```
# 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
```

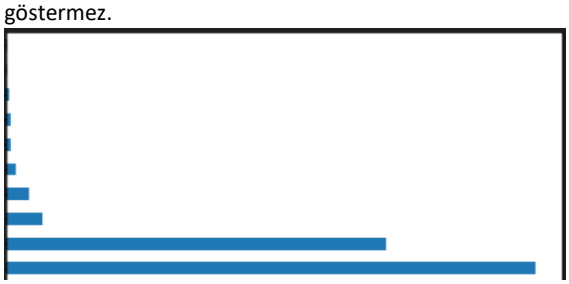
- Değişken ve gözlem sayısına erişmek için shape kullanılır. (Satır ve sutunlar)
- Sadece değişken isimlerine erişmek istenirse columns kullanılır.
- Betimsel istatistikler için describe kullanılıyordu. T ile transpozu alınıp yanı satır sutuna çevirme olayı. T'nin diğer bir özelliği ise Eksi gözlemler 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()))
```

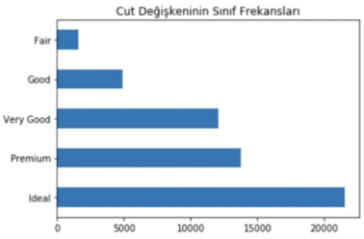
- 3.Satırla sürekli değişkenleri seçtik.
- 5. Betimsel istatistiklerini programa verdik.
- 6. Satırda sadece 1 değişkenin betimsel istatistikler.
- 7. Türkçe yapmak istersek bu şekilde cıktı verebiliriz.

```
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.

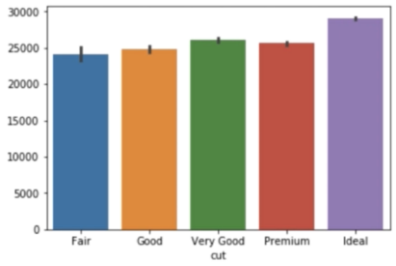


```
(df["cut"]
 .value_counts()
 .plot.barh()
 .set_title("Cut Değişkeninin Sınıf
Frekansları"));
```

- Yukarıdaki işlemin daha düzenli hali. (.value_counts()) Neyi görselleştireceğimiz , (.plot.barh()) görselleştirme taktiği , (.set_title) grafik başlığı gibi.

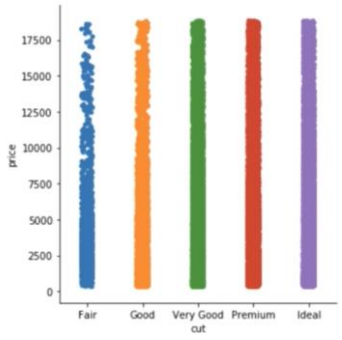
```
sns.barplot(x = "cut", y = df.cut.index, data=
df);
```

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



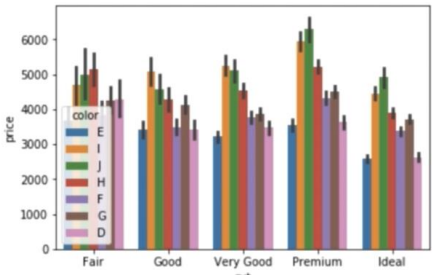
```
sns.catplot(x = "cut", y = "price", data =
df);
```

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



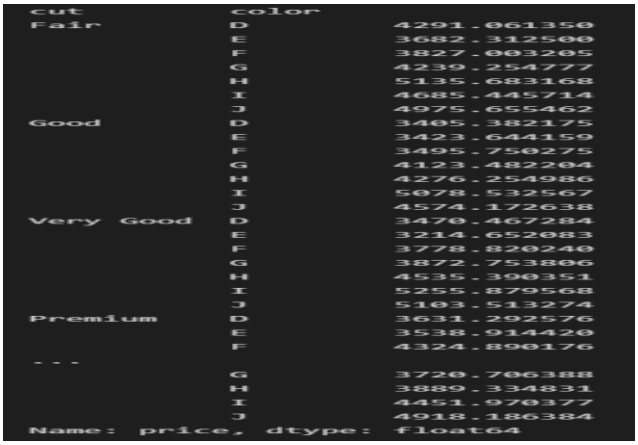

```
sns.barplot(x = "cut", y = "price", hue = "color",
data = df);
```

- 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.



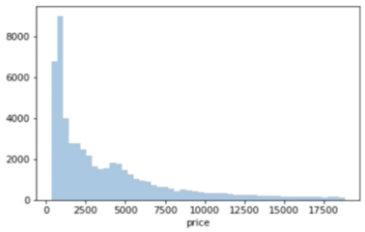

```
df.groupby(["cut","color"])["price"].mean()
```

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



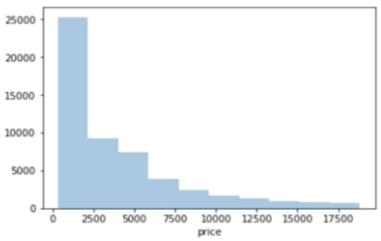
```
sns.distplot(df.price, kde = False);
```

- 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.



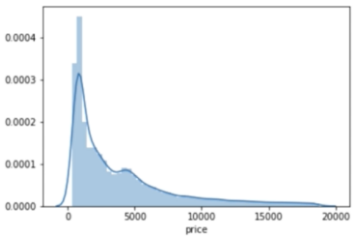

```
sns.distplot(df.price, bins = 10, kde = False);
```

- Bins = çubuk sayısını ifade ediyor.



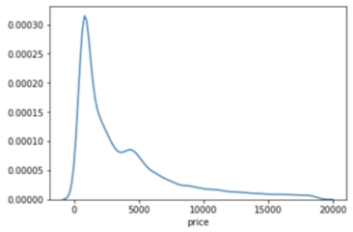
```
sns.distplot(df.price);
```

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



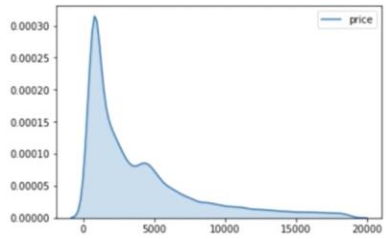

```
sns.distplot(df.price, hist = False);
```

- Histogramı false yapıp sadece dağılım grafiği gösterilebiliyor.



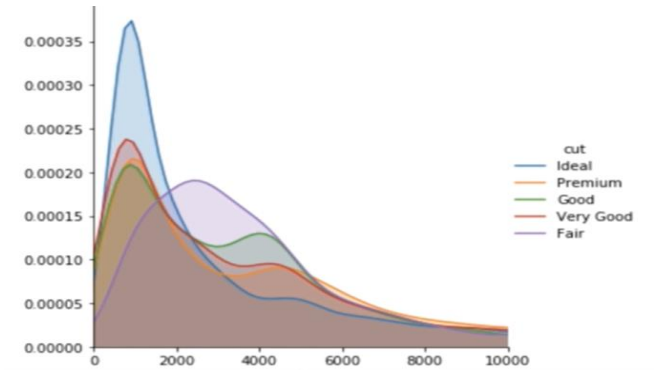

```
sns.kdeplot(df.price, shade = True);
```

- Alt kısmı boyamak için girilebilir.



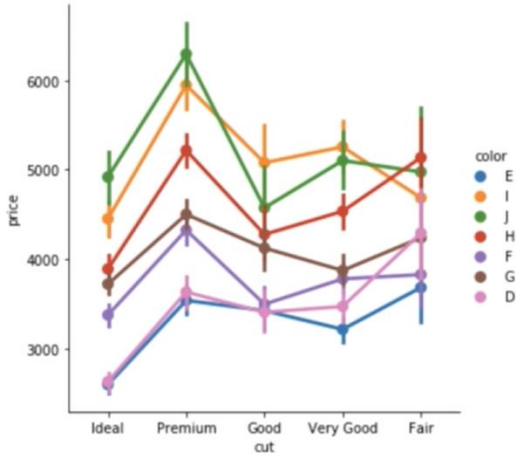
```
(sns
 .FacetGrid(df,
            hue = "cut",
            height = 5,
            xlim = (0, 10000))
 .map(sns.kdeplot, "price", shade= True)
 .add_legend()
);
```

- 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	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

- 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
```

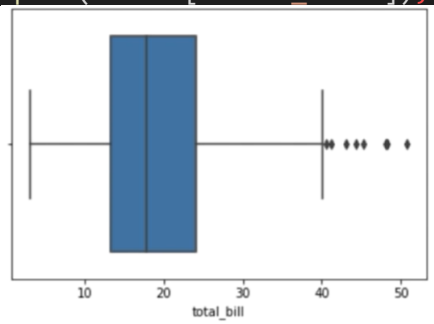
	count	mean	std	min	25%	50%	75%	max
total_bill	244.0	19.785943	8.902412	3.07	13.3475	17.795	24.1275	50.81
tip	244.0	2.998279	1.383638	1.00	2.0000	2.900	3.5625	10.00
size	244.0	2.569672	0.951100	1.00	2.0000	2.000	3.0000	6.00

```
df["sex"].value_counts()
```

```
Male      157
Female     87
Name: sex, dtype: int64
```

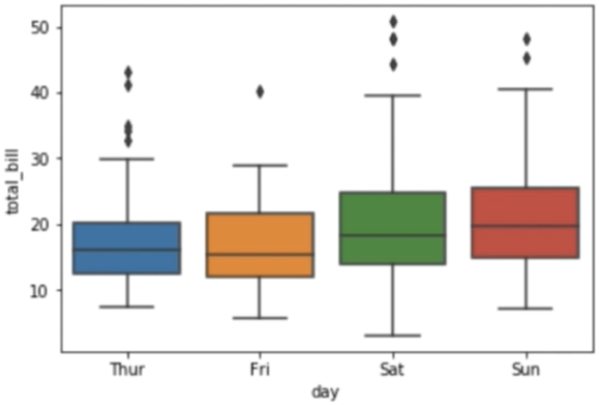
- Kategorik değişkenlere ulaştık. 157 erkek,87 kadın.

```
sns.boxplot(x = df["total_bill"]);
```



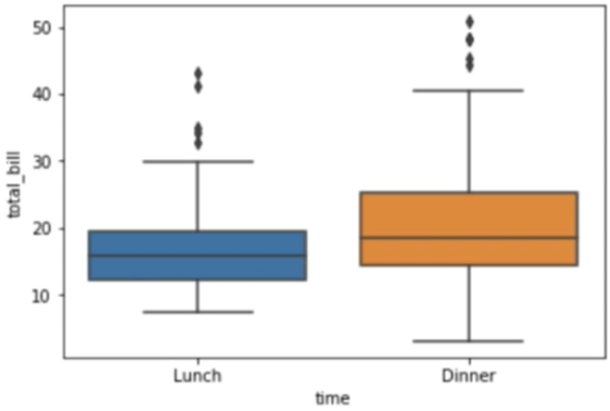
- Grafikte en soldaki çubuk en düşük değer ,en soldaki nokta en yüksek değer. Ortadaki 3 çubuğun en soldaki %25. Değer , ortadaki %50. Değer(median) , en soldaki ise %75. Değerdir.
- Soldaki noktalar aykırı kısımları belirtir. İlerleyen bölümde anlatılcak.
- Dikey içinde sns.boxplot(x = df["total_bill"], 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 .

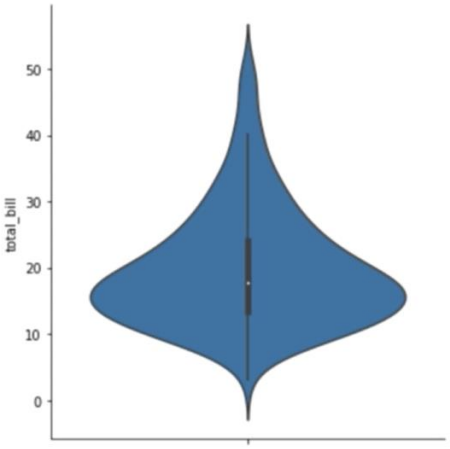
```
Sabah mı akşam mı daha çok kazanıyoruz?
sns.boxplot(x = "time", y = "total_bill", data
= df);
```



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

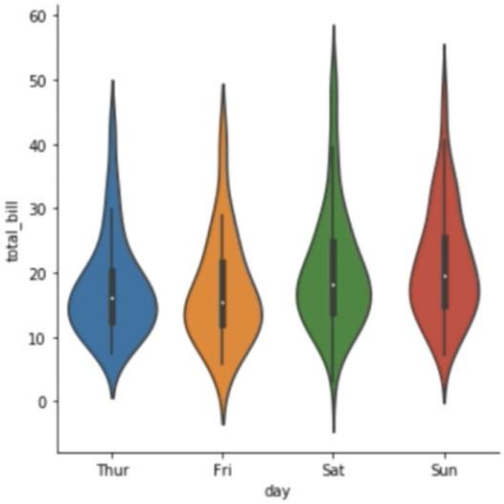
Violin

```
sns.catplot(y = "total_bill", kind = "violin",
data = df);
```



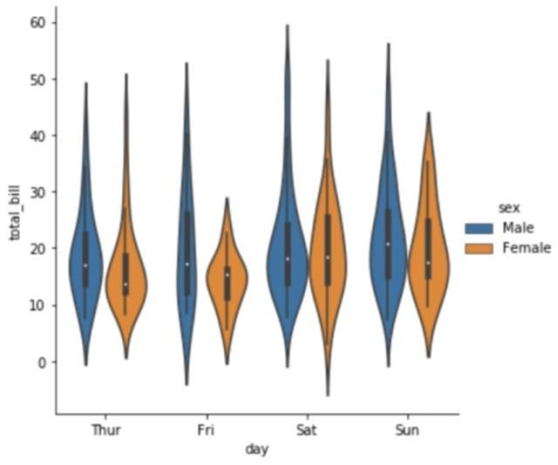
- Violin grafik daha cok tercih meselesidir. Bu grafik merkezi eğilimi sunan ve ortasında boxplot benzeri bir grafik bulunur.

```
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)

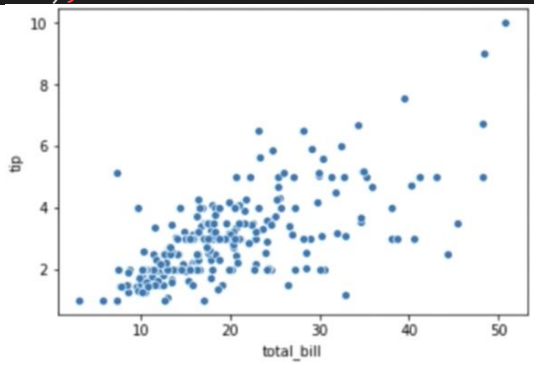
time: ne zaman? (0=Day, 1=Night)

size: grupta kaç kişi var?

```
import seaborn as sns
tips = sns.load_dataset("tips")
df = tips.copy()
df.head()
```

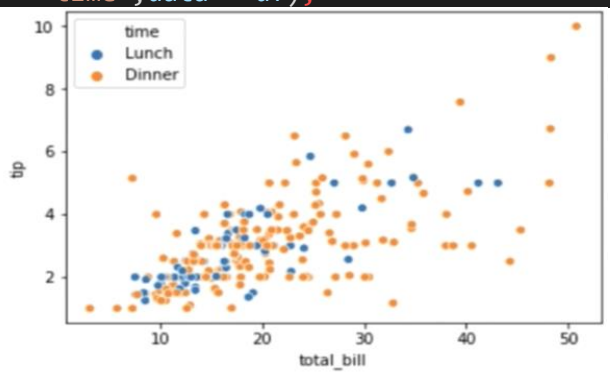
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
sns.scatterplot(x = "total_bill", y = "tip",
data = df);
```



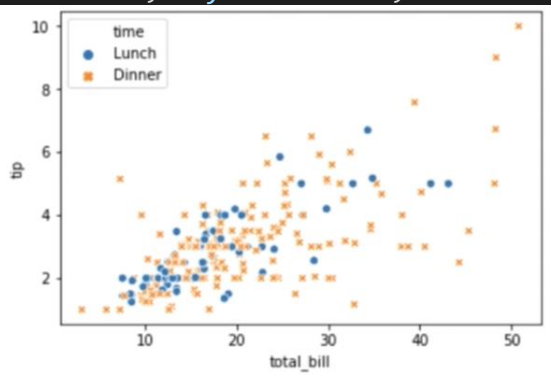
- Bu grafik ile genel olarak ödenen miktar arttıkça bahşiş(tip) arttığı gözlemlenebilir.

```
sns.scatterplot(x = "total_bill", y = "tip",
hue = "time",data = df);
```



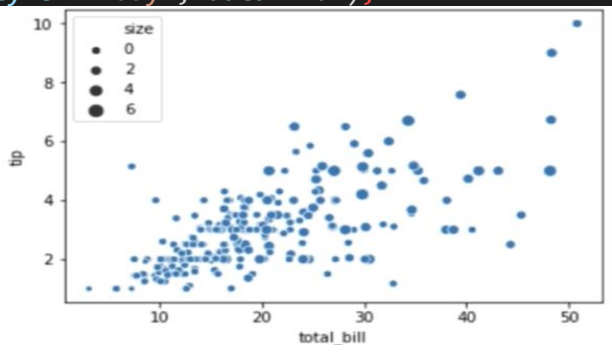
- Time değişkeni eklendiğinde akşam yemeklerinin fiyatı arttıkça bahşişlerin artmış olduğunu görüyoruz.

```
sns.scatterplot(x = "total_bill", y = "tip",
hue = "time", style = "time", data = df);
```



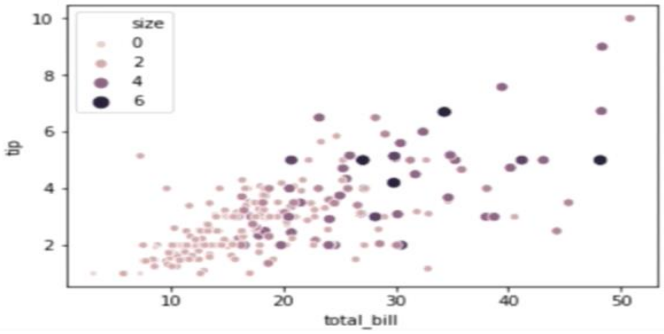
- Akşamları çarpı sembolü ile gösterdik.(Style kategorik için)

```
sns.scatterplot(x = "total_bill", y = "tip",
style = "day", data = df);
```



- Sayısal bir değer için size değişkeni girilir.

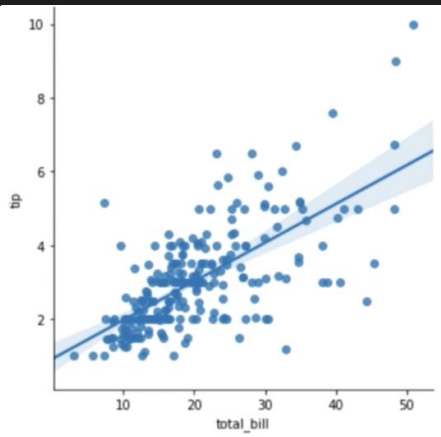
```
sns.scatterplot(x = "total_bill", y = "tip",
hue= "size", size = "size", data = df);
```



- Hue ile kullandığımızdada renklendirme işlemi oldu.

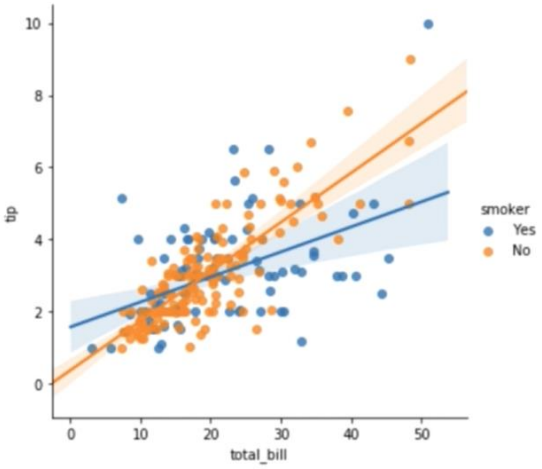
Doğrusal İlişkinin Gösterilmesi : İki değişken arasındaki ilişkiyi scatterplot gibi dağılımla değil de doğrusal bir grafikle gösterim şeklidir.

```
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);
```



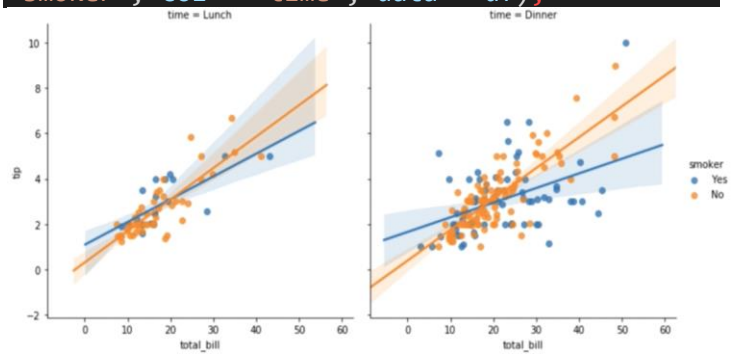
- Hafif açık olan maviler sapmaları gösteriyor. Sondaki kısımda bir anda tipin 10 olması mesela.

```
sns.lmplot(x = "total_bill", y = "tip", hue =
"smoker", data = df);
```



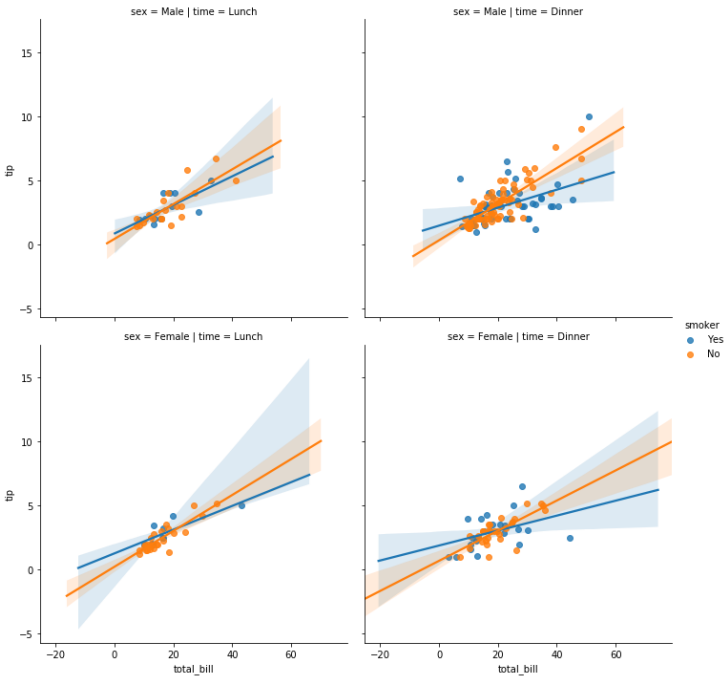
- Ödenen miktar arttıkça bahşişlerde şiddetli bir şekilde artmıştır diyebiliriz lakin sigara içmeyenler için geçerli bu.
- Sigara içenlerde eğim düştüğünü ve içmeyenlere kıyasla sert bir yükseliş yok.

```
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()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

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

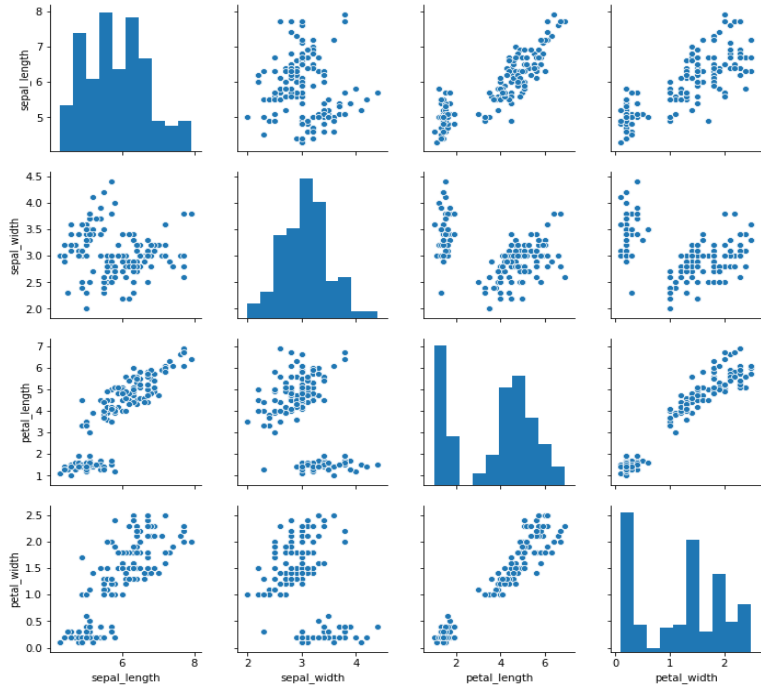
```
df.dtypes; df.shape
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

```
(150, 5)
```

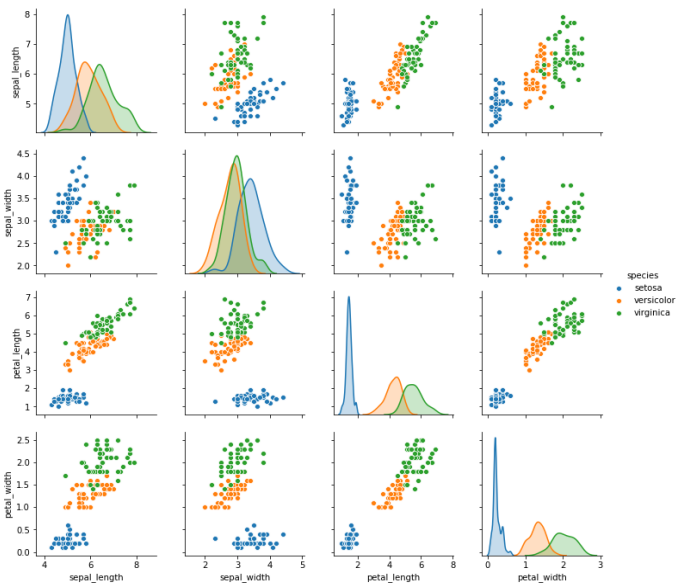
- 1 tane kategorik , 4 sayısal değişkenimiz var ve boyut 150-5.

```
sns.pairplot(df);
```

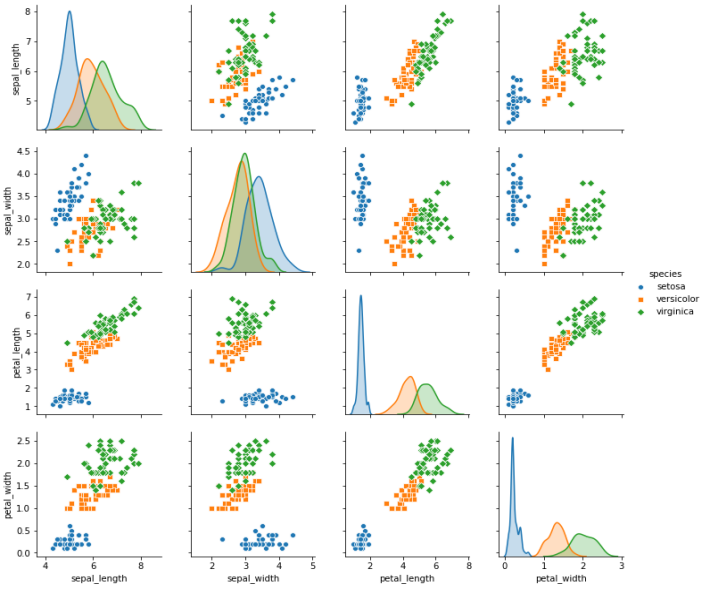


- Toz bulutu şeklinde veya hiç grafik yoksa bu iki değişken arasında ilişki yoktur demektir.
- Çeşitli noktalarda bir kümelenme varsa bu muhtemelen farklı grupları temsil eder.

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

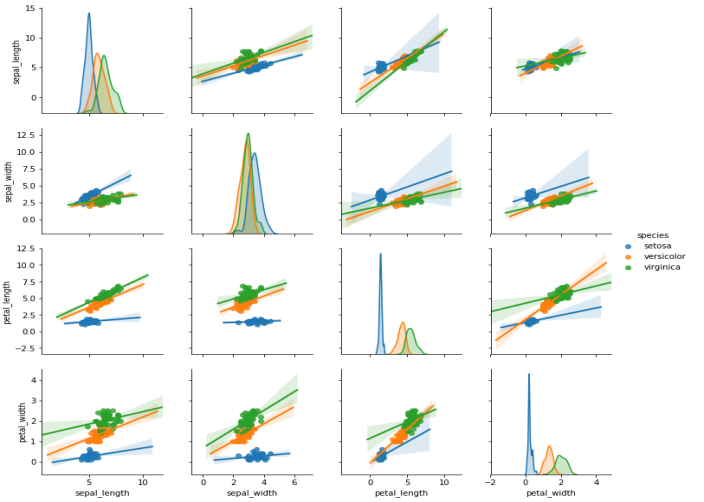



```
sns.pairplot(df, hue = "species", markers =
["o","s","D"]);
```



- 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
```

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121

```
(144, 3)
```

- Yıllara ve aylara göre yolculuk sayıları değişmektedir.

```
df["passengers"].describe()
```

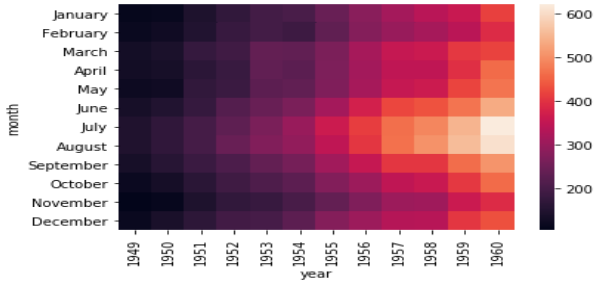
```
count    144.000000
mean     280.298611
std      119.966317
min      104.000000
25%      180.000000
50%      265.500000
75%      360.500000
max      622.000000
Name: passengers, dtype: float64
```

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

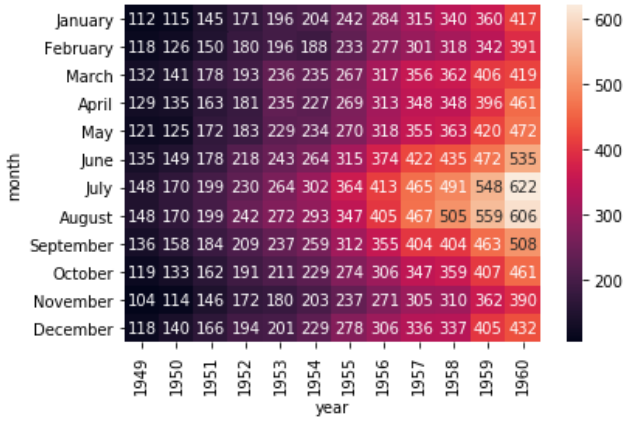
year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
January	112	115	145	171	196	204	242	284	315	340	360	417
February	118	126	150	180	196	188	233	277	301	318	342	391
March	132	141	178	193	236	235	267	317	356	362	406	419
April	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
June	135	149	178	218	243	264	315	374	422	435	472	535
July	148	170	199	230	264	302	364	413	465	491	548	622
August	148	170	199	242	272	293	347	405	467	505	559	606
September	136	158	184	209	237	259	312	355	404	404	463	508
October	119	133	162	191	211	229	274	306	347	359	407	461
November	104	114	146	172	180	203	237	271	305	310	362	390
December	118	140	166	194	201	229	278	306	336	337	405	432

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

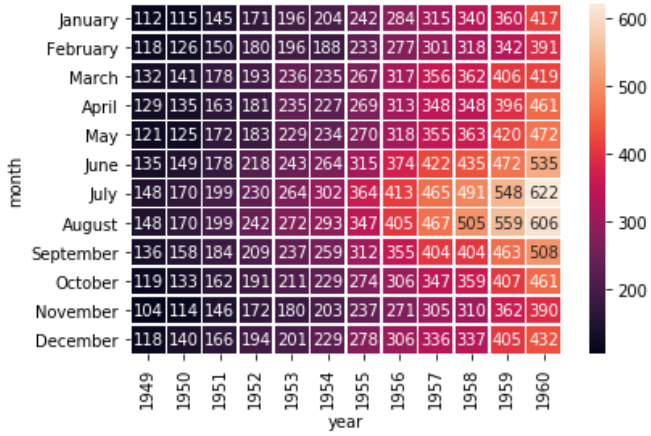
```
sns.heatmap(df);
```




```
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()
```

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

- 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
```

```
(1064, 5)
```

```
df["timepoint"].describe()
```

```
count    1064.000000
mean        9.000000
std         5.479801
min         0.000000
25%         4.000000
50%         9.000000
75%        14.000000
max        18.000000
Name: timepoint, dtype: float64
```

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

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

```
timepoint
0     56
1     56
2     56
3     56
4     56
5     56
6     56
7     56
8     56
9     56
10    56
11    56
12    56
13    56
14    56
15    56
16    56
17    56
18    56
Name: signal, dtype: int64
```

- Anlaşılacağı üzere 18 zaman diliminde 56 tane sinyal ölçülmüş.

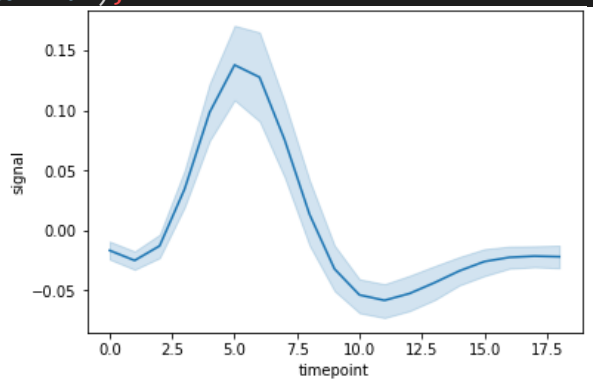
```
df.groupby("timepoint")["signal"].describe()
```

timepoint	count	mean	std	min	25%	50%	75%	max
0	56.0	-0.016662	0.028326	-0.064454	-0.039169	-0.018382	0.003539	0.074399
1	56.0	-0.025002	0.030641	-0.082174	-0.046299	-0.024533	-0.005388	0.063558
2	56.0	-0.012873	0.035440	-0.110565	-0.034944	-0.013183	0.009318	0.077277
3	56.0	0.034446	0.058260	-0.089708	-0.001157	0.028430	0.061840	0.185581
4	56.0	0.098194	0.092838	-0.046347	0.030912	0.070166	0.144911	0.346775
5	56.0	0.137725	0.123353	-0.017946	0.042762	0.096535	0.211638	0.476055
6	56.0	0.127515	0.137332	-0.054405	0.022409	0.068850	0.218919	0.564985
7	56.0	0.075660	0.129704	-0.108222	-0.016252	0.032486	0.144781	0.494787
8	56.0	0.013420	0.104216	-0.181241	-0.049453	-0.012834	0.030396	0.337143
9	56.0	-0.032041	0.072728	-0.152929	-0.075693	-0.038496	0.008717	0.221716
10	56.0	-0.053685	0.053148	-0.176453	-0.078893	-0.052906	-0.015302	0.089231
11	56.0	-0.058194	0.053828	-0.238474	-0.093127	-0.045699	-0.022522	0.030528
12	56.0	-0.052526	0.056991	-0.255486	-0.090391	-0.042294	-0.016239	0.055766
13	56.0	-0.043532	0.053598	-0.224351	-0.069285	-0.031612	-0.012958	0.059510
14	56.0	-0.033660	0.045983	-0.169312	-0.055110	-0.022165	-0.006797	0.050133
15	56.0	-0.025880	0.039092	-0.134828	-0.050536	-0.018207	0.000486	0.047102
16	56.0	-0.022414	0.035035	-0.131641	-0.041122	-0.020777	-0.001380	0.057105
17	56.0	-0.021368	0.034797	-0.121574	-0.042946	-0.017070	-0.000026	0.073757
18	56.0	-0.021867	0.036322	-0.103513	-0.046781	-0.020225	-0.002821	0.090520

- Zaman noktalarının betimsel istatistikleri.

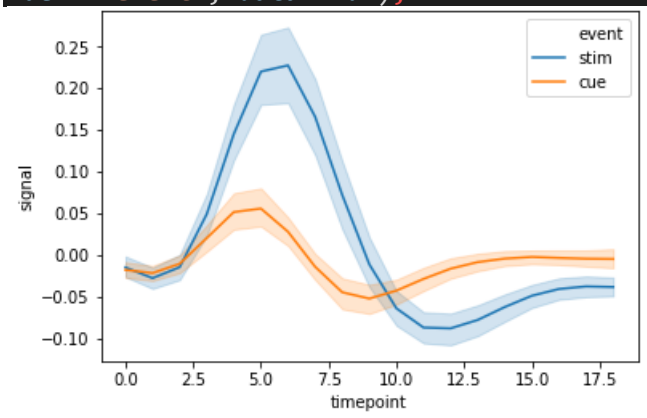
Çizgi Grafik(Lineplot()):

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



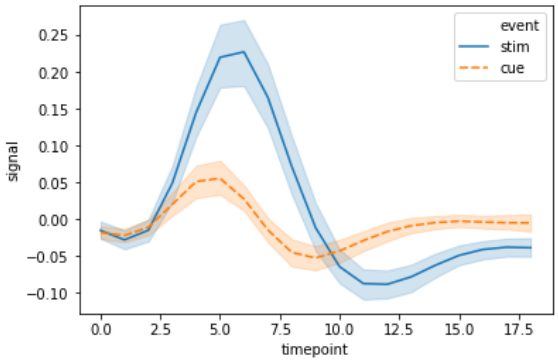
- Mavi çizgiler ortalama, acık maviler sapma.

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



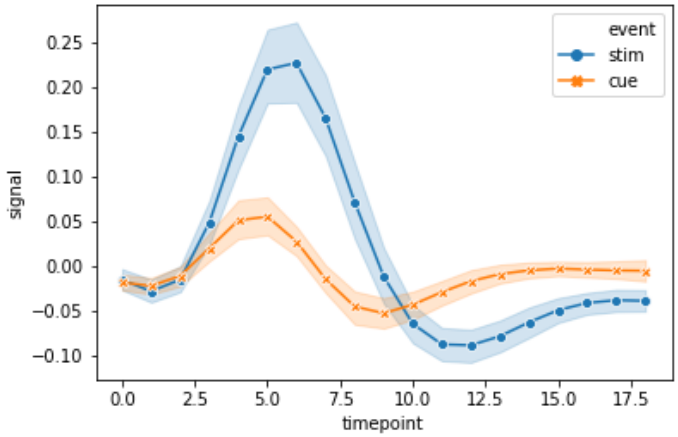
- 5.0 daki sıçrama stimden kaynaklanıyormuş.

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



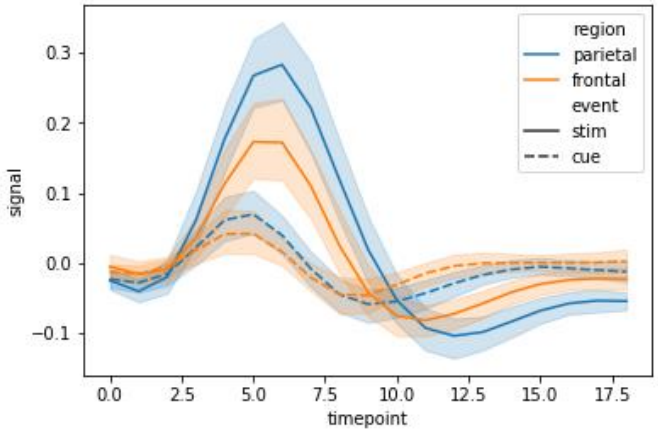
- Style ile kesik çizgili yaptık.

```
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

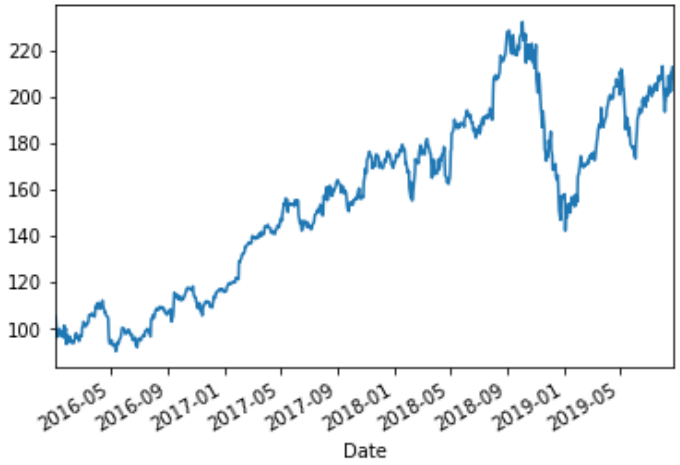
```
df = pr.get_data_yahoo("AAPL", start = "2016-
01-01", end = "2019-08-25")
```

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

```
df.head()
```

Date	High	Low	Open	Close	Volume	Adj Close
2016-01-04	105.370003	102.000000	102.610001	105.349998	67649400.0	98.742249
2016-01-05	105.849998	102.410004	105.750000	102.709999	55791000.0	96.267815
2016-01-06	102.370003	99.870003	100.559998	100.699997	68457400.0	94.383888
2016-01-07	100.129997	96.430000	98.680000	96.449997	81094400.0	90.400467
2016-01-08	99.110001	96.760002	98.550003	96.959999	70798000.0	90.878479

```
kapanis = df["Close"]
kapanis.plot();
```



- 4'er ay artarak gitmiş.
- Eksenlerin bazen tarih değişkeni mesela tarih değişkeni olduğunu belirtmemiz lazım.

```
kapanis.index
```

```
DatetimeIndex(['2016-01-04', '2016-01-05', '2016-01-06', '2016-01-07',
               '2016-01-08', '2016-01-11', '2016-01-12', '2016-01-13',
               '2016-01-14', '2016-01-15',
               ...
               '2019-08-12', '2019-08-13', '2019-08-14', '2019-08-15',
               '2019-08-16', '2019-08-19', '2019-08-20', '2019-08-21',
               '2019-08-22', '2019-08-23'],
              dtype='datetime64[ns]', name='Date', length=917, freq=None)
```

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

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

Bölüm Sonu...