EKLER

JUPYTER KISA YOLLARI

Python için kullanılabilecek çeşitli geliştirme ortamları bulunmakla birlikte özellikle veri bilimi söz konusu olduğunda en çok kullanılan ve son yıllarda da kullanımı giderek artan geliştirme ortamı Jupyter Notebook veya JupyterLab'dır. Bu ikisinde kullanılabilecek kısa yolları bilmek sizlere çalışmalarınızda çok fazla zaman kazandıracaktır.

Jupyter'de hücreler komut modunda (command prompt) ya da yazma modunda (edit mode) olabilir. Komut modunda olduğunda hücre içinde yanıp sönen bir imleç görünmez ve hücre pasif durumdadır. Yazma modunda ise hücre içinde yanıp sönen bir imleç görürsünüz. Bir hücreyi komut moduna almak için e*scape*, yazma moduna almak içinse *enter* tuşları kullanılır.

Komut Modu Kısayolları:

Kısayol	Açıklama
Ctrl-Enter	Hücreyi çalıştır
Shift-Enter	Hücreyi çalıştır ve bir sonraki hücreye geç
Alt-Enter	Hücreyi çalıştır, altına yeni bir hücre aç
Enter	Yazım moduna geç
Y	Hücreyi kod (Code) formatına çevir
M	Hücreyi Markdown formatına çevir
R	Hücreyi Raw (Ham) formata çevir
1,2,3,4,5,6	Heading 1,2,3,4,5,6 formatına çevir
Yukarı Ok / K	Bir üstteki hücreye geç
Aşağı Ok / J	Bir alttaki hücreye geç
A	Hücrenin üstüne yeni bir hücre ekle
B	Hücrenin altına yeni bir hücre ekle
X	Seçili hücreyi kes
C	Seçili hücreyi kopyala
Shift-V	Kopyalanmış hücreyi yukarı yapıştır
V	Kopyalanmış hücreyi aşağı yapıştır
DD	Seçili hücreyi sil
Z	Hücre silme işlemini geri al
Shift-M	Seçili hücreyi bir alttaki hücre ile birleştir
Ctrl-S	Dosyayı Kaydet
L	Satır numaralarını göster / gizle
O	Hücre çıktısını göster / gizle
Shift-O	Hücre çıktısını kayan pencerede göster
Escape	Hücre içinden çıkmak için
H	Kısayol ekranını göster
I, I	Çekirdeği (Kernel) durdur
0, 0	Çekirdeği (Kernel) yeniden başlat
Space	Sayfayı aşağı kaydır

Yazma modu kısayolları:

Kısayol	Açıklama
Tab	Yazılan kelimeyi otomatik tamamlar ya da bir kelime yazılmıyorsa bir sekme kadar boşluk bırakır.
Shift-Tab	Geçerli fonksiyon, komut vb hakkında yardım
Ctrl-A	Tüm hücreyi seç
Ctrl-Z	Geri al
Ctrl-Shift-Z	İleri al
Ctrl-Y	İleri al
Ctrl-Home	Hücre başlangıcına git
Ctrl-End	Hücre sonuna git
Ctrl-Sağ Ok	Bir kelime sağa git
Ctrl-Sol Ok	Bir kelime sola git
Esc	Komut moduna çık
Ctrl-M	Komut moduna çık
Ctrl-Enter	Hücreyi çalıştır
Shift-Enter	Hücreyi çalıştır ve bir sonraki hücreye geç
Alt-Enter	Hücreyi çalıştır, altına yeni bir hücre aç
Ctrl-Shift-Eksi	Hücreyi imlecin bulunduğu satırdan ikiye böl
Ctrl-/	Bulunan satırı ya da seçili satırları koda ya da yoruma çevir

SQL SORGULARI

Python ile ilgili bir kitapta SQL hakkında kısa da olsa bir bölüm olması tuhaf görülebilir. Ancak, halen birçok kurumda veri tabanlarının SQL yapısında olduğunu dikkate alırsak günlük uygulamalarınızda Python dahi kullanıyor olsanız, verileri çekerken SQL bilmeniz fayda sağlayabilir. Nitekim, "veri okuma" bölümünde ilişkisel veri tabanlarından da veri okumayı gördük. Python'dan ilişkisel verileri okurken SQL yapısını bilmek günlük çalışmalarda işinize yarayabilir.

İlişkisel veri tabanlarının tablolardan tabloların da satır (kayıt, gözlem vb) ve sütunlardan (değişken, özellik vb) meydana geldiğini daha önce belirtmiştik. Bir tabloda sorgu yapmak için kullanılabilecek ilk komut tablonun tamamını seçmek olabilir. Yani,

```
SELECT * FROM Tablo1
```

Bu komutta SELECT, seç anlamına gelir. * işareti ise hepsi anlamındadır. SELECT sütun seçmek için kullanılır. Sonuç olarak yukarıdaki komut "tablodan bütün sütunları seç" anlamına gelir. Dikkat ederseniz henüz satırlara ilişkin bir seçim yapmadık. Bu nedenle satırların da tamamı seçilmiş oluyor.

Bütün sütunlar yerine sadece seçtiğimiz bazı sütunları almak için * yerine istediğimiz sütun isimlerini aralarına virgül koyarak yazabiliriz.

```
SELECT Ad, Soyad, Musteri_No FROM Tablo1
```

Bazen istediğimiz sonuçların sadece farklı olanları görmek isteyebiliriz. Örneğin tabloda binlerce satır yer alıyor olabilir ama bize sadece ürün kodlarının listesi gerekiyordur. Sadece farklı kayıtları görmek için DISTINCT anahtar kelimesini kullanabiliriz:

```
SELECT DISTINCT urun_kodu FROM Tablo1
```

Tablodaki bir kaydın sayısını görmek için COUNT() fonksiyonunu kullanabiliriz. Örneğin tabloda kaç satır olduğunu görmek için:

```
SELECT COUNT(*) FROM Tablo1
```

Tabloda kaç tane farklı müşteri olduğunu görmek için:

```
SELECT COUNT(müşteri_no) FROM Tablo1
```

Sadece farklı kayıtların sayısını görmek için COUNT ve DISTINCT kelimelerini bir arada kullanabiliriz.

```
SELECT COUNT(DISTINCT urun_kodu) FROM Tablo1
```


Bu aşamaya kadar istediğimiz sütunların bütün kayıtlarını seçtik. Sadece belirli şartı sağlayan satırları seçmek istersek WHERE kelimesini kullanmamız gerekir. Where kelimesinin Türkçe karşılığı, "nerede"dir. Ancak SQL sorgulaması açısından ben olan diye okumayı daha kolay buluyorum. Aşağıdaki sorguyu dikkate alalım.

```
SELECT Ad, Soyad, Musteri_No, Borc FROM Tablo1 WHERE Borc > 1000
```

Sorguyu Türkçe'ye çevirecek olursak şöyle diyebiliriz: Tablo1'**DEN** (FROM), Borc tutarı 1000'den büyük **OLAN** (WHERE) Ad, Soyad, Musteri_No sütunlarını **SEÇ** (SELECT).

Birden fazla şarta göre AND ve/veya OR anahtar kelimelerini kullanarak seçim yaptırmak da mümkündür.

```
SELECT Ad, Soyad, Musteri_No, Borc
FROM Tablo1
WHERE Borc > 1000 AND Gecikme_Gun_Sayisi > 10
Şimdi aşağıdaki sorguyu dikkate alalım:
SELECT * FROM Tablo 1
WHERE Borc >= 1000 AND Borc < = 5000
Aşağıdaki sorgu aynı işlemi yapar.
SELECT * FROM Tablo 1
WHERE Borc BETWEEN 1000 AND 5000
```

Seçimimizi uzun bir liste içinden yapacaksak listedeki her eleman için eşitlik yazıp aralarına AND koymak çok fazla zaman kaybına yol açar. Bunun yerine IN kullanabiliriz.

```
SELECT * FROM Tablo 1
WHERE Musteri_No IN (123, 234, 345, 456, 567, 678)
```

Benzer şekilde metin tipi veriler için kullanılabilecek bir operatör de LIKE operatörüdür. Bu da belirli bir formattaki metin verileri bulmak için kullanılır. Örneğin, A ile başlayan müşterileri bulmak için aşağıdaki sorguyu yazabiliriz.

```
SELECT Ad, Soyad, Musteri_No
FROM Table1
WHERE Ad LIKE 'A%'
```

% işareti tüm karakterleri kapsar. Sadece bir karakter için alt çizgi (_) karakteri kullanılır. NOT LIKE terimi de sorguda olmasını istemediğimiz kayıtlar için kullanılır.

Sorgularda toplam ya da ortalama değerleri hesaplatmak da mümkündür. Örneğin, gecikmede olan borçların toplamını görmek için

```
SELECT sum(Borc)
FROM Tablel
WHERE Gecikme_Gun_Sayisi > 0
```

Benzer şekilde ortalama için avg(), minimum için min() ve maximum için de max() fonksiyonları kullanılır.

Şimdi sorgularımızı bir adım daha ilerletelim. Yukarıdaki sorguda 1000 TL'den daha fazla borcu olan müşterileri seçip bunların sadece Ad, Soyad, Musteri_No ve Borc sütunlarını çektik. Yaptığımız sorguda satırların Gecikme_Gun_Sayisi sütununa göre sıralanmasını istiyoruz. Bunun için de ORDER BY komutunu kullanacağız. Order by komutunu Türkçe'ye "e göre sırala" şeklinde çevirebiliriz.

```
SELECT Ad, Soyad, Musteri_No, Borc
FROM Tablol
WHERE Borc > 1000
ORDER BY Gecikme_Gun_Sayisi
```

Sıralamayı büyükten küçüğe yapmak içinse ilgili satırın sonuna DESC kelimesi eklenir.

```
ORDER BY Gecikme_Gun_Sayisi DESC
```

Sorgularımızı belirli kategorik değişkenlere göre gruplandırmak da mümkündür. Sözgelimi çalışanların sayısını eğitim seviyesi gruplarına göre görmek istiyoruz. Aşağıdaki sorgu, her eğitim grubunda kaç çalışan olduğunu gösterecektir.

```
SELECT egitim, count(*)
FROM calisanlar
GROUP BY egitim
```

İlişkisel veri tabanlarının birbirlerine anahtar verilerle bağlı çok sayıda tablolardan oluştuğunu biliyoruz. "İlişkisel Veri Tabanlarından Veri Okuma" bölümünde kütüphane veri tabanı örneğini vermiştik. Bu veri tabanı, Kitap, Üye, Ödünç Kitap ve Yayınevi Listelerinden oluşuyordu. Zaman zaman farklı veri tablolarındaki verileri birleştirerek yeni bilgiler elde etmek isteyebiliriz. Örneğin ödünç listesinde sadece kitabı alan üyenin numarası yer alıyor. Buraya, üyenin diğer bilgilerini de getirmek isteyebiliriz. SQL'de farklı tablolarda yer alan bilgileri bir araya getirme JOIN komutu ile gerçekleştirilir. Son olarak Üye Listesi ile Ödünç Listesini birleştirip hangi kitapların hangi müşteride olduğunu görelim.

```
SELECT Adi, Soyadi
FROM uye_listesi
JOIN odunc_listesi
ON uye_listesi.uye_no = odunc_listesi.uye_no
```