Pandas ile Numpy birbiribe yardımcı kütüphaneler olarak düşünülebilir. Pandas ile farklı dosya türlerineden gelen bilgileri kolaylıkla kullanılabilir hale getirebiliriz.

Numpy’ın aksine liste içerisine heterojen yapıda değerler atayabiliriz, örneğin string ile integer pandas’ta aynı listede olabilir, pandas kütüphanesi her bir değişkeni “object” tipine dönüştürür.

“pd.Series(some)” komutuyla some değişkenini indexleri belirtilecek şekilde liste içerisine atayabiliriz. Fonksiyon içerisine (some’ın yanına) liste olarak index değerlerini eklersek değerleri fazladan girdiğimiz index kadar okutabiliriz. Bu liste içerisindeki indexler bir sayı olmak zorunda değildir, farklı parametrelere de sahip olabilir.



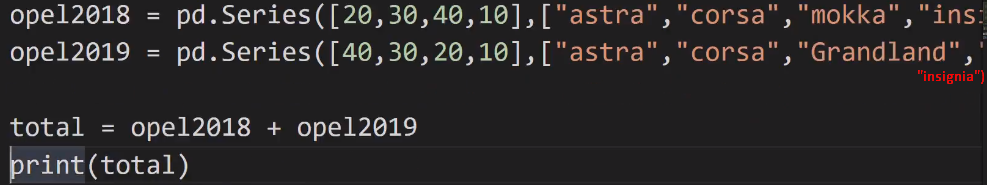


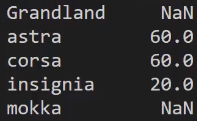
Bu fonksiyonu dictionary ile kullanırsak sistem index olarak key’i karşısındaki değer olarak da value’yi tayin eder.

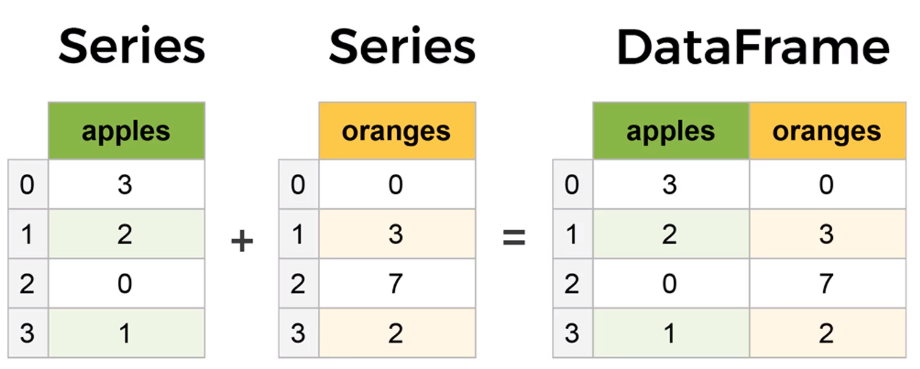
Liste içerisindeki değer çekme işlemleri için numpy notlarına bakabilirim. Ondan farklı olarak indexte rakam dışında atanan değerler string olarak belirtilerek (Örn: pandas\_series[“d”]) değer çekilebilir. Olmayan bir index isteği girdiğimiz takdirde NaN cevabını alırız.

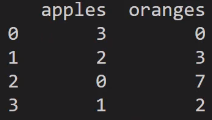
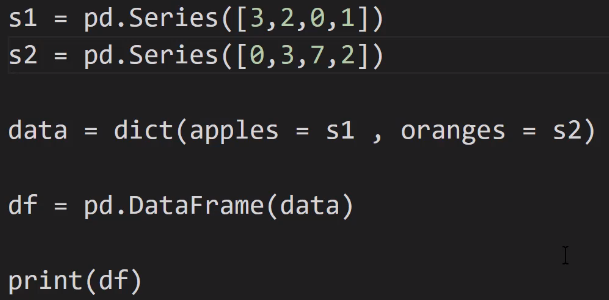


“.ndim” komutu ile boyutu öğrenebilir, “.dtype” komutu ile data type’ı bulabilir, “.shape” komutu ile kaça kaçlık bir liste olduğunu, “.sum()” metodu ile değerler toplamını hesaplayabiliriz. “sum()“ gibi diğer matematiksel komutlar için Numpy kütüphanesine bakabilirim.





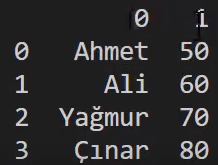
Bu kütüphanede DataFrame adında önemli bir kavram bulunuyor, bu kavram serilerin birleşmesidir. 



“dict()” fonksiyonuyla bir data değişkeniyle dictionary oluşturduk, “pd.DataFrame(param)” fonksiyonu ile data frame yapısına dictionarymizi entegre ettik ve yazdırdığımızda sağdaki resimdeki şekilde bir output aldık.

“.DataFrame()” komutuna index eklemezsek sıra ve sütunu sıfırdan başlatarak rakamlarla ilerler, parametre de eklemediğimiz takdirde “Columns: []\nIndex: []” cevabını alırız.



Eğer sıra ve sütunlarda belirlediğimiz değerleri istersek;

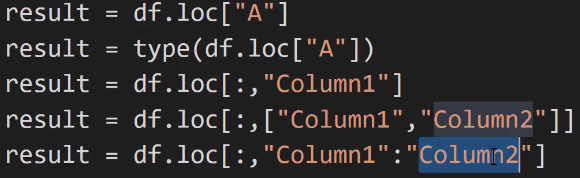


Liste değerlerini data olarak tanımlayıp fonksiyon içerisine örnekteki gibi değer atayabiliriz, ikinci index indexi, üçüncü index sütunu belirtir fakat “columns =” veya “index =” diye konum belirttiğimiz takdirde istediğimiz sırayla komut girebiliriz. “dtype = float” olarak ayrı bir komut atadığımızda değerleri integer formdan alıp float’a çekebiliriz.

[**Kaynak**](https://www.kaggle.com/datasets) **🡪** Linkten birçok farklı konuda dataset ile ilgili işlem yapılabilir.

Farklı datasetleri koda çekmek için “pd.read\_(filecode)” komutunu kullanırız (filecode olarak .json,.csv vb olabilir).

“df.loc[“”]” komutu ile dataframedeki herhangi bir indeksin sıra veya sütün değerini çekebiliriz, sıra değeri çekmek için “.loc[“row”]”, sütun değeri çekmek için “.loc[“:”,”column”]” şeklinde ifade etmemiz gerekir, aşağıdaki örnekte uygulama şekillerini görebiliriz.



Index değerini özelleştirmiş olsak bile “df.iloc[numb]” komutu ile standart index belirterek istediğimiz indexe ulaşabiliriz.



Oluşturduğumuz dataframe’e yeni bir sütun eklemek istersek aşağıdaki yöntemi kullanabiliriz.



“df.drop(“column”, axis = 0/1)” komutu ile dataframe’den sütun silme (sütun için 1, satır için 0) işlemi yapabiliriz, eğer bu silme işleminin bir kopya halinde değil de (bu haliyle df üzerinde bir değişiklik olmaz) df’nin kendisi olmasını istersek axis bilgisinin ardından “inplace = True” değerini girmemiz gerekir, böylece df matrisi sütun çıkarılmış halini alır.



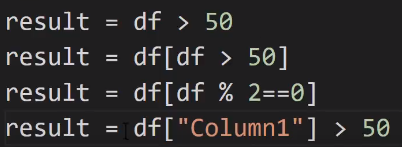
“df.columns” metodu ile dataframe’deki sütun isimlerini ve bu sütunları tipini öğrenebiliriz. “df.head” komutuyla beş sütuna sahip matristen ilk beş değeri çektik, burada anladığım kadarıyla head komutuyla sıra/sütun sayısı eşit şekilde ilk değerleri önümüze getiriyor, aynı durum “df.tail” komutuyla son beş değeri çekmek için de geçerli.



Şeklinde kullandığımız takdirde “Column1” sütununun ilk beş değeri aktarılır. “df.Column1.head()” komutu da aynı işlevi görmektedir.



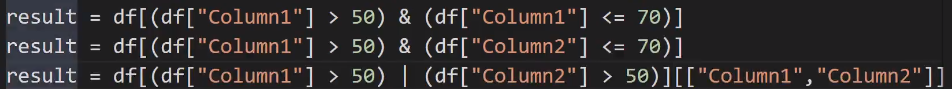
Örnekteki gibi “def[5:15]” şeklinde bir komut eklersek “.head” için 6:10 arası değerleri çekerken “.tail” için 11:15 arası değerleri çekecektir.



Yukarıdaki örnek, 15x5 martisinin filtreleme kısmıdır, bu filtrelemede ilk komutu uygularsak dataframedeki tüm değerler koşulu sağlayıp sağlamamalarına göre True/False olarak gösterilecektir, ikinci satıra göre ise elliden büyük olanlar sayı olarak gözükecek, küçük olanlar ise “NaN” cevabı olarak yansıtılacaktır. Dördüncü örnekte ise “Column1”’deki elliden büyük olan değerler True/False olarak belirtilir.

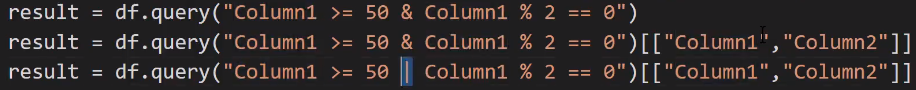


Dördüncü örneğin bu hali ise “Colum1” değerlerinin elliden büyük olan indexlerinin oluşturulduğu ve ilk iki sütunun (Column1, Column2) gösterildiği koda hitap eder.

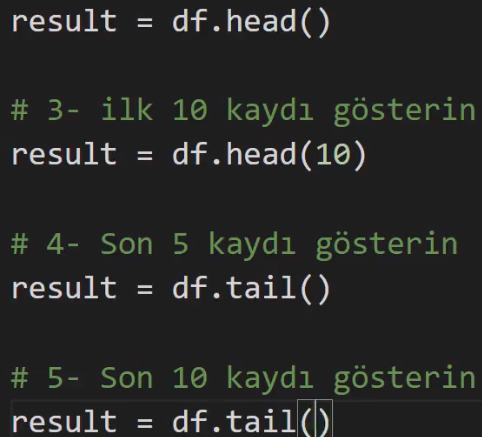


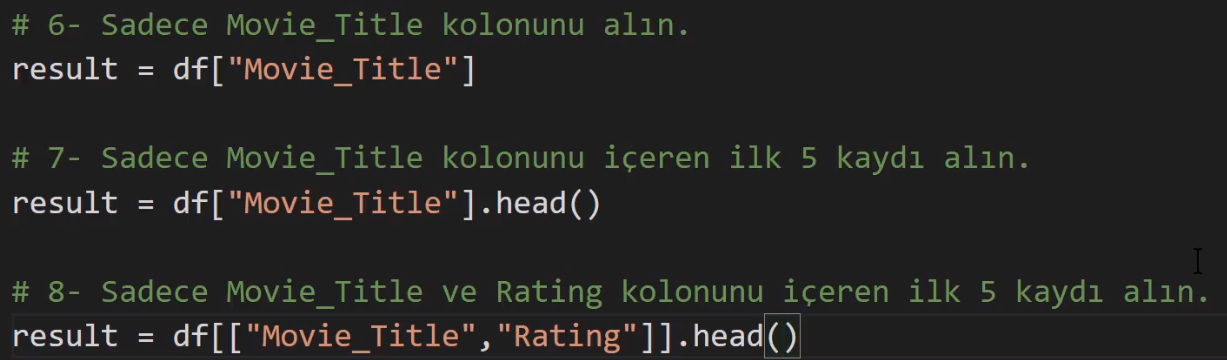
Yukarıda örnekte olduğu gibi df database’ini birden fazla koşulla kontrol edebiliriz, bu koşullar &(and) |(or) komutları ile birleştirilebilir, üçüncü örnekte dataframe, ilk iki sütunda bu değerler gözetilecek şekilde filtrelenmiştir.

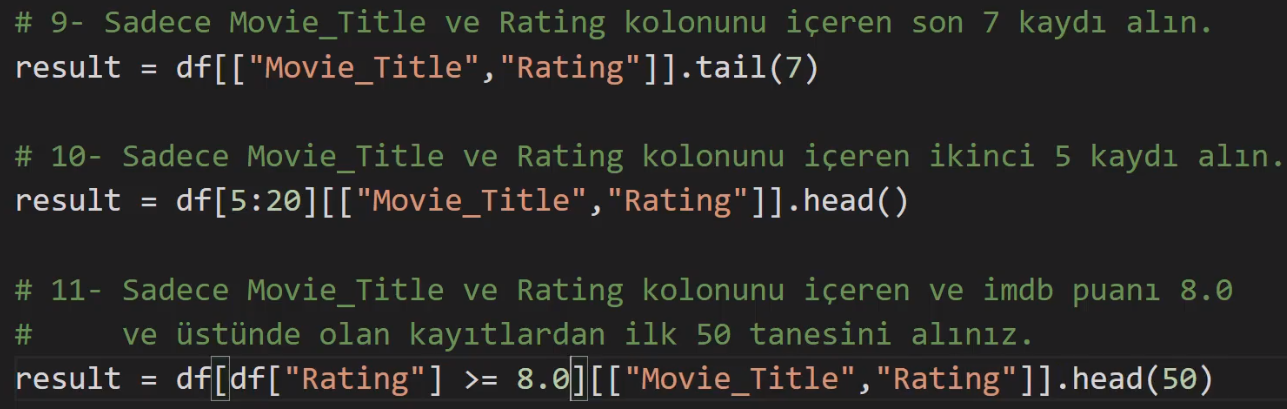
“df.query(“conditioning”)” komutu ile koşullu filtreleme işlemini quote içerisine uygulayarak çok daha hızlı bir şekilde uygulamamız mümkün.

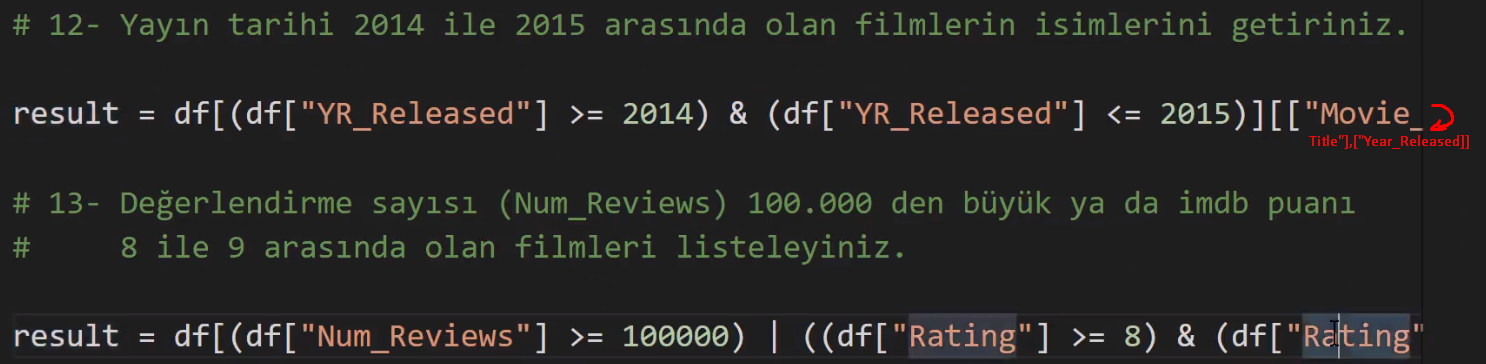


**IMDB Veri Uygulaması**

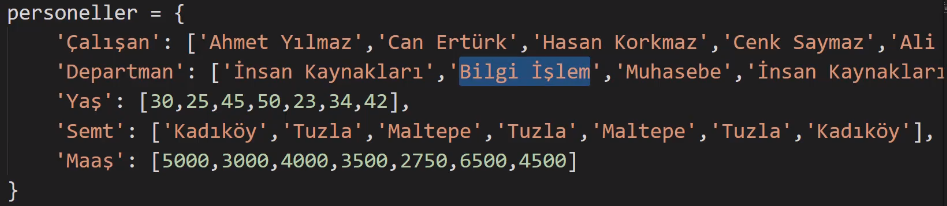




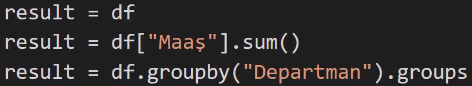




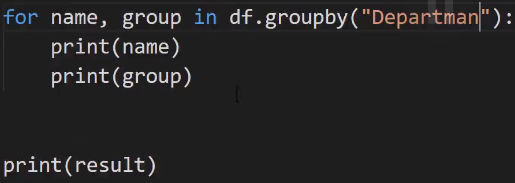




“pd.DataFrame()” komutu dictionary ile uyumlu çalışır. İkici kod ile örnekteki çalışan maaşlarının toplamını yazdırabiliriz. üçüncü kod ile “Departman” sütunundaki değişkenleri indexlerine göre ikinci resimdeki gibi yansıtabiliriz. “.groupby()” komutu liste içeriklerini de desteklemektedir. Bu işlemler değişkenleri filtlereyerek gruplandırmaya olanak tanır.









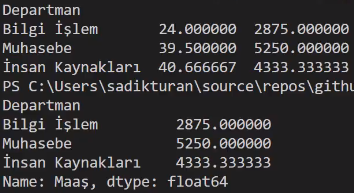
For döngüsü kullanmadan bir klasman içerisindeki grupları çekmek istersek “get\_group(“group2”)” komutu ile istenilen grubu çekebiliriz.



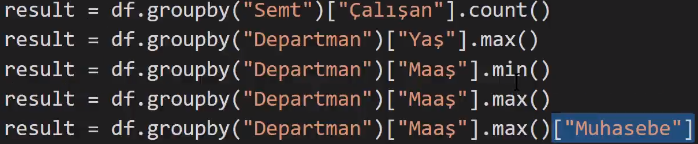
“.mean()” ve “.sum()” komutlarını “.groupby()”’ın ardından uygulayarak ortalama ve toplam değerlerini aşağıda örnekte görüldüğü gibi elde edebiliriz.



Üstteki örneğin ikinci satırında departmana göre maaş miktarlarını görüntüledik.

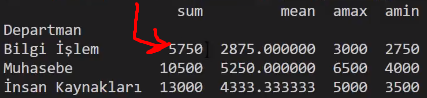


“.count” komutuyla kodun sayısal değerlerini index vesaire belirtmeden direkt çekeriz ve yalnızc değerler yansıtılır. “.max/min()” komutlarıyla DataFrame’deki minimum ve maksimum değerleri çekebiliriz.



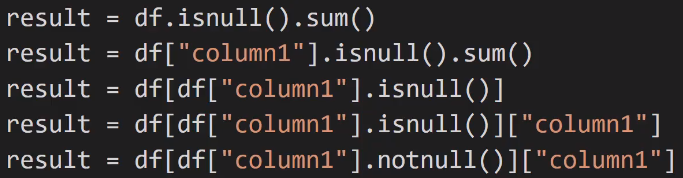
“.agg()” komutu numpy kütüphanesiyle birlikte kullanılır. Burada komutun kullanım amacı birden fazla işlemi tek bir yerden yapabiliyor oluşumuzdan kaynaklanır, kısacası belirlenen eksen üzerinde bir veya daha fazla işlem yapabiliriz.



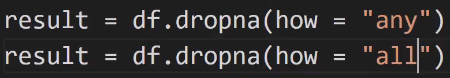


“df.reindex([])” metoduyla oluşturduğumuz indeksi yeniden oluşturabiliriz, eğer değerlere karşılık gelen bir indeks yoksa, o indekslerde “NaN” cevabını alırız.

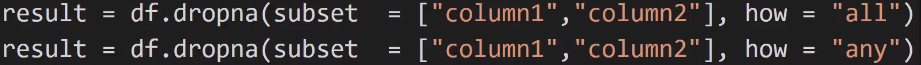
“df.isnull()” komutu ile değer içeren parametreler False, NaN olanlar True olarak gösterilir, “df.notnull()” ise bu işlevin tam tersini sağlar.



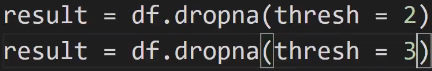
“df.dropna()” komutu ile NaN olan satırlardaki (axis = 0) değerler silinir, parantez içerisine “(axis = 1)” yazdığımız takdirde NaN sütunu silme işlemi yapılır, NaN değeri bir adet olsa bile ekseni siler.



Örnekte görüldüğü gibi “how = any” komutu default değerdir, tamamının NaN olması gerekirse “how = all” diyerek koda bu işlemi yaptırabiliriz.

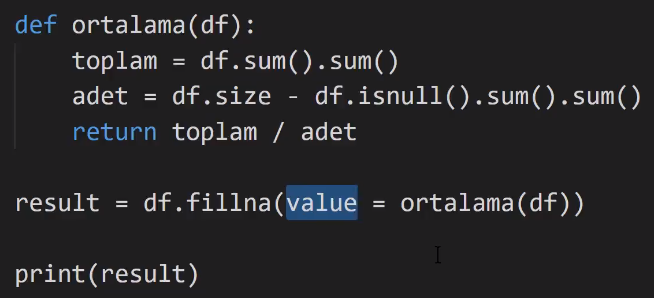


Parantez içerisine yazılan “subset = [“column”]” komutu ile aranacak NaN’ları sütun içerisinde sınırlayabiliriz.

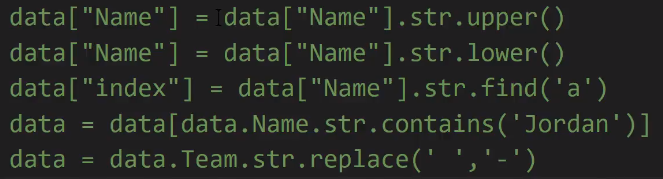


Parantez içerisinde “thresh = number” komutuyla sütunda (ilk örnek için) 2 veya daha fazla sayı varsa satırı yazdırmasını bir işlem yapabiliriz.

“df.fillna(value = “some”)” komutu ile NaN değerlere “some” kelimesini yazdırabiliriz, bu değer bir kelime değil de tam sayı da olabilir

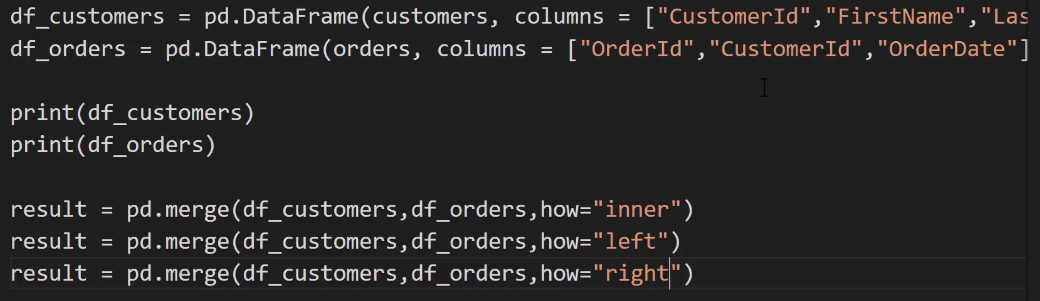


Yukarıdaki uygulama ile Dataframe şeklindeki verilerin NaN değerlerini matrisin ortalama değerlerine dönüştürdü.



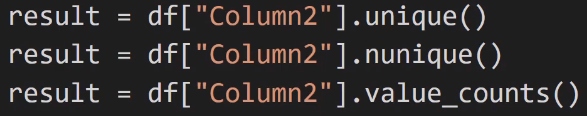
İlk komutta “Name” altındaki stringlerin tüm harflerini büyük, ikincide küçük yazdırdık. Üçüncü komutta “Name” içerisindeki kelimelerde “a” karakterini içerenlerin indeksini çektik. Dördüncü komutta ise “Name” içerisinde “Jordan” kelimesini içeren satırları aldık, son komutta “Team” içerisinde boşluk karakteri içeren takımları “-“ ile değiştirdik.

[Kaynak](https://www.btkakademi.gov.tr/portal/course/player/deliver/s-f-rdan-ileri-seviye-python-programlama-5877) 🡪 Dakika 10’da isim/soy-isim verilerini birbirinden ayırıp farklı indekslere yerleştirmekle ilgili bir komut yazılıyor, ihtiyacım olursa bakarım.

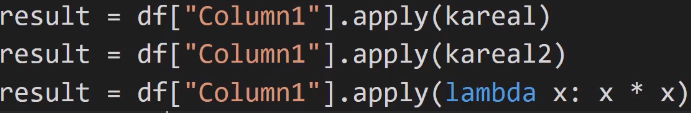


“customers/orders” adlı önceden oluşturulmuş 2 farklı dictionary’nin dataframe’ini oluşturmamızın ardından “pd.merge(firstframe, secondframe, how=”command”)” komutu ile iki farklı dataframe’i birbiriyle birleştirerek SQL joins metoduna göre verileri farklı biçimlerde çekebiliriz, “inner” komutu iki tablodaki ortak verileri (bu tablo için CustomerId: 1, 2: olanları), left dersek “customers” tablosunu, right dersek “orders” tablosunu sistem getirecektir. “outer” dersek sağ ve saol tablodaki tüm kayıtlar getirilir.

İki farklı dataframe’i birbiriyle birleştirmek için “pd.concat([firstframe, secframe], axis=0/1)” komutunu kullanabiliriz, “axis = 0” ikinci dataframe’i sıra olarak eklerken (alt alta), “axis = 1” ikinci dataframe’i sütun olarak ekler (yan yana). Sıra olarak ekleme yapacağımız takdirde “axis = 0” diye yazmak zorunda değiliz çünkü komutun default hali bu şekildedir.



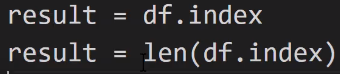
“.unique()” metoduyla sütun içerisindeki tekrarlanmayan değerleri çekebiliriz, “.nunique()” metodu ise bu tekrarlanmayan değerlerin kaç adet olduğunu bize gösterir. “value\_counts()” komutu ile hangi değerden kaç adet olduğunu öğrenebiliriz.



“.apply(function)” komutuyla def fonksiyonu olarak kodladığımız bir işlem topluluğunu dataframe içerisinde kullanmamız mümkün hale gelir, bu komut def fonksiyonunu kabul ettiği gibi lambda işlemlerine de olanak sağlar.

 Sütun bilgilerini ifade eden komut.

Kaç adet sütun var görebiliriz.

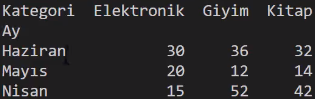


“.index” metodu sıra indexlerinin ne olduğunu (kaçtan başlayıp kaçta bittiğini vs) ifade eder. “df.info” ifadesi bu özellikler ile ilgili detaylı bilgi verir.

“df.sort\_values(“columnx)” komutu ile “Columnx”’deki ifadeler bize sıralı biçimde gelir (küçükten büyüğe doğru/alfabetik sıraya göre), “ascending = False” dersek (default olarak True değeridir) sıralamayı büyükten küçüğe veya alfabenin sonundan başına doğru yapar.

“df.pivot\_table(index= “some1”, columns = “some2”, values = “some3”)” komutu ile dictionary içerisindeki parametreleri indeks, sütun ve değer olarak istediğimiz şekilde atayabiliriz.





Sqlite3?