**PANDAS**

**APPEND, CONCAT, MERGE, JOIN**

|  |
| --- |
| **\*\*append() ve concat()\*\* ile dikey birleştirme yapabiliriz.**  **\*\*concat(), join() ve merge()\*\* ile yatay birleştirme yapabiliriz.**  **\*\*append() ve concat()\*\* ile aynı anda iki veya ikiden fazla dataframe i birleştirebiliriz.**  **\*\*append() ve concat()\*\* da ortak sütun olması gerekmez.**  **\*\*concat()\*\* ile hem satır hem de sütun bazlı birleştirme yapabiliriz. Yani "axis" sadece concat() de var.**  **\*\*join()\*\* de aynı anda iki veya daha fazla dataframe i birleştirebiliriz ancak aynı sütun isimleri varsa en fazla iki df birleştirebiliriz.**  **\*\*merge()\*\* ile aynı anda iki dataframe i birleştirebiliriz.**  **\*\*merge()\*\* ile birleştirme işlemimizi ortak sütun üzerinden yapıyoruz.** |

#concat, merge, append ve join methodlarini gorecegiz.

#merge'u kullanmadan yapamayacaklarimiz var

#hem append hem concat'la yapilabilenler var.

#satirlari alt ata getirerek dikey birlestirme yapmis oluyuoruz

#append dikey, merge, join, concat ise yatay birlesme

#birden fazla df birlestirebiliyoruz.

#ayni sutun ismi yoksa merge ile yapamiyoruz.

#merge'de ancak sutunlar ayni ise birlestirebiliyor.

#concat, append ortak sutun olmasa da yapiyor.

#karsilik gelen deger yoksa nan ile dolduruyor bazilari

**APPEND (Vertical)**

append() fonksiyonu, iki veri kümesini birleştirmek için kullanılan en temel yöntemdir.

#append alir vertical birlestirir. yani satirlari altalta birlestirir vertical olarak

**one.append(two)**

#boyle yazarsam vertical bir birlestirme yapiyor

#en temel birlestirme bu

#2 kere ayni index'le deger veriyor.

#ama bu karisikliga sebep oluyor

#bunu duzeltmem gerek

#ignore\_index=True diyince 0’dan baslayip yeniden sirali index veriyordu, bu sekilde duzeltebilirim.

**new\_df=one.append(two, ignore\_index=True)**

#append ile ayni anda birden fazla df ekleyebilirsin

#kalici degisiklik yapmasi icin atama yapman gerek

**CONCAT (AXIS=0,1)**

concat(), iki veya daha fazla DataFrame'i satırlar veya sütunlar bazında birleştirmek için kullanılır.

Farklı veri kümelerini aynı veri kümesinde bir araya getirmek veya veri kümesine yeni sütunlar veya satırlar eklemek için kullanılabilir.

Veri kümeleri arasında herhangi bir ortak sütun gerekmez.

**pd.concat([one, two])**

**#pd.concat ([x,y])**

#concat alt alta hepsini koyuyor, olmayan degerlere de nan koyuyor. Default axis=0

#concat yazdigimiz axis'e gore birlestirir.

#axis=0 yazdigimizda vertical birlestirir concat. appendle ayni

#concat'da axis=1 yazdigimiz zaman horizontal birlestirme yapar, satirlari yanyana yazar

#concat'da inner ve outer var.

#innerj iki df’nin ortak sutunun indeksi alip getiriyor

#outerjoin ise hepsini alip getiriyor. karsiliginda deger yoksa nan atar.

#append'den farki concat'in axisle kullanilmasi

**pd.concat([one, two], ignore\_index=True, axis=1)**

#burda yanyana siraladik, kullanisli bir gorunum olmadi

#vertical daha mantikli burda

#istedigimiz satir ve sutunlari cagirtabiliyoruz iloc ile

**pd.concat([one.iloc[ :,:2], two.iloc[:,1:]])**

#one'in tum satirlari ile 2.sutuna kadar olan degerleri getir dedik

#two icin de, tum satirlari getir ve 1.sutundan sonuna kadar getir dedik

#one icin 2. indeksi dahhil etmedigimiz icin sutunda nan yapti

#two icin de 0. indeksi sutunda cagirmadigimiz icin nan yapti.

**MERGE (horizontal, HOW, ON)**

merge(), iki veya daha fazla veri kümesini, ortak bir sütuna göre birleştirme işlemidir.

Ortak sütunlardaki değerlere göre veriler birleştirilir ve tek bir veri kümesi oluşturulur.

merge(), veri kümesindeki eksik veya farklı değerleri tamamlamak veya farklı veri kaynaklarını bir araya getirmek için sıkça kullanılır.

#merge'de **how** parametresi var. **left right inner ve outer** var how’a cvp olarak

#ortak bir sutun olmasi lazim iliskilendirmesi icin

##**on** parametresine o sutunu yaziyorsun, sql'deki gibi

#merge'de **horizontal** yanyana siralama yapacak.

#bu da ortak sutun uzerinden gidecek

#kullanimi da concat gibi ama **ON** ile ortak sutunu yazarak birlestiriyoruz.

#merge yaparken foreign key gibi iliskilendirebilecegi bir sutun uzerinden birlestiriyor

#iki df birlestirdim int degerler var diyelim, nan degerler gelecegi/geldigi icin int degerler de float'a donusur.

**pd.merge(x, y, how="inner", on="key")**

#on ile key yazmasak da anliyor aslinda ortak sutunun key oldugunu

#horizontal bicimda sutunlari yanyana birlestirdi

#zaten merge vertical birlestirme yapmiyor

#how'a inner dedigimiz icin K3'u almadi

#cunku K3 ortak degil, digerleri ortakti sadece

**pd.merge(x, y, how="outer")**

#outer diyince bu sefer K3'u de getirdi, ama x'ten deger gelmedigi icin K3 sutununun karsiligna nan koydu

**pd.merge(x, y, how="left")**

#left ile birlestirince, x'i aliyor. X’i left’e yazdim cunku

#x ile y'nin ortak degerleri getiriyor bir de

**pd.merge(x, y, how="right")**

#right ile birlestirince, y'yi aliyor. Right’da y var cunku

#x ile y'nin ortak degerleri getiriyor bir de

**#left\_on, right\_on** gibi methodlari var

**pd.merge(df1, df2, left\_on="lkey", right\_on="rkey", how="inner")**

#left\_on'a ilk yazdigimiz yani soldaki df'in sutununu

#right\_on'a da 2. yani sola yazdigmiz df'teki iliski kuracagimiz sutunu yazariz

#y, a, b sutunlari ortak olmadigi icin onlari almadi

#cunku inner dedik how icin.

#x icin 3 deger var,ikisi ilkinden, biri 2.den geliyor

#x'i alir 4. deger olarak yine 3. degerini yazar. 8

#outer deseydik hepsini birlestirirdi

#merge'de sadece 2 df birlestiriyoruz.

**JOIN(Horizontal, leftjoin, rightjoin, innerjoin, outerjoin)**

**df3.join(df4)**

**#how condition'in default hali merge'de inner**

**#join'in default hali ise left**

#yani lefti alir, sagdakiyle ortak olani da alir.

#horizontal sekilde birlestirir

**leftjoin** 1.nin hepsini getiriyor.

#**rightjoin** 2.nin hepsini getiriyor. ilkinin sadece ortak olanlarini getiriyor

#**innerjoin** sadece ortak olani getiriyor

#**outer join** hepsini getiriyor.

#ortak sutunlar varsa isimlerini A1 gibi farkli veriyor.

#outer yapsaydik, hepsini birlestirecekti.

#ama karsiligina value denk gelmezse nan doldurur.

**df3.join(df4, how="right")**

#right yapinca da sagin(df4) hepsini alir, ortak olanlari da alir, ortak olanlari getrir

#bos olan yerlere nan doldurur

**df3.join(df4, how="outer")**

#outer yapinca tum satirlari alir

**df3.join(df4, how="inner")**

#inner'da her zaman kesisimini getirdigi icin hic null deger gelmez

**df4.join(df3)**

#boyle yaparsam da once C ve D gelir sonra A ve B

#yazim sirasina bagli

**#df5.join(df6)** boyle yazarsak hata verir. ortak sutunlar var der joinle birlestirirken

#lsuffix ve rsuffix leri kullanarak birlestirebilirim:

**df5.join(df6, lsuffix= "\_left", rsuffix="\_right")**

**df5.set\_index("key")**

#set index napiyordu. bir sutnu alip df'in indexi yapiyordu

#key sutunun yazarsam onu index yapar, boylece sorunsuz birlestirebilirim.

**df6.set\_index("key")**

#ikisine de ayni islemi yapinca artik sorunsuz birlestirebiliyorum

**TEXT AND TIME DATA**

**Text Data - String Methods**

|  |
| --- |
| **\*\*str.lower():\*\* Bir stringi küçük harfe dönüştürür**  **\*\*str.upper():\*\* Bir stringi büyük harfe dönüştürür**  **\*\*str.capitalize():\*\* İlk karakteri büyük harfe dönüştürür**  **\*\*str.title():\*\* Her kelimenin ilk karakterini büyük harfe dönüştürür**  **\*\*str.swapcase():\*\* Büyük harfleri küçük, küçük harfleri büyük harfe dönüştürür**  **\*\*str.isalpha():\*\* Tüm karakterlerin alfabede yer alıp almadığını kontrol ederek True veya False değeri döndürür**  **\*\*str.isnumeric():\*\* Tüm karakterlerin sayısal değer içerip içermediğini kontrol ederek True veya False değeri döndürür**  **\*\*str.isalnum():\*\* Tüm karakterlerin harf veya sayısal değer içerip içermediğini kontrol ederek True veya False değeri döndürür**  **\*\*str.endswith():\*\* Bir stringin belirtilen değerle bittiğini kontrol ederek True veya False değeri döndürür**  **\*\*str.startswith():\*\* Bir stringin belirtilen değerle başladığını kontrol ederek True veya False değeri döndürür**  **\*\*str.contains():\*\* Bir alt dizinin bir dizede bulunup bulunmadığını kontrol ederek her bir öğe için True veya False değeri döndürür**  **\*\*str.strip():\*\* Bir stringin başındaki ve sonundaki boşlukları temizler ve kırpılmış bir versiyonunu döndürür**  **\*\*str.replace():\*\* Belirtilen bir değeri başka bir değerle değiştirilmiş bir string döndürür**  **\*\*str.split():\*\* Bir stringi belirtilen ayraçta böler ve bir liste döndürür**  **\*\*str.find():\*\* Bir stringde belirtilen bir değeri arar ve bulunduğu konumun indeksini döndürür**  **\*\*str.findall():\*\* Bir desendeki tüm örneklerin bir listesini döndürür**  **\*\*str.join():\*\* Bir iterable ın öğelerini bir stringe dönüştürür** |

**#butun df uzerinde islem yapiyorsak str. olarak yaziyoruz fonksiyonu**

#yani df'in tum satir ve sutunlarinda islem yapmak istedigimde str kullanmam gerek.

**df.job.str.isalpha()**

#alfebedeki degerlerden mi olusuyor diye kontrol ediyor

#bosluk oldugu icin False verdi

**df.age.str.isnumeric()**

#neden NaN geldi, age sutununda **-** isareti oldugu icin, age sutununu object kategorisinde goruyor.

#o yuzden int gormedigi icin nan verdi.

#ben age sutunun kategorisini object'den str'ye cevirirsem, o zaman check edebilirim

**df.age.astype("string").str.isnumeric()**

# **“-”** bir str ifade oldugu icin, sutunu int'e degistiremedik.

#o yuzden sutunu komple str'ye cevirdik. Sonra **str.isnumeric** mi diye check ettik

**df.sales.str.isalnum()**

#hem alpha hem numeric birlikte kontrol eder isalnum

#icerdigi karakterler hem numeric hem alfabetikse True doner

#$ isareti dolayisisyla False verdi digerlerine.

#$'i ozel karakter olarak aliyor alfabetik olmayan

#sadece space tespit eden **isspace** var mesela

**df.sales.str.isascii()**

#ascii karaktere sahip olanlara True dondurur bu da

**df.job.str.startswith("f")**

#neyle basladigini kontrol etmek icin kullanilir

#mesela f harfi ile baslayanlari getir gibi

#bool type donduruyor hep bunlar

**df.job[df.job.str.startswith("data")]**

#boyle yazarsak da bool type degil dogrudan degerleri verir

**df.job.str.endswith("st")**

#st ile bitenleri getir

**df.job.str.contains("data")**

#icinde data contain edenleri verir

**df[df.job.str.contains("data")]**

#koseli paranteze alip basina df yazarsan, o kosulu saglayan degerleri getirir.

#datayi temizlemek icin kullandiklarim: strip, lstrip, rstrip

**df.sales.str.strip("$, dolar")**

#bu mesela $ ve dolar ifadelerini temizledi.

#yukardakinin aynisini yapiyor.

**df.sales.str.rstrip("dolar").str.lstrip("$")**

#sales sutununda, sag tarafta isime yaramayan dolari, sol taraftaki $ isaretini temizliyor

**ya da**

**df.sales.str.strip("dolar$")**

#tum bunlar dolar ve $’dan kurtariyor datalari

**df.sales.str.strip("dolar$").replace(",", "")**

#replace yazarsam boyle str yazmadan, aradan "," secip silmez.

#bazilarinda , var bazilarinda yok ya hepsini tek tip hale getirmeye calisiyorum ki int'a cevireyim

**df.sales.str.strip("dolar$").str.replace(",", "")**

#str.replace yazarak, dogrudan datanin icine girip duzenleme yapabildim.

#artik virgullerden kurtulmus oldum

#ama dtype hala object bunu astype ile float'a cevirebilirim.

**df.sales=df.sales.str.strip("dolar$").str.replace(",", "").astype("float")**

**df**

#bu sekilde temizlenmis ve float'a cevrilmis datayi yeni bir degiskene atadim

#boylece degismis oldu df.

#birden cok method girince icine, inplace=True olmasina ragmen kalici degisiklik yapmaz.

##ozetle dolar ve $ yazilarini attim, virgulleri attim, floata cevirdim, yeni bir degiskene atayarak kalici hale getirdim.

#age'deki - isaretinden de kurtulmak istiyorum ki float'a cevirebileyim

#-'yi null yaparsam float'a cevirebilecegim. Cunku NaN’in tipi float.

**df.age.replace("-", np.nan)**

**df.age=df.age.replace("-", np.nan)**

#str.replace df’in tum datalari uzerinde islem yapar.

#yazdigimiz kosulu tum df'nin datalarina uygulatabiliriz. Datanin icine girerek donusturtebiliriz.

#replace ise tek bir satir, ya da tek bir deger uzerinde islem yapar. Komple degistirir.

#mesela arada virgul var diyelim, bunu replace ile yapamam.

#ama tek bir deger var ve onu komple baska bir seye degistirmek istiyoprum o zaman ancak replace yapabilirim.

**df.name.str.title().str.split().str[0]**

#bunu title'a cevirip strip ile ayirip, 0. index'teki elemani alirsam isim kismini

#boylece ad soyad olarak ayirabilirim

**df.name.str.title().str.split().str[1]**

#1. indeksi alirsam da soyadini alir.

**df["first\_name"]=df.name.str.title().str.split().str[0]**

**df["last\_name"]=df.name.str.title().str.split().str[1]**

#Kalici olmasi icin yeni degerlere atadim

#artik name sutununu dusurmem gerek

**df.drop("name", axis=1, inplace=True)**

#artik kalici olarak name droplandi

**df.job.str.find("developer")**

#icine yazdigimiz seyi arayacak ve degerin index degerini dondurur

#bulamadiklari icin -1 dondurur

#buldugu yerin indexini verir.

#9 demek d"nin ilk basladigi yerin indeksi demek

#frontend developer kelimesinde 9. indekste basliyor "developer"

**df.job.str.findall("er")**

#finall ise, indexi deger degeri donduruyor,

#yani buldugu ifadeyi liste icinde bize getirdi

#yoksa bos liste getiriyor.

**df.job.str.findall("d")**

#find index getirirken findall deger getiriyor liste icinde

**df.job.str.findall("d").count()**

#burda count kac tane satir varsa onu saydi, d”yi saydirmadi.

#peki ben kac tane d oldugunu nasil bulacagim

#count, len gibi ifadeler hepsini sayar.

#ama apply'i veya transformu kullanirsak, herbir satirda islem yapacagi icin

#herbir satirda d'yi kac tane bulduysa onu yazar apply ve transform(len) ile

**df.job.str.findall("d").apply(len)**

#apply olmadan len veya count kullanirsak tum satirlari sayar getirir

#ama ben satirlar bazinda calissin istiyorsam, her satirda ne kadar var bana onu getirsin istiyorsam **apply ile len** kullanirim.

**df.skills.str.join(",")**

#herbirini alsin virgulle ayirsin liste disinda birlestirsin istiyorum

#artik liste icinde degil, disari cikardi

#kelimeler , ile birbiriyle birlesmis halde cikti.

#df'de bazi sutunlarda liste icinde bazi sutunlarda ayri ayri tableu, ptyhon, sql var.

#tableu, ptyhon, sql ayri bir sutun olsun istiyorum:

#liste ise onu al join kullan , ile birlestirerek listeden cikar,

#degilse aynen birak diyecegim.

**df.Skills.apply(lambda item: ",".join(item) if type(item)==list else item)**

#if yapisi kullanarak, dedim ki tipi listeyse , ile birlestirerek listeden cikar

#listede degilse(else) aynen getir

#tableu, ptyhon, sql ayri bir sutun olsun istiyorum nasil yaparim?

#df'de Skills sutununda liste icinde ve disinda datalar var.

#oncelikle onlari tek tipe cevirmem lazim

#liste ise onu al join kullan , ile birlestirerek listeden cikar,

#degilse aynen birak diyecegim.

**df.Skills.apply(lambda item: ",".join(item) if type(item)==list else item)**

#if yapisi kullanarak, dedim ki tipi listeyse , ile birlestirerek listeden cikar

#listede degilse(else) aynen getir

**DUMMY OPERATIONS**

**get\_dummies(), makine öğrenimi ve veri analizinde yaygın olarak kullanılan bir işlevdir.**

**Genellikle kategorik değişkenleri veya text datalarını, makine öğrenimi algoritmaları tarafından kolayca işlenebilen sayısal veya binary dönüştürmek için kullanılır.**

#get\_dummies()

#kategorik sutunlari sayisala donusturmek icin kullanilir.

#makine ogrenmesinde kullaniyoruz daha cok

#makineye bilgileri verip, anlamli sonuclar cikarmasini saglamak icin kullaniyoruz

#kategorik degerleri alir tek basina sutun yapar

#daha sonra da bunu numerik hale cevirir

#M'nin oldugu yere 1 yazdi, olmayan yerlere 0 yaziyor

#bunu L ve S icin de yapiyor.

#binary degerler 0 yok 1 var anlaminda, buna donusturmus oluyor

**pd.get\_dummies(df.dept)**

Dept sutununda, HR ve IT vardi 5 satir seklinde. Bunu HR ve IT sutun olacak sekilde bir df yapip, HR’in bulundugu satirlara 1, bulunmadiklarina 0 yazdi. Aynisini IT icin de yapti. Boylece kategorik bir degere donusturmus oldu.

**pd.get\_dummies(df.dept, drop\_first=True)**

#get\_dummies’lerde 2 kategorik data varsa, bunu 1’e cevirmeliyim. 3 varsa 2’ye.

#o yuzden ilk satiri dusuruyoruz.

**df.Skills.str.get\_dummies(sep=",")**

#sonra Skills sutununa geldi, ona get dummies yapti.

#, separatorunu kullanarak da, Python, SQL, Tableu gibi skillsleri ayri sutunlar halinde gelmesini sagladi.

#Orjinal halinde, tabloda yanyana virgulle siralanmis halde bulunuyordu bazilari (Python, Tableau, SQL) gibi

**skills\_dummy = df.Skills.str.get\_dummies(sep=",").add\_prefix("skills\_")**

**skills\_dummy**

#Skills sutununa git, datalarindan get dummies ile yapay kategorik veri olustur.

#aralari virgulle ayrili birden fazla data varsa, onlari ayri sutunlar olarak separate et.

#add\_prefix ile de “skills\_” ismini koy baslarina sutunlarda.

#bunu da skills\_dummy gibi yeni bir degiskene atayarak kalici hale getir.

**df\_final=df[["dept", "job", "sales", "Skills"]]**

#df\_final ile birlestirmemiz gerekecek simdi skill\_dummy'i:

**df\_final=df\_final.join(skills\_dummy)**

#Skills sutunun get dummies ile kategorik veriye donusturdugum icin

#artik onu droplamam gerekecek:

**df\_final.drop("Skills", axis=1, inplace=True)**

#Skills’in ilk sutununu drop first True ile dusurmus olduk.

#df\_final ile skills\_dummy’i join ile birlestirmis olduk.

#df\_final olarak yeni bir degiskene atayarak da kalici hale getirdik

#df\_final'i get dummies yapiyoruz simdi:

**pd.get\_dummies(df\_final, drop\_first=True)**

#sales sutunu sayisal oldugu icin onu kategorik data haline getiremedi.

**IMPORT AND EXPORT DATAFRAME**

**Simdi bu uzerinde calistigim temizleyip kategorik hale getirdigim df\_final df’ini csv dosyasi olarak kaydediyorum. Export:**

**df\_final.to\_csv("df\_final.csv")**

#burdaki df'yi export ederek, csv olarak kaydettim.

#daha sonra buna kolayca ulasabilecegim

**df\_new=pd.read\_csv("df\_final.csv")**

**df\_new.head()**

#read ile okuttum ve ilk 5 satirini cagirttim

#unnamed olarak gelen indexleri (sutun) dan kurtulmam gerek

**df\_new=pd.read\_csv("df\_final.csv", index\_col=0)**

**df\_new.head()**

#bu sekilde ilk index'teki index numaralarini gosteren sutundan kurtulmus oldum

#index\_col'dan sonra hangi index nosunu yazarsam, o sutunu atar. ayni seyi dropla da yapabilirim.

**df\_html=pd.read\_html("https://en.wikipedia.org/wiki/List\_of\_countries\_by\_population\_(United\_Nations)")**

#bu da bir html dosyasinin icindeki datayi okuyup getiriyor.

#ama anlamli uzerinde calisabilecegim bir formatta getirmiyor bu

**table=pd.read\_html("https://en.wikipedia.org/wiki/List\_of\_countries\_by\_population\_(United\_Nations)")**

#bu sekilde table’a atarsam table olarak getirir.

**df\_country=table[0]**

#bu sayfa icindeki ilk tabloyu alir

**#1 yazarsam varsa 2. tabloyu getirir**

**df\_country.head()**

#df\_country tablosunu bu wikipedia adresinden cekmis bir df olusturmus oldum

#head ile de ilk bes satirni getirdim tablonun

**Time Data**

#zaman serilerinde kullanacagiz daha cok

**df=pd.read\_csv("time\_exercise.csv")**

**df.head()**

**df["entry\_date"]=pd.to\_datetime(df["entry\_date"])**

#datetime modulu sadece gunler uzerinde islem yapiyor

#tipi objecti, datetime'a cevirdi tipini

#artik tarihler uzerinde islem yapabilirim.

**df["order\_date"]=pd.to\_datetime(df["order\_date"])**

#order\_date’e de aynisini uyguladik

**df.entry\_date.min()**

#ilk tarihi gostermis oldu. depoya ilk giren

**df.entry\_date.max()**

#depoya son giren urunun tarihini bulabilirsin

**df.entry\_date.max()-df.entry\_date.min()**

#Max ve min tarihlerin farklini aliyor

**df.entry\_date.dt.year**

#dt-sadece yillari gostersin istersem **dt.year** yaparim

#day var, week var farkli kullanimlari var

**df.entry\_date.dt.month**

**Not:#max baslibasina bir method yaninda () kullandik o yuzden**

**#ama month year'i dt ile kullandigimiz icin parantez kullanmadim**