taraması, yüksek maliyetli işlemler) önceden tespit edilerek optimize edilebilir.

Sürükle bırak teknolojisi için örnek verildi :
- https://www.knime.com/
- Algoritmaların hepsi node olarak yazıyor.

- **Roll-up Table**, bir veri ambarında (data warehouse) özet veri depolamak için kullanılan önceden hesaplanmış bir tablodur.
  - Amacı: Detay seviyesindeki büyük veri kümeleri üzerinde karmaşık sorguları ve toplama (aggregation) işlemlerini çok daha hızlı çalıştırmaktır.
  - Nasıl Çalışır? Veriler, sık sorgulanan boyutlar (örneğin zaman, bölge, ürün kategorisi) seviyesinde özetlenir ve bir tabloda saklanır. Böylece her seferinde ham veriyi işlemek yerine, bu önceden hesaplanmış özet tablo doğrudan okunur ve sorgu performansı büyük ölçüde artar.

FİLTRELEME

```sql
SELECT musteri_adi, sehir
FROM musteriler
WHERE yas > 30 AND sehir = 'İstanbul'
LIMIT 100;
```

- İlk 100 satırı sınırlandırarak buradaki istenilen kritere uygun müşterileri seçmektedir.

```sql
SELECT d.region, SUM(m.fuel_type= 'Electric') AS ev_sales,
SUM(m.fuel_type <> 'Electric') AS ice_sales,
COUNT(*) AS total
FROM sales s
JOIN dealers d ON d.dealer_id = s.dealer_id
JOIN models m ON m.model_code = v.model_code
WHERE s.sale_date >= CURRENT_DATE - INTERVAL 60 DAY
GROUP BY d.region
ORDER BY total DESC;
```

SELECT d.region -> Bölge verisi alınıyor. Gruplamanın temel boyutunu ifade eder.
SUM(m.fuel_type= 'Electric') AS ev_sales, -> Elektrikli araç satış verilerinin sayısını alıyoruz.
SUM(m.fuel_type <> 'Electric') AS ice_sales -> Elektrikli olmayan araçların sayılarını alıyoruz.
COUNT(*) AS total -> Gruptaki toplam satır sayısını verir.

FROM sales s -> satış kayıt tablosu alınıyor
JOIN dealers d ON d.dealer_id = s.dealer_id -> satış tablosu bayiler tablosu ile birleştirildi. Her satışın hangi bayide yapıldığını anlamak için birleştiriliyor.
JOIN models m ON m.model_code = v.model_code -> Satışın hangi aracı kapsadığını görmek için araç tablosunu bağlıyoruz. Aracın modelinden yakıt tipine erişmek için modeller tablosuna bağlıyoruz.

Region	Ev_Sales	İce_sales	Total
Marmara	15	35	50
Ege	8	22	30
İç Anadolu	5	10	15

```
DROP TABLE IF EXISTS sales;

CREATE TABLE dealers(
  dealer_id INT PRIMARY KEY,
  dealer_name VARCHAR(10) NOT NULL;
  city VARCHAR(50) NOT NULL,
  region VARCHAR(50) NOT NULL)
  ENGINE=InnoDB;

 CREATE TABLE customers(
   customer_id INT PRIMARY KEY,
   full_name VARCHAR(100) NOT NULL,
   city VARCHAR(50) NOT NULL,
   region VARHCAR(50) NOT NULL,
   age INT NOT NULL,
   customer_type ENUM('B2C', 'B2B') NOT NULL,
   income_band ENUM('LOW', 'AWG', 'HIGH', 'ULTRA') NOT NULL)ENGINE = InnoDB;

CREATE TABLE models(
  model_code VARCHAR(20) PRIMARY KEY,
  series VARCHAR(20) NOT NULL,
  model_name VARCHAR(40) NOT NULL,
  body_type VARCHAR(20) NOT NULL,
  fuel_type ENUM('Petrol', 'Diesel', 'Hybrid', 'Electric') NOT NULL,
  drivetrain ENUM('RWD', 'AWD', 'FWD'),
  segment VARCHAR(10) NOT NULL) ENGINE=InnoDB;

CREATE TABLE sales(
  sale_id INT PRIMARY KEY, --bu tabloya ait olduğu için foreign key vermeye gerek yok
  sale_date DATE NOT NULL,
  dealer_id INT NOT NULL,
  customer_id INT NOT NULL,
  sale_price DECIMAL(12,2) NOT NULL,
  discount_amt DECIMAL(12,2) NOT NULL DEFAULT 0,
  FOREIGN KEY(dealer_id) REFERENCES dealers (dealer_id),
  FOREIGN KEY(customer_id)REFERENCES customers (customer_id) )

INSERT INTO dealers VALUES
(1, 'Borusan BMW Istanbul', 'İstanbul', 'Marmara');

SELECT d.dealer_id, d.dealer_name, d.city,
FROM dealers d
WHERE NOT EXIST (
  SELECT 1
  FROM sales s
  WHERE s.dealer_id = d.dealer_id AND s.sale_date >= '2025-08-01' AND s.sale_date < '2025-09-01');

SELECT d.region, SUM(m.fuel_type= 'Electric') AS ev_sales,
SUM(m.fuel_type <> 'Electric') AS ice_sales,
COUNT(*) AS total
FROM sales s
JOIN dealers d ON d.dealer_id = s.dealer_id
JOIN models m ON m.model_code = v.model_code
WHERE s.sale_date >= CURRENT_DATE - INTERVAL 60 DAY
GROUP BY d.region
ORDER BY total DESC;


--buna ekleme sorgu ödevi verildi, ses kaydının sonlarını dinle onları yazamadım
```