**# Veriye İlk Bakış**

import seaborn as sns

import pandas as pd

planets = sns.load\_dataset("planets")

planets.head()

df = planets.copy()

df.head()

df.tail()

df.dtypes

df.method = pd.Categorical(df.method)

df.dtypes

* Veriye ilk baktığımız zaman ilk olarak hikayesini bilmemiz lazım yani ne nereden geldiği bilinmeli.
* İnceleme başlandığı zaman ilk olarak copy ile kopyasını oluşturmamız gerekli.
* Tail() ve dtypes ile ne içindeki objeler türü bilinmeli işte 3 int ,5 str kaç değişken var vs vs.

df.shape

df.columns

df.describe().T

df.describe(include = "all").T

* Değişken ve gözlem sayısına erişmek için shape kullanılır. (Satır ve sutunlar)
* Sadece değişken isimlerine erişmek istenirse columns kullanılır.
* Betimsel istatistikler için describe kullanılıyordu. T ile transpozu alınıp yanı satır sutuna çevirme olayı. T’nin diğer bir özelliği ise Eksi gözlemler göz ardı edip katagorik değişkenler dışarıda bırakılır.(sayısal)
* İnclude ile tüm değerler göz önüne alınıp hesaplama yapılır.

planets = sns.load\_dataset("planets")

df = planets.copy()

df.isnull().values.any()

df.isnull().sum()

df["orbital\_period"].fillna(0, inplace = True)

df["mass"].fillna(df.mass.mean(), inplace = True)

df.fillna(df.mean(), inplace = True)

* 3. Sayırda data framemizde eksik gözlem(değer) varmı sorgulaması yapıldı. İsnull sorgulama yap , values üzerinde , any herhangi birinide bile varsa true dön
* 4. Satırda hangisinden kaçartane eksik var öğrenilir
* 5. Satır kodu eksik olan değerleri sıfır yapılabilir. İstenirse ortalamada yazılabilir fakat dikkat edilmesi gereken hassas konudur.
* 6. Satırda eksik değerlere ortalama atadık 5. De bütün eksik değerlere ortalama atandı.
* Şuan veri setinin ayarlarıyla oynandı. Orjinali kopyaladığımız için durmaktadır.

planets = sns.load\_dataset("planets")

df = planets.copy()

kat\_df = df.select\_dtypes(include =["object"])

kat\_df.method.unique()

kat\_df["method"].value\_counts().count()

kat\_df["method"].value\_counts()

df["method"].value\_counts().plot.barh();

* 3. Kat\_df diye sadece kategorik değişkenleri içeren dataframe oluştu.
* 4. Kategorik değişkenleri gösterdi.
* 5. Kaç çeşit kategorik değişken sınıfı olduğu öğrenildi.
* 6. Frekansına (Sınıflar kaçar tane olduğuna) baktık.
* 7. Satırda barlar ile gösterdik. “;” konursa sonuna hata göstermez.



planets = sns.load\_dataset("planets")

df = planets.copy()

df\_num = df.select\_dtypes(include = ["float64", "int64"])

df\_num.describe().T

df\_num["distance"].describe()

print("Ortalama: " + str(df\_num["distance"].mean()))

* 3.Satırla sürekli değişkenleri seçtik.
* 5. Betimsel istatistiklerini programa verdik.
* 6. Satırda sadece 1 değişkenin betimsel istatistikler.
* 7. Türkçe yapmak istersek bu şekilde cıktı verebiliriz.

from pandas.api.types import CategoricalDtype

import seaborn as sns

diamonds = sns.load\_dataset('diamonds')

df = diamonds.copy()

df.cut =df.cut.astype(CategoricalDtype(ordered = True))

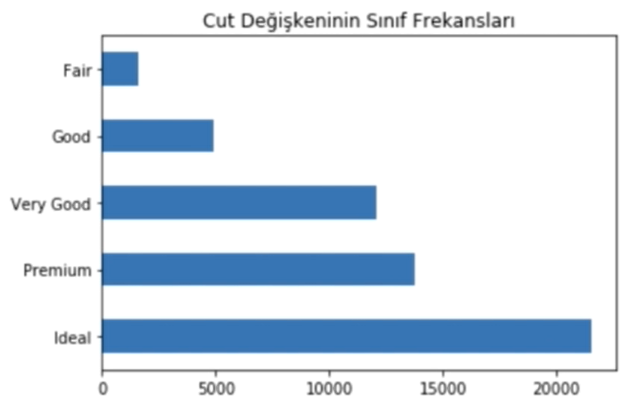
cut\_kategoriler = ["Fair","Good","Very Good","Premium","Ideal"]

df.cut = df.cut.astype(CategoricalDtype( categories = cut\_kategoriler, ordered = True))

* 5.satırda astype işlemi cut’u kategorik olarak dönüştür. Bunu sıralı bir şekilde yap.(ordered)
* 5.satırda sıralama yaparken düzgün bir sıralama yapamadı.Bu yüzden 7. Satırda düzgün sırayı belirleyip 9. Satırda düzgün sıralamayı df\_cut içine attık.(veriyi düzelttik.)

df["cut"].value\_counts().plot.barh().set\_title("Cut Değişkeninin Sınıf Frekansları");

* Bir önceki kodun işlemleri işlemleri grafikleştirdik. Set title grafiğe bilgi eklemek için kullanılır.



(df["cut"]

 .value\_counts()

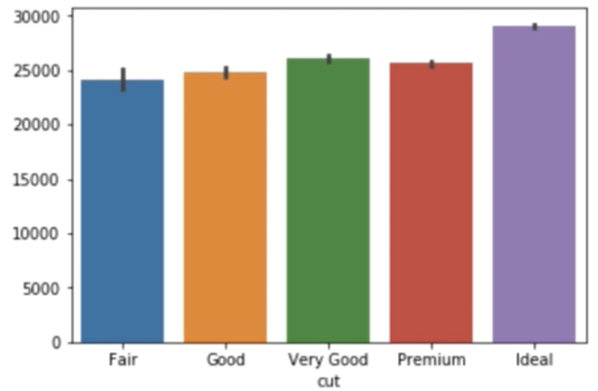
 .plot.barh()

 .set\_title("Cut Değişkeninin Sınıf Frekansları"));

* Yukarıdaki işlemin daha düzenli hali. (.value\_counts()) Neyi görselleştireceğimiz , (.plot.barh()) görselleştirme taktiği , (.set\_title) grafik başlığı gibi.

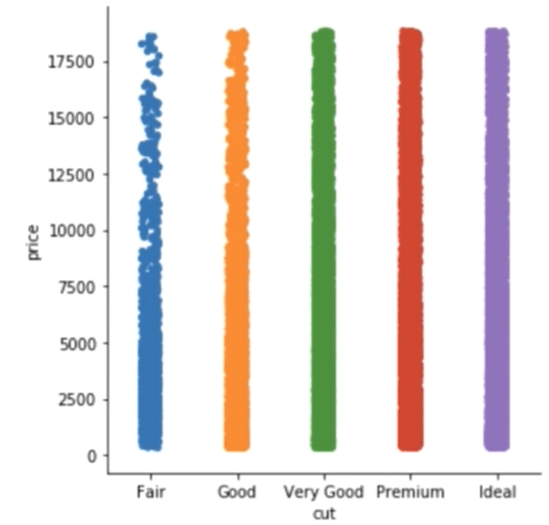
sns.barplot(x = "cut", y = df.cut.index, data= df);

* X eksenime “cut”’u koy , y eksenime cut’un frenkansını koy ve sonuna data argümanı ile data framemizi gösterdik.



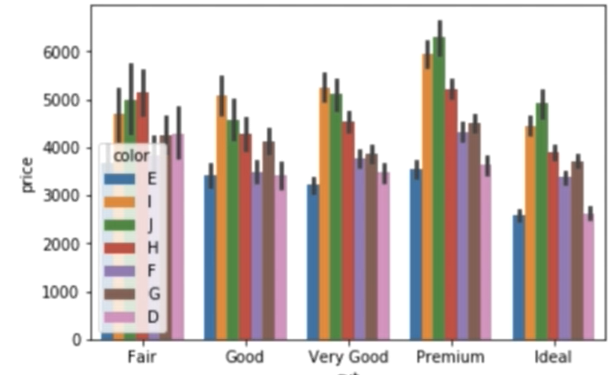
sns.catplot(x = "cut", y = "price", data = df);

* Price ve cut değişkenlerini birlikte ele aldık. Grafik incelenecek olursa fair kısmında dağılım yoğunluğu aşağıda daha fazla bulunuyor.



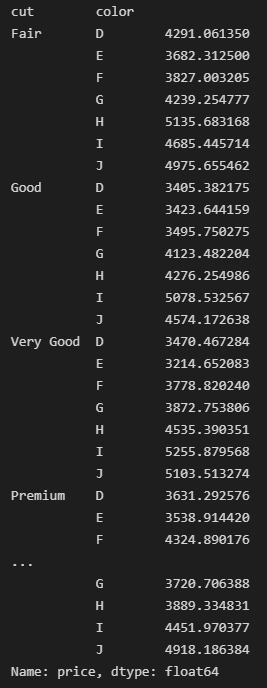
sns.barplot(x = "cut", y = "price", hue = "color", data = df);

* Bir önceki catplotta oluşan grafiğin dağılımını incelemiştik. Şimdi bunu neye göre dağıldığını hue=color yazarak 3. Değişkenide işin içine dahil edip daha detaylı gözlem yapabiliriz.
* Grafikteki cubuklar standart sapmayı temsil eder.



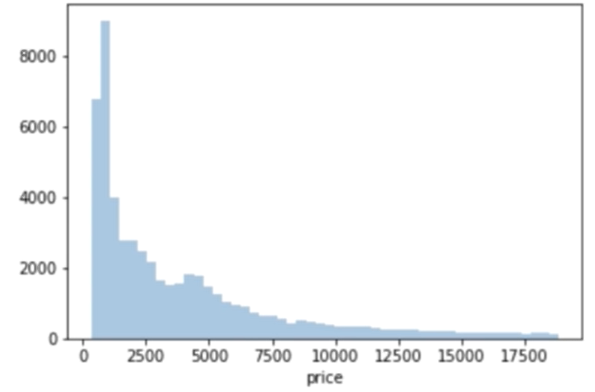
df.groupby(["cut","color"])["price"].mean()

* İle doğrulama işlemi yapabiliriz bu grafiğe göre. Bakıldığında mesela good da E ‘nin değeri 3500 gibi bir şey olması lazım. Aşağıda 3423 olduğu görülebilir..



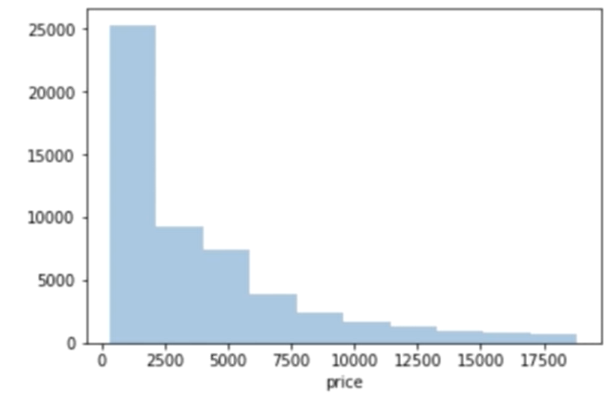
sns.distplot(df.price, kde = False);

* Histogram sayısal değerler için kullanılıp dağılımını ifade eder.
* Distplot dağılım göstermek için kullanılan fonksiyon
* İlk olarak değişken , kde ise yoğunluk üzerine konup konmaması ile ilgili. İleride True yapacağız.
* Histogramın diğerleri ile farkı , belli aralıklara bölüp görselleştirir.



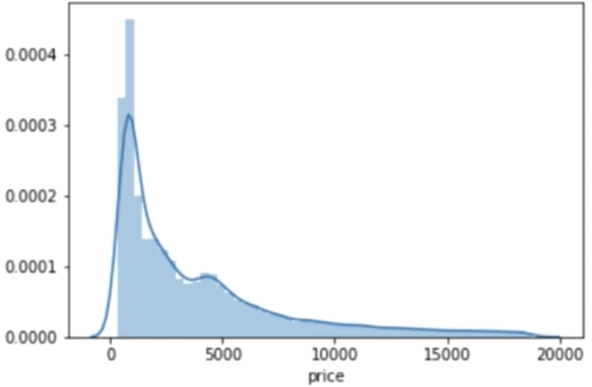
sns.distplot(df.price, bins = 10, kde = False);

* Bins = çubuk sayısını ifade ediyor.



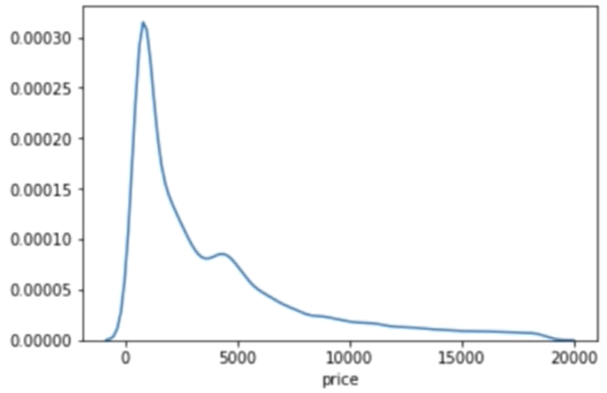
sns.distplot(df.price);

* Çubukla yoğunluğu gösterdik. Girilmezse kde=True değeri dönmüş olur.



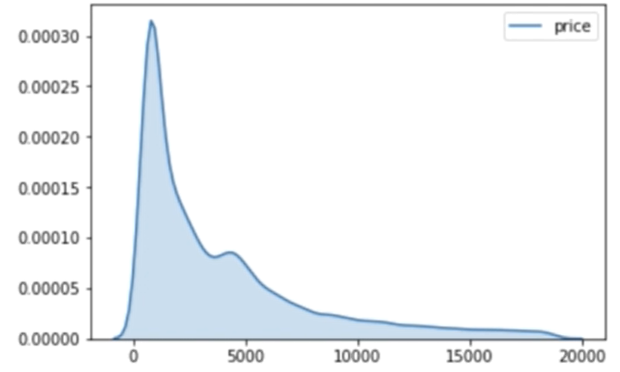
sns.distplot(df.price, hist = False);

* Histogramı false yapıp sadece dağılım grafiği gösterilebiliyor.



sns.kdeplot(df.price, shade = True);

* Alt kısmı boyamak için girilebilir.



(sns

 .FacetGrid(df,

              hue = "cut",

              height = 5,

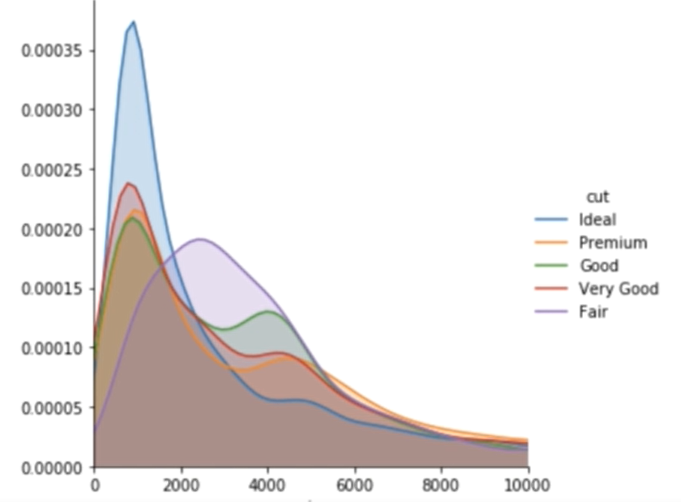
              xlim = (0, 10000))

 .map(sns.kdeplot, "price", shade= True)

 .add\_legend()

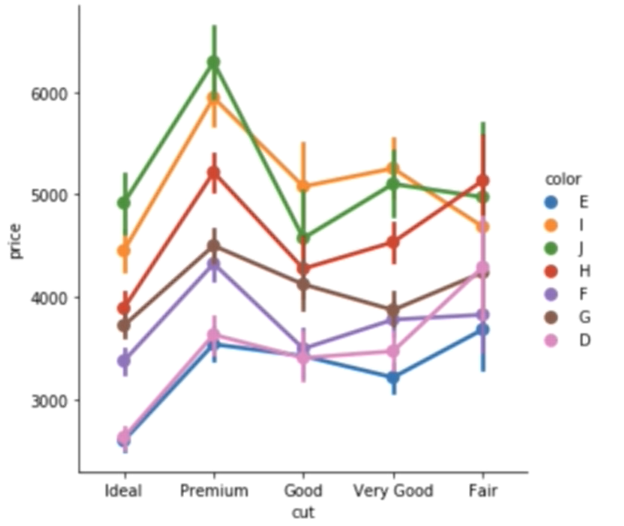
);

* Sns.FacetGrid parçalı bölmek için kullanılan fonksiyon.
* Hue= boyut ekleme argümanı.
* Height yükseklik için girildi.
* xlim argümanı ile x ekseninin boyutlarını ayarladık.(belirli alan)
* .map(sns.kdeplot) ile bir yoğunluk grafiği üzerine boyut eklenir
* Add\_legend() fonksiyonuda bilgi eklemek için.



sns.catplot(x = "cut", y = "price", hue = "color", kind = "point", data = df);

* X için cut, y için price ve ekstra color olmasanı da istedim.
* Kind tür için kullanılır. Point olduğu için noktalarla ifade edlir.



* Çubuklar sapmaları gösterirken noktalar ortalamayı gösteriyor.

BoxPlot

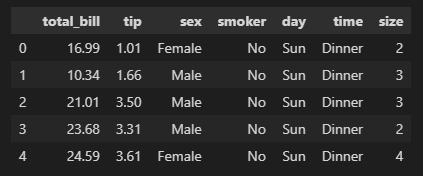
import seaborn as sns

tips = sns.load\_dataset("tips")

df = tips.copy()

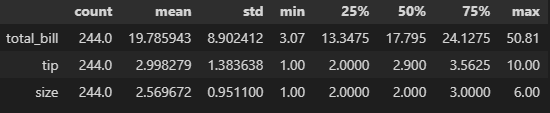
df.head()

* Tips veri setimizi tanımlayıp kopyaladık. İlk 5 e bakıyoruz.

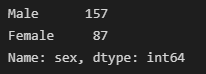


* Total\_bill : yemeğin toplam fiyat (fiyat+bahşiş)
* Tip : Bahşiş
* Sex : Ücreti ödeyenin cinsiyeti
* Smoker : Sigara içen var mı (0-no, 1-yes)
* Day : Günler(1 pazartesi 2 Salı 3 Çarşamba …)
* Time : Ne zaman (0: sabah 1: Akşam)
* Size: Gruptaki kişi sayıları

df.describe().T

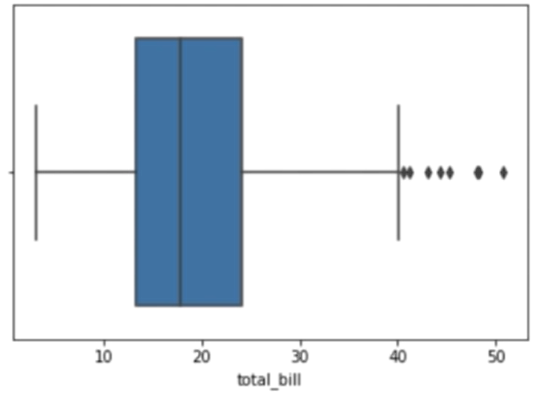


df["sex"].value\_counts()



* Kategorik değişkenlere ulaştık. 157 erkek,87 kadın.

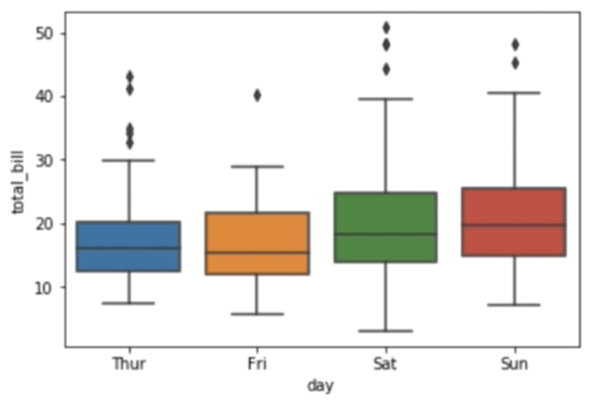
sns.boxplot(x = df["total\_bill"]);



* Grafikte en soldaki çubuk en düşük değer ,en soldaki nokta en yüksek değer. Ortadaki 3 çubuğun en soldaki %25. Değer , ortadaki %50. Değer(median) , en soldaki ise %75. Değerdir.
* Soldaki noktalar aykırı kısımları belirtir. İlerleyen bölümde anlatılcak.
* Dikey içinde sns.boxplot(x = df["total\_bill"], Orient = “v”); kodu kullanılabilir.

Hangi gunler daha fazla kazanıyoruz?

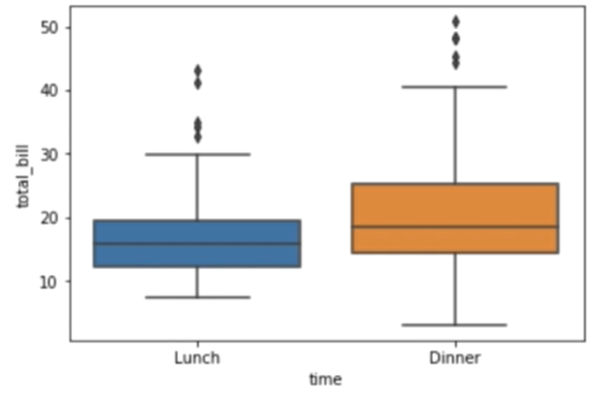
sns.boxplot(x = "day", y = "total\_bill", data = df);



* Grafiğe bakıldığında cumartesi daha çok müşteri gelse de en yüksek ciro Pazar günü yapılmıştır .

Sabah mı akşam mı daha çok kazanıyoruz?

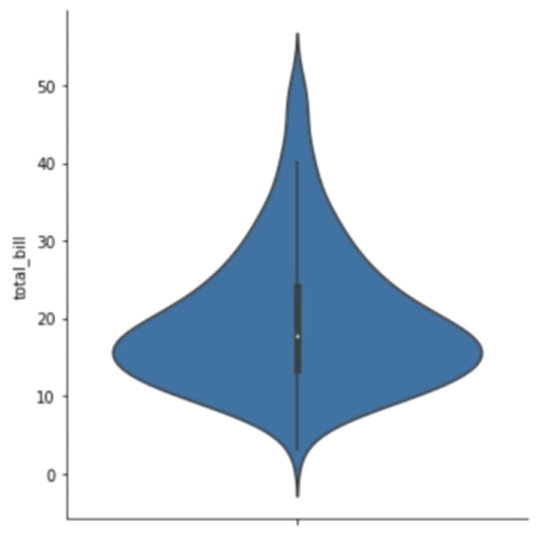
sns.boxplot(x = "time", y = "total\_bill", data = df);



* Grafiğe göre akşamları gelen müşteri sayısı hem daha fazla hem ciro daha yüksektir.

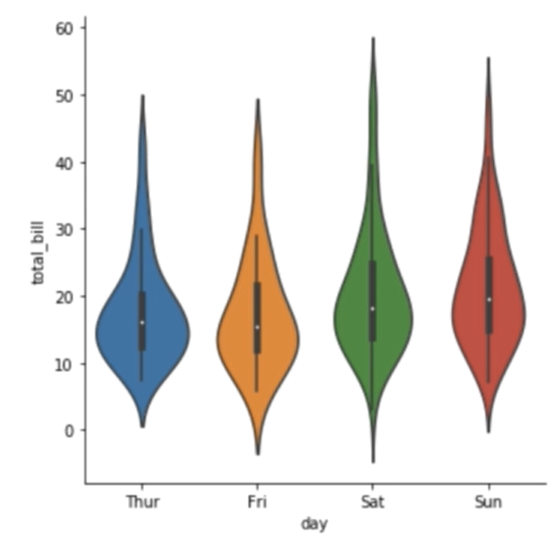
Violin

sns.catplot(y = "total\_bill", kind = "violin", data = df);



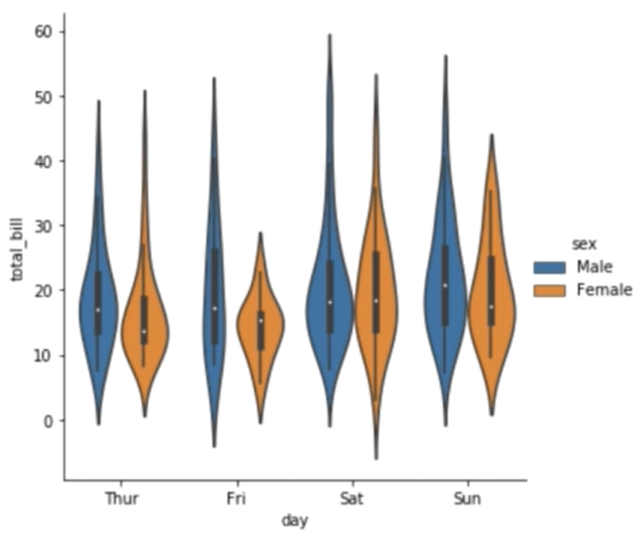
* Violin grafik daha cok tercih meselesidir. Bu grafik merkezi eğilimi sunan ve ortasında boxplot benzeri bir grafik bulunur.

sns.catplot(x= "day", y = "total\_bill", kind = "violin", data = df);



* Kırılım,çapralama yeni boyut eklemedir.

sns.catplot(x= "day", y = "total\_bill", hue = "sex",kind = "violin", data = df);



* Cinsiyet ekledik.

Korelasyon Grafikleri

* Korelasyon değişkenler arasındaki ilişkiyi temsil eden istatistiksel bir terimdir.

**Scatterplot(Saçılım Grafiği):** İki değişken arası ilişkiyi ifade etmek için kullanılan ve en çok bilinen yaklaşımdır. Sayısal değişkenler arası ilişkiyi gösterir.

* Kategorik değişkenler -> Sütun Grafik
* Sayısal Değişkenler -> Histogram , Yogunluk , Boxplot Ve Violin
* Artık tek değişkenli veri görselleştirmeden 2 değişkenli veri görselleştirmeye geçmiş bulunmaktayız.
* Önceki tips verilerinden devam edeceğiz.

**total\_bill: yemeğin toplam fiyatı (bahşiş ve vergi dahil)**

**tip: bahşiş**

**sex: ücreti ödeyen kişinin cinsiyeti (0=male, 1=female)**

**smoker: grupta sigara içen var mı? (0=No, 1=Yes)**

**day: gün (3=Thur, 4=Fri, 5=Sat, 6=Sun)**

**time: ne zaman? (0=Day, 1=Night)**

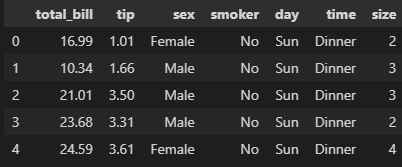
**size: grupta kaç kişi var?**

import seaborn as sns

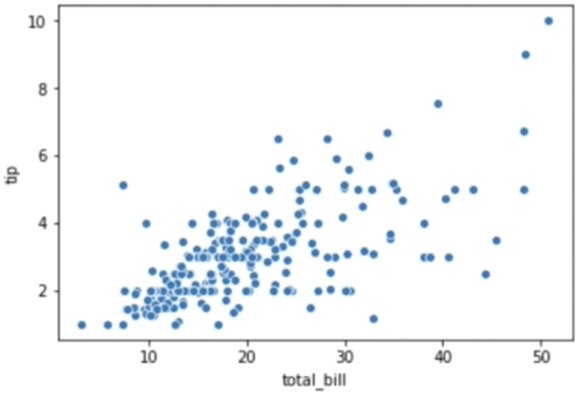
tips = sns.load\_dataset("tips")

df = tips.copy()

df.head()

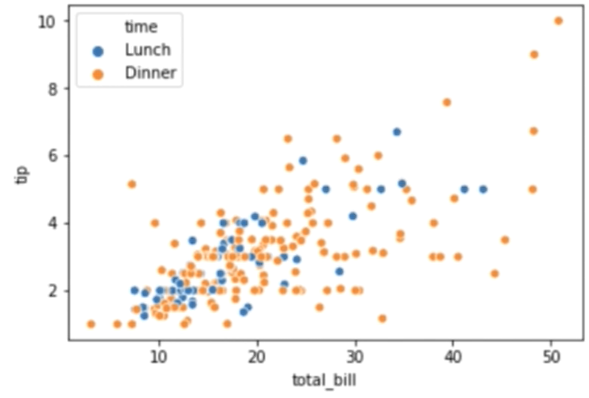


sns.scatterplot(x = "total\_bill", y = "tip", data = df);



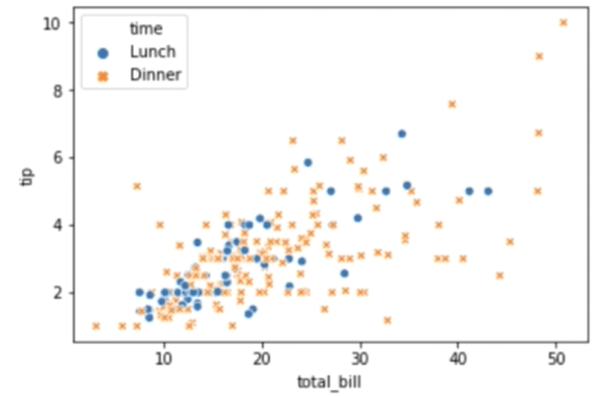
* Bu grafik ile genel olarak ödenen miktar arttıkça bahşiş(tip) arttığı gözlemlenebilir.

sns.scatterplot(x = "total\_bill", y = "tip", hue = "time",data = df);



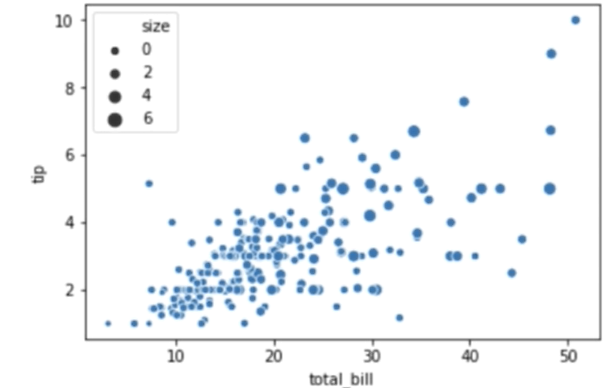
* Time değişkeni eklendiğinde akşam yemeklerinin fiyatı arttıkça bahşişlerin artmış olduğunu görüyoruz.

sns.scatterplot(x = "total\_bill", y = "tip", hue = "time", style = "time", data = df);



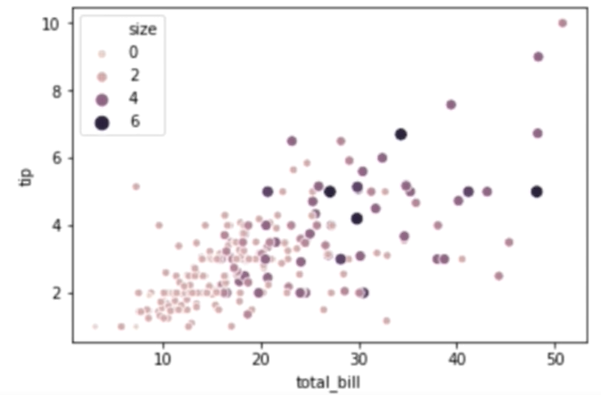
* Akşamları çarpı sembolü ile gösterdik.(Style kategorik için)

sns.scatterplot(x = "total\_bill", y = "tip", style = "day", data = df);



* Sayısal bir değer için size değişkeni girilir.

sns.scatterplot(x = "total\_bill", y = "tip", hue= "size", size = "size", data = df);



* Hue ile kullandığımızdada renklendirme işlemi oldu.

**Doğrusal İlişkinin Gösterilmesi :** İki değişken arasındaki ilişkiyi scatterplot gibi dağılımla değil de doğrusal bir grafikle gösterim şeklidir.

import seaborn as sns

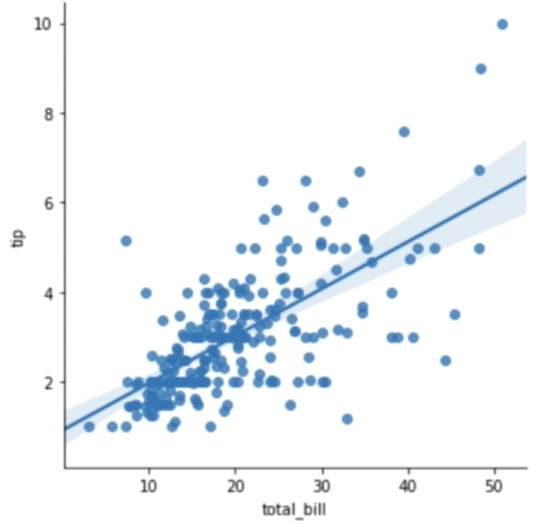
import matplotlib.pyplot as plt

tips = sns.load\_dataset("tips")

df = tips.copy()

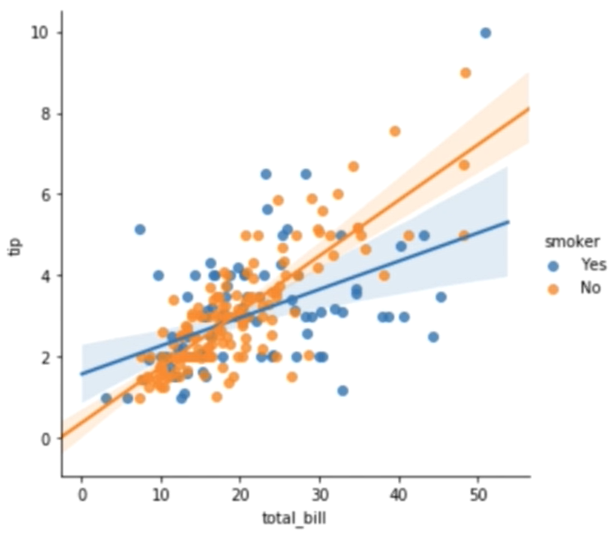
df.head()

sns.lmplot(x = "total\_bill", y = "tip", data = df);



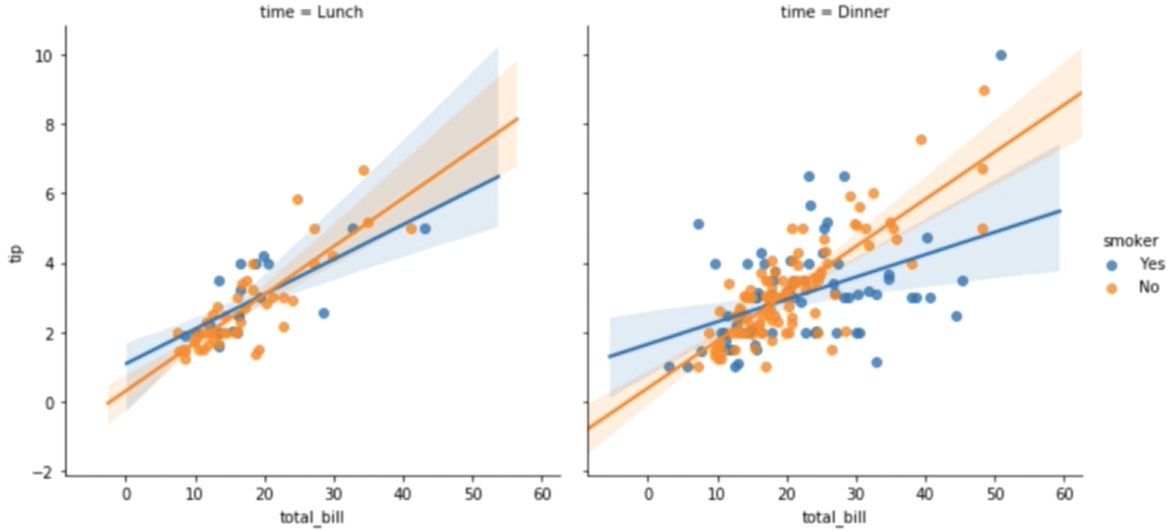
* Hafif açık olan maviler sapmaları gösteriyor. Sondaki kısımda bir anda tipin 10 olması mesela.

sns.lmplot(x = "total\_bill", y = "tip", hue = "smoker", data = df);



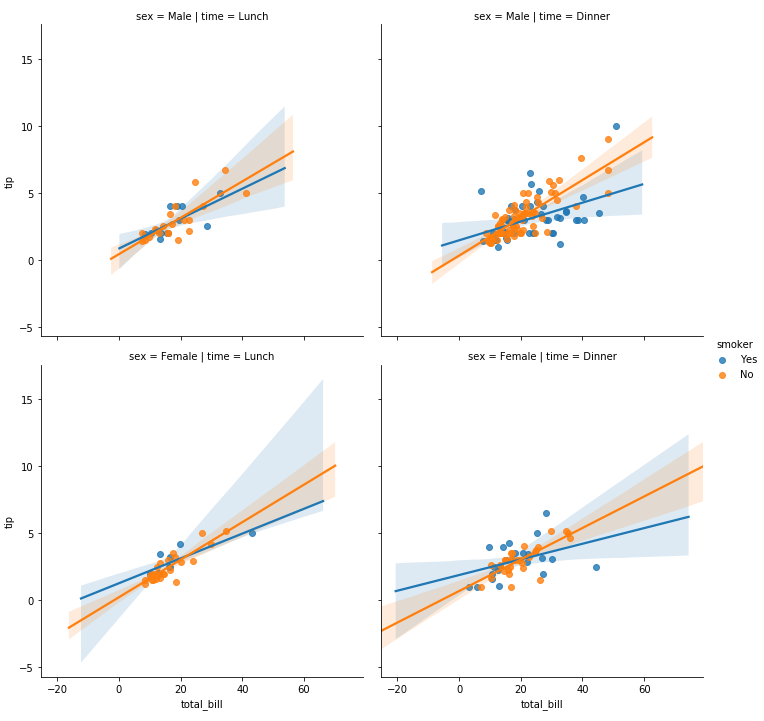
* Ödenen miktar arttıkça bahşişlerde şiddetli bir şekilde artmıştır diyebiliriz lakin sigara içmeyenler için geçerli bu.
* Sigara içenlerde eğim düştüğünü ve içmeyenlere kıyasla sert bir yükseliş yok.

sns.lmplot(x = "total\_bill", y = "tip", hue = "smoker", col = "time", data = df);



* Col argümanı ile sabah-akşam yemek yiyip sigara içip içmeyenlere göre ayırdık.(Sabah aksam ayrımı)

sns.lmplot(x = "total\_bill", y = "tip", hue = "smoker", col = "time", row = "sex", data = df);



* Row ile bir değişken daha ekleyip cinsiyet üzerinde de inceleme yaptık.

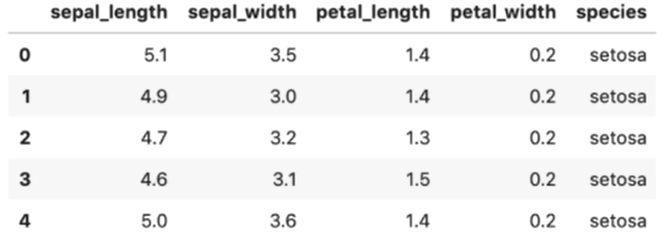
**Scatterplot Matrisi :** Veri seti içerindeki tüm sayısal değerler için matris formunda bu ilişkiyi ifade etmede kullanılır.

import seaborn as sns;

iris = sns.load\_dataset("iris")

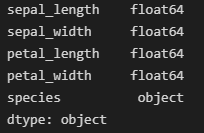
df = iris.copy()

df.head()



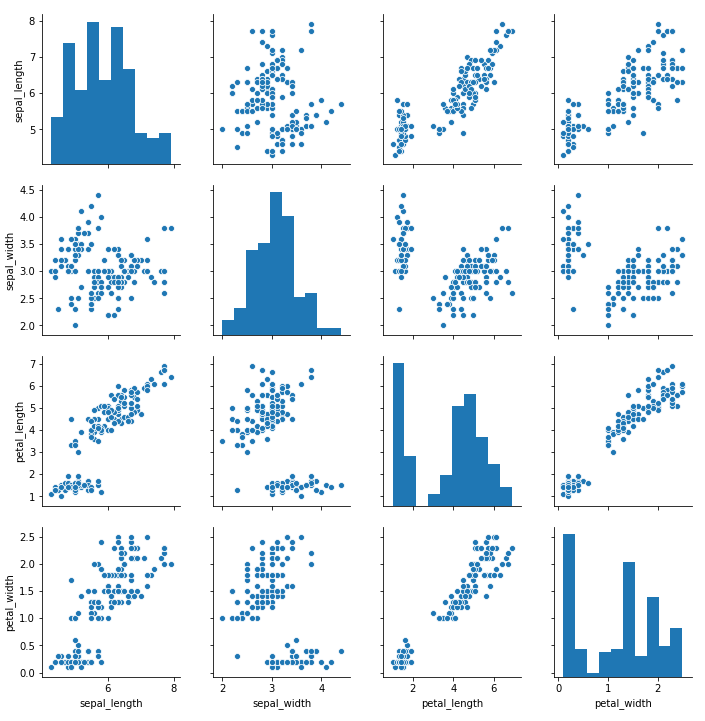
* Veri setinin hikayesi bazı çiçek türleri var ve bu türlerin özelliklerini ifade eden değişkenler var.

df.dtypes; df.shape



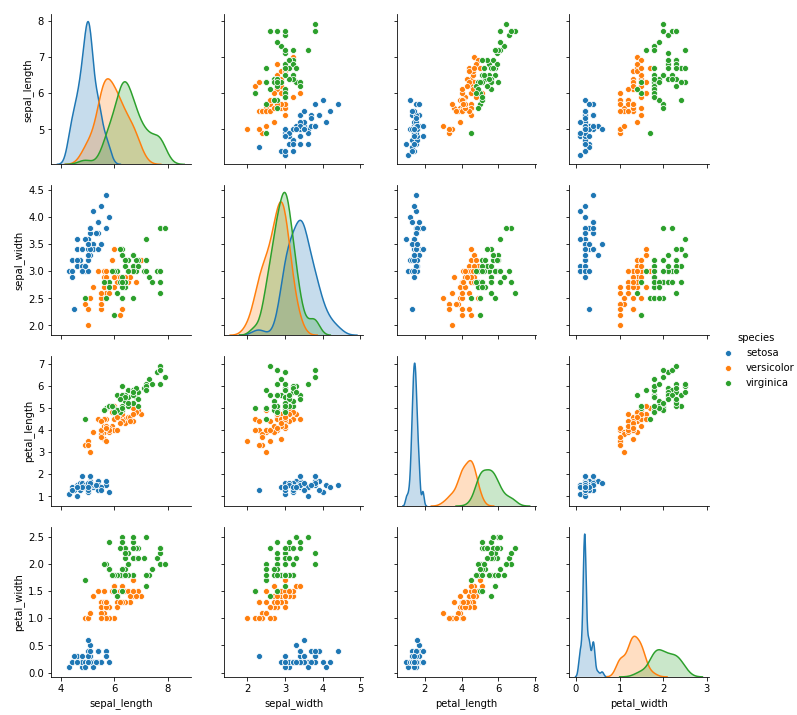
* 1 tane kategorik , 4 sayısal değişkenimiz var ve boyut 150-5.

sns.pairplot(df);

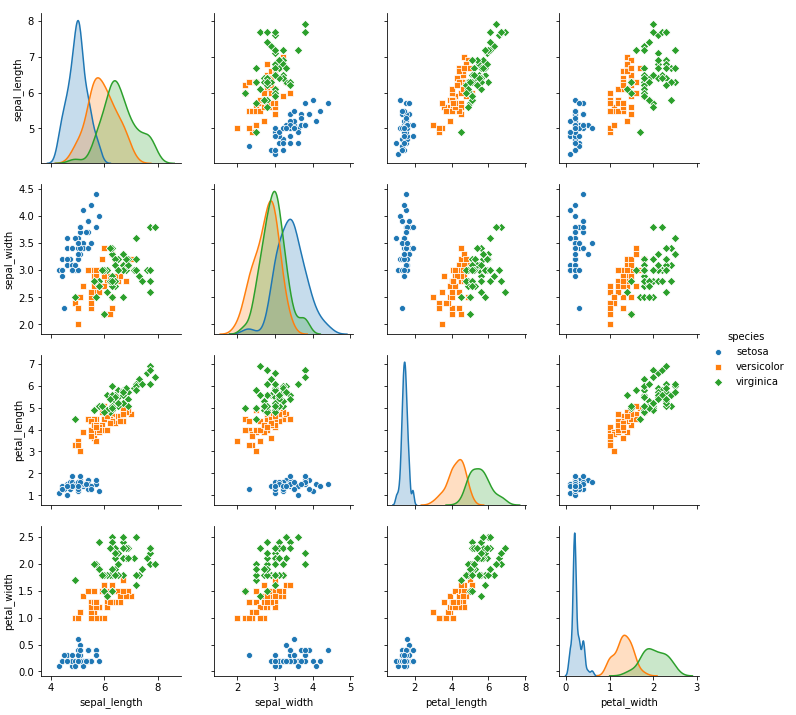


* Toz bulutu şeklinde veya hiç grafik yoksa bu iki değişken arasında ilişki yoktur demektir.
* Çeşitli noktalarda bir kümelenme varsa bu muhtemelen farklı grupları temsil eder.

sns.pairplot(df, hue = "species");

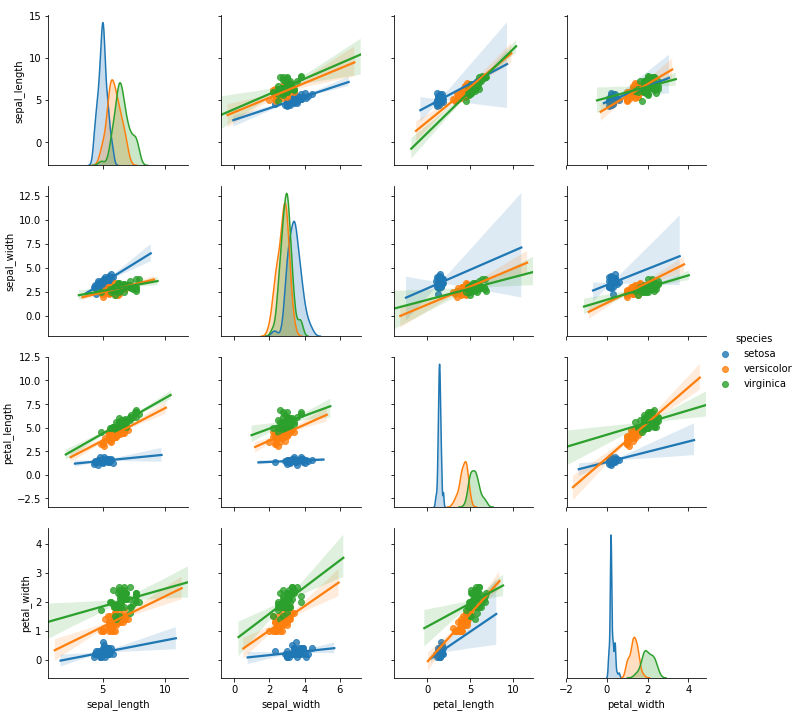


sns.pairplot(df, hue = "species", markers = ["o","s","D"]);



* Markers ile şekil eklemesi yaptık.

sns.pairplot(df, kind = "reg", hue = "species");



* Kind ile doğru ekleme işlemi gerçekleştirdik.(Eğimler)

**Heatmap(Isı Haritası):** Yapısal anlamda daha geniş şekilde görmemizi sağlar. Uzun vadeli verilerde(zaman, yıl ay), bu dönemlere karşılık sayısal değişkenler olduğunda yada daha büyük ölçeklerle tekrar eden durumlar olursa kullanılır.

import seaborn as sns

flights = sns.load\_dataset('flights')

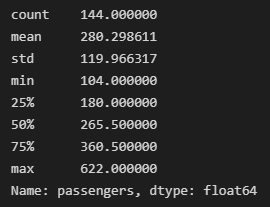
df = flights.copy()

df.head() ; df.shape

* Yıllara ve aylara göre yolculuk sayıları değişmektedir.

df["passengers"].describe()



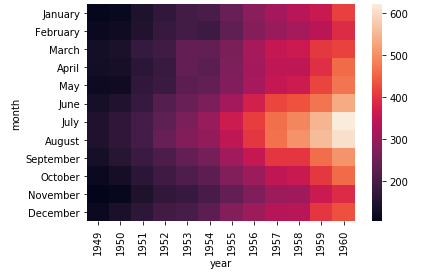
df = df.pivot("month", "year", "passengers");

df

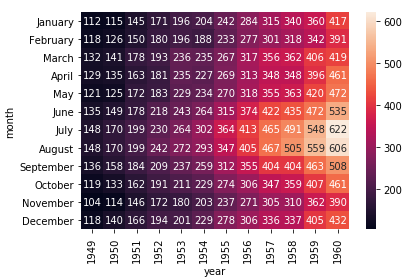


* Isı haritası için daha okunaklı hale getir.
* Pivotlar x eksenii month y ekseni year ve değişkenler passengers yaptık.

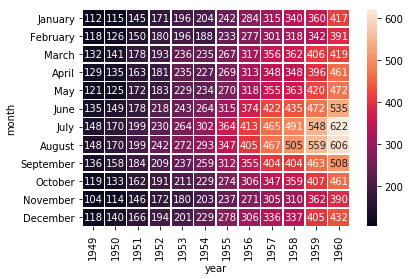
sns.heatmap(df);



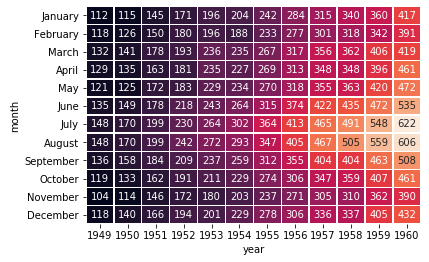
sns.heatmap(df, annot = True, fmt = "d");



sns.heatmap(df, annot = True, fmt = "d", linewidths = .5);



sns.heatmap(df, annot = True, fmt = "d", linewidths = .5, cbar = False);



* cbar argümanı ile yandaki sıcaklık değerlerini kaldırdık.
* linewidths argümanı ile kutucuk arası boşluk koyduk.
* annot = True, fmt = "d" argümanlar ile kutucuklar arası değerler girildi.
* Grafiğe baktığımızda artan yıllara göre yolcu sayısı artmış. Genel olarak June,July,August aylarında kullanım diğer aylara göre daha fazla olduğu gözlemlenebilir.

**Çizgi Grafik:** Makinelerin urettiği veriler , daha zor problemlerde kullanılacak , zamana bağlı.

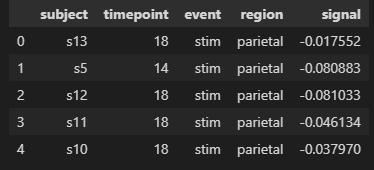
**Veri seti hikayesi:**

import seaborn as sns

fmri = sns.load\_dataset("fmri")

df = fmri.copy()

df.head()

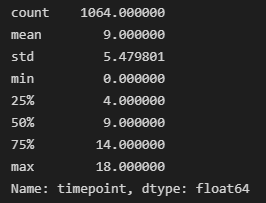


* Hikaye: Beyne bağlanan cihazla toplanan sinyalleri ifade eder.
* Subject : Verilerin toplandığı kişileri ifade eder.
* Timepoint : zaman noktalarını ifade ediyor.
* Event : birbirinden farklı olaylar
* Region : Sinyalin toplandığı bölge ifade ediliyor.
* Signal : Gelen sinyalleri ifade eder.

df.shape

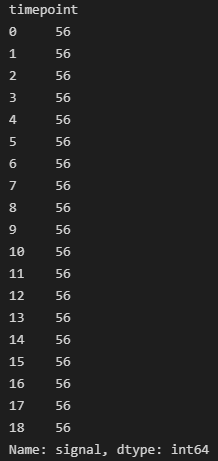


df["timepoint"].describe()



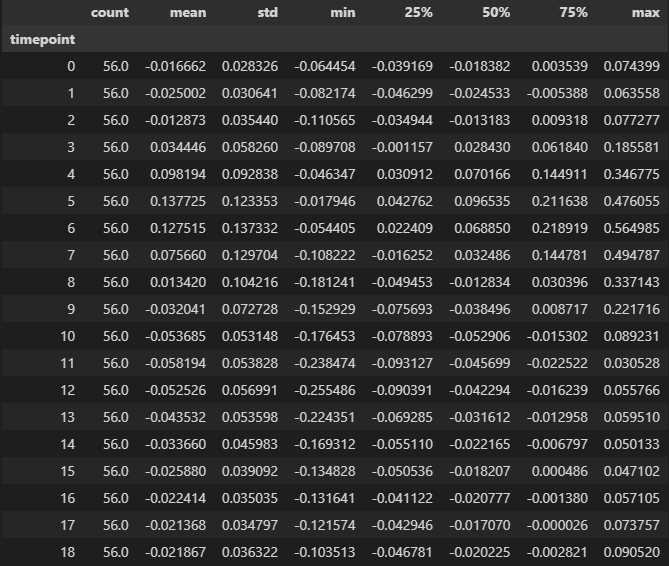
* Değerler kesikli olduğu düşünülür , hepsi tam sayı.
* İstenirse kategorik değişken olarakda değerlenebilir.

df.groupby("timepoint")["signal"].count()



* Anlaşılacağı üzere 18 zaman diliminde 56 tane sinyal ölçülmüş.

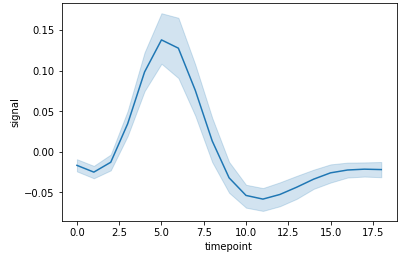
df.groupby("timepoint")["signal"].describe()



* Zaman noktalarının betimsel istatistikleri.

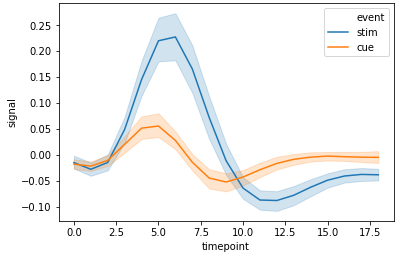
**Çizgi Grafik(Lineplot()):**

sns.lineplot(x = "timepoint", y = "signal", data = df);



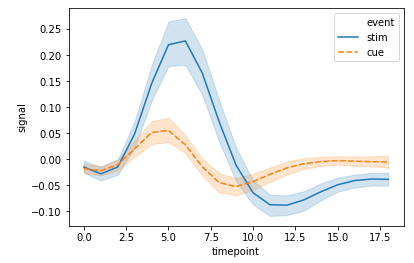
* Mavi çizgiler ortalama, acık maviler sapma.

sns.lineplot(x = "timepoint", y = "signal", hue = "event", data = df);



* 5.0 daki sıçrama stimden kaynaklanıyormuş.

sns.lineplot(x = "timepoint", y = "signal", hue = "event", style = "event", data = df);



* Style ile kesik çizgili yaptık.

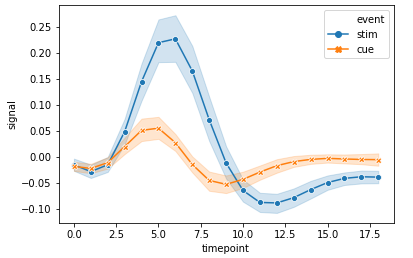
sns.lineplot(x = "timepoint",

             y = "signal",

             hue = "event",

             style = "event",

             markers = True,  dashes = False, data = df);



* Ortalama değerler işaretlendi.

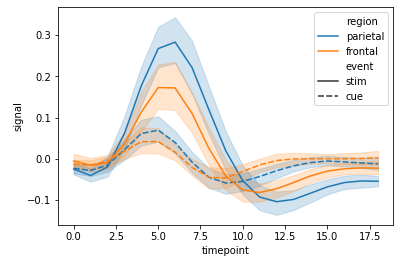
sns.lineplot(x = "timepoint",

             y = "signal",

             hue = "region",

             style = "event",

             data = df);



* Bir boyut daha ekledik hue değiştirerek..

**Basit Zaman Serisi Grafiği:** Apple den hisse verilerini çekip işleyeceğiz.

!pip install pandas\_datareader

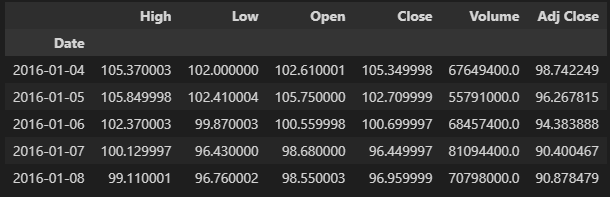
import pandas\_datareader as pr

* Pandas\_datareader çekip kullandık

df = pr.get\_data\_yahoo("AAPL", start = "2016-01-01", end = "2019-08-25")

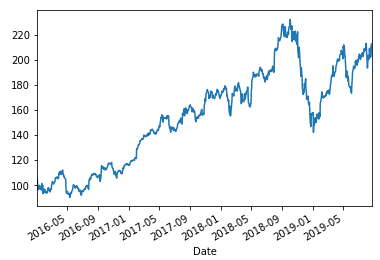
* Applenin hisse verilerini 2016 nın 1. Ayının 1.gününden 2019.8.ay 25 güne kadar al.

df.head()



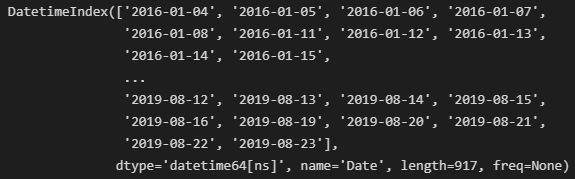
kapanis = df["Close"]

kapanis.plot();



* 4’er ay artarak gitmiş.
* Eksenlerin bazen tarih değişkeni mesela tarih değişkeni olduğunu belirtmemiz lazım.

kapanis.index



kapanis.index =pd.DatetimeIndex(kapanis.index)

* İle bu düzenleme yapılabilir. Düzenli olduğu için tekrar yapmış olduk değişen bir şey olmadı.

Bölüm Sonu…