```
# EDA ==> exploratory data analysis
```

```
# univariate analysis----> analysis on a single independent column.
# bivariate analysis----> analysis on two columns.
# multivariate analysis----> analysis on more then 2  columns
```

```
# numerical data==> countinuous data====> age(month,year,days)
# categorical data===> discrete data===> total number of employees
```

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt      # visualization libarary
import seaborn as sns                # matplotlib's update version is seaborn
```

```
df=pd.read_csv("/content/titanic - titanic.csv")
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga	female	22.0	1	1	3101298	12.2875	NaN	S

```
# 1.univariate analysis
```

```
df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
df['Survived'].value_counts()
```

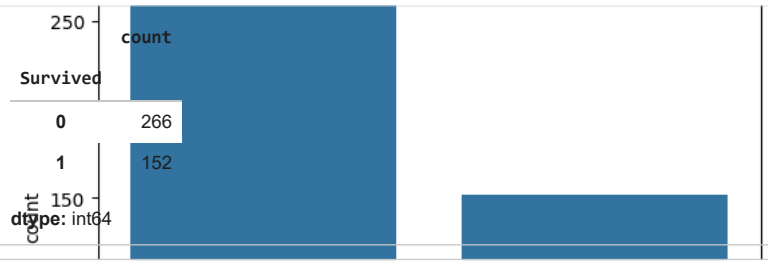
Survived	count
0	266
1	152

dtype: int64

```
sns.countplot(x= df['Survived'])
```

```
<Axes: xlabel='Survived', ylabel='count'>
```

```
df['Survived'].value_counts()
```

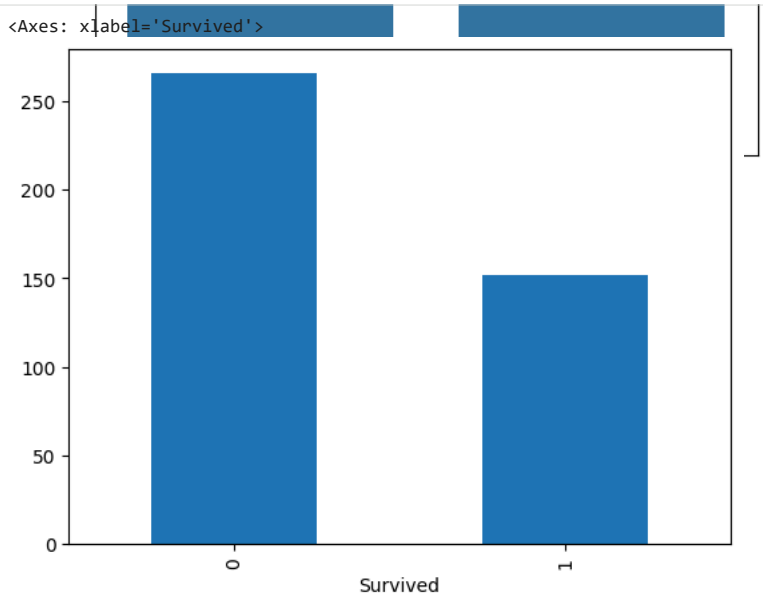


Survived	count
0	266
1	152

dtype: int64

```
df['Survived'].value_counts().plot(kind='bar')
```

```
<Axes: xlabel='Survived'>
```



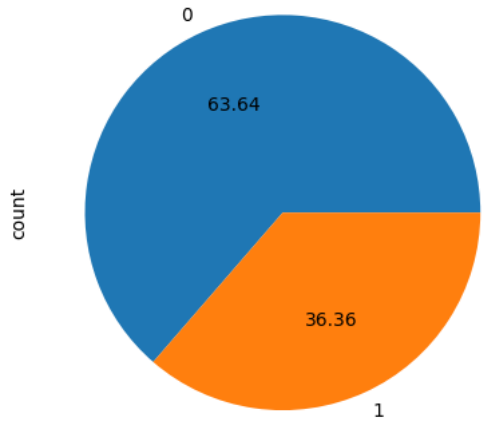

```
# if we want to find out percentage then use piechart.
df['Survived'].value_counts()
```

Survived	count
0	266
1	152

dtype: int64

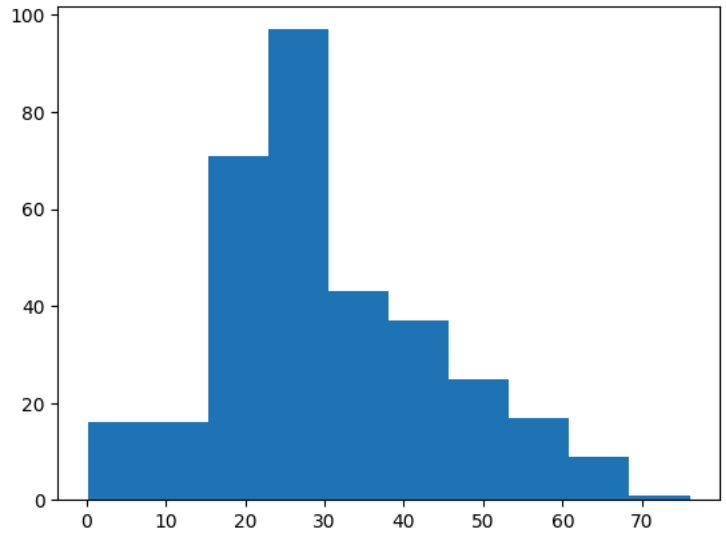
```
df['Survived'].value_counts().plot(kind='pie',autopct='%.2f')
```

```
<Axes: ylabel='count'>
```




```
# if we want numerical data then we use histrogram
# because if find the distribution
```

```
plt.hist(x=df['Age'])
plt.show()
```




```
# boxplot
# for find the outliers

# 1.lower fence
# 2. 25% data
# 3.iqr(inter quartile range)(75%-25%)
# 4.75% data
# 5.upper fence
```

```
# x=1,2,3,4,5
# mean=(1+2+3+4+5)/5=15/5=3
# x=1,2,3,4,5,100
# mean= 15+100/6=115/6=19.1                # here it 100 outlier
```

```
sns.boxplot(x=df['Age'])
```

```
# app reviews sentiment analysis
```

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("/content/linkedin-reviews - linkedin-reviews (3)d.csv")
df.head()
```

	Review	Rating
0	Does absolutely nothing for a LinkedIn beginne...	1
1	Force close(galaxy tab)	1
2	Slow and it tries to upload your contacts with...	1
3	Add ability to customize the profile and move ...	4
4	Good app, but it's a pain that it's not possib...	4

```
import matplotlib.pyplot as plt
import seaborn as sns
```

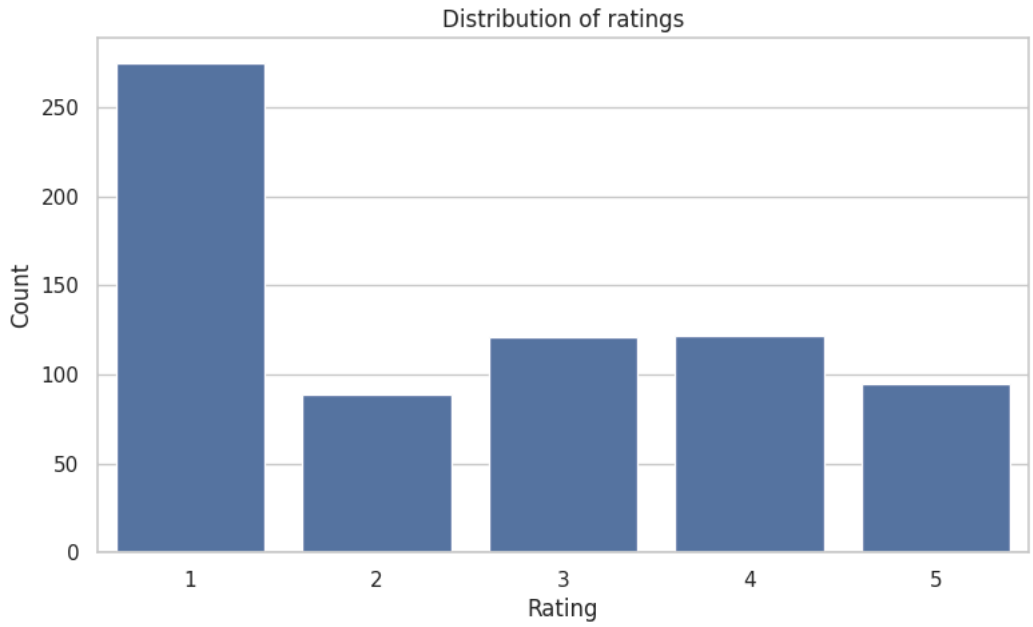
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 702 entries, 0 to 701
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Review  702 non-null    object
 1   Rating  702 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 11.1+ KB
```

```
# exploratory data analysis.
# we will start by analyzing the distributions of ratings.it will provide insight into the overall
# sentiment of the reviews .then we can explore further. such as analyzing the length of reviews
# and possibly derive insights from the next of the reviews
```

```
# plotting the distribution of ratings
```

```
sns.set(style='whitegrid')
plt.figure(figsize=(9,5))
sns.countplot(data=df,x='Rating')
plt.title('Distribution of ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```




```
from textblob import TextBlob
```

```
def textblob_sentiment_analysis(reviews):
  sentiment = TextBlob(reviews).sentiment
  if sentiment.polarity>0.1:
   return 'positive'
  elif sentiment.polarity<-0.1:
   return'negative'
  else:
    return 'neutral'
```

```
df['sentiment']=df['Review'].apply(textblob_sentiment_analysis)
```

```
from textblob import Textblob
```

```
 df.sample(2)
```

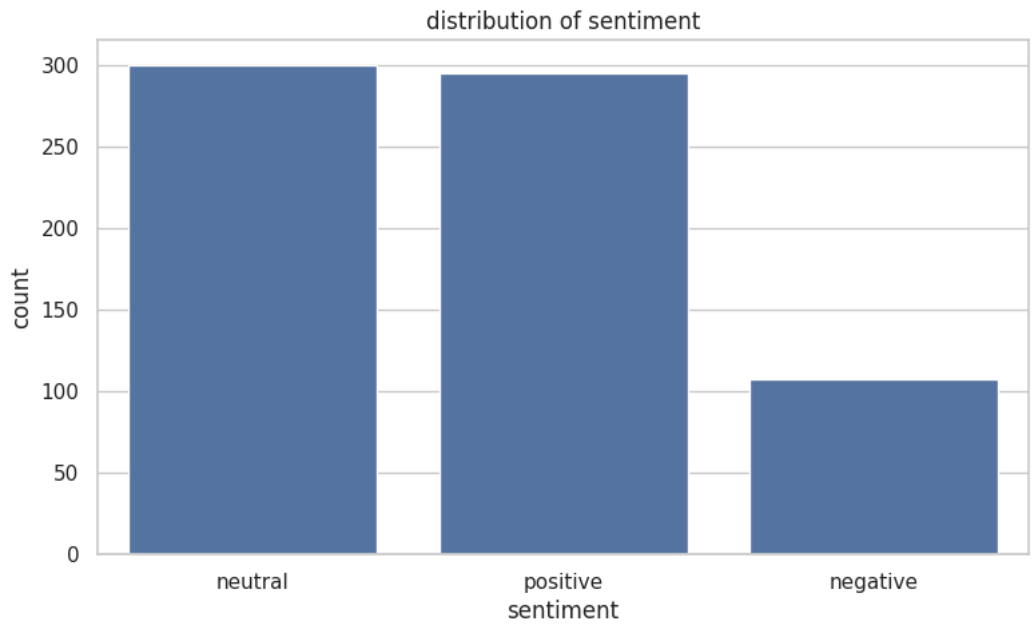
```
sentiment_distribution=df['sentiment'].value_counts()
sentiment_distribution
```

sentiment	count
neutral	300
positive	295
negative	107

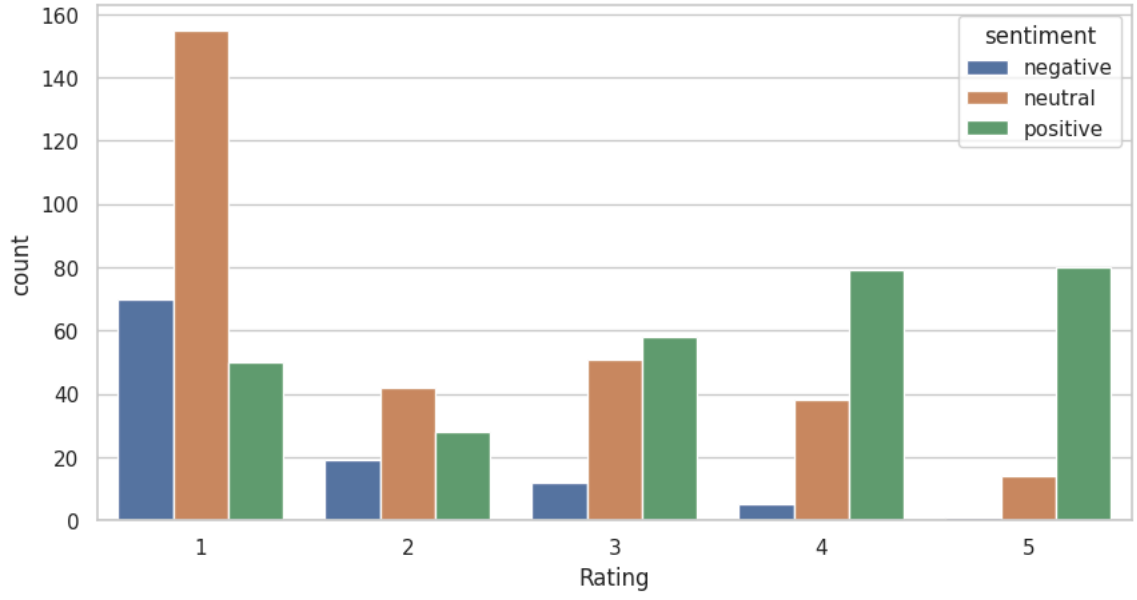
dtype: int64

```
plt.figure(figsize=(9,5))
sns.barplot(x=sentiment_distribution.index,
            y=sentiment_distribution.values)
plt.title("distribution of sentiment")
```

```
plt.xlabel("sentiment")
plt.ylabel("count")
plt.show()
```




```
plt.figure(figsize=(10,5))
sns.countplot(data=df,
              x='Rating',
              hue='sentiment')
plt.xlabel("Rating")
plt.ylabel("count")
plt.legend(title='sentiment')
plt.show()
```




```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=sns.load_dataset('tips')
df.head()
```

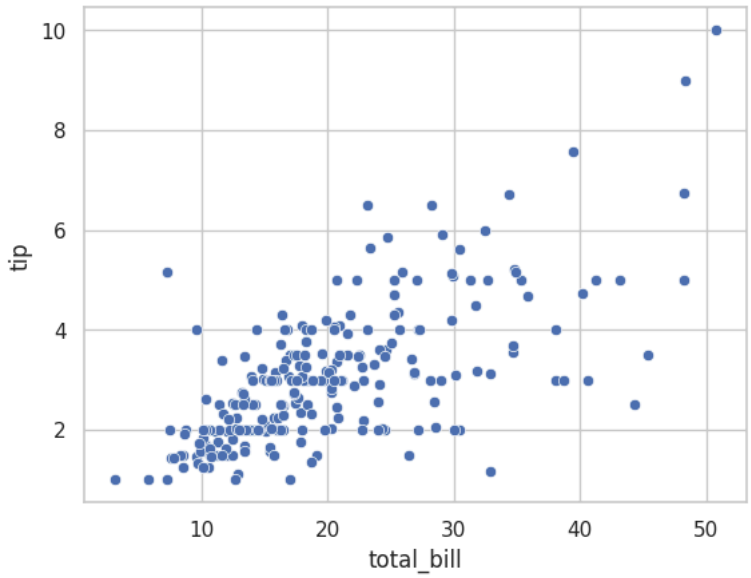
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
# bivariate analysis
```

```
# 1.scatterplot(numerical column - numerical column)
```

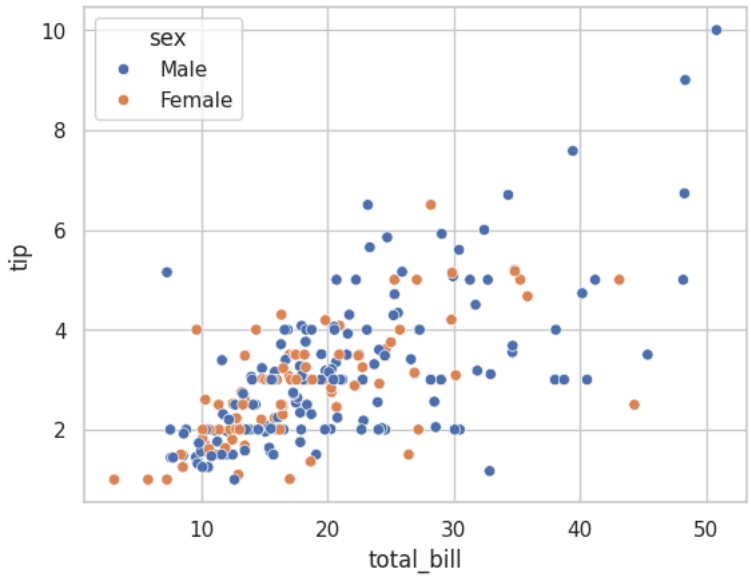
```
sns.scatterplot(x=tips['total_bill'],
                y=tips['tip'])
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



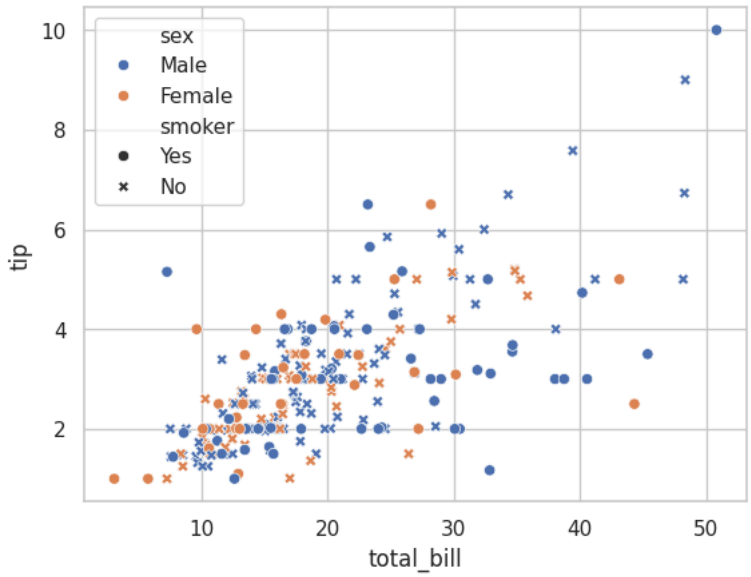

```
sns.scatterplot(x='total_bill',y='tip',data=tips,hue=tips['sex'])
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```




```
sns.scatterplot(x='total_bill',
                y='tip',data=tips,
                hue=tips['sex'],
                style=tips['smoker'])
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```

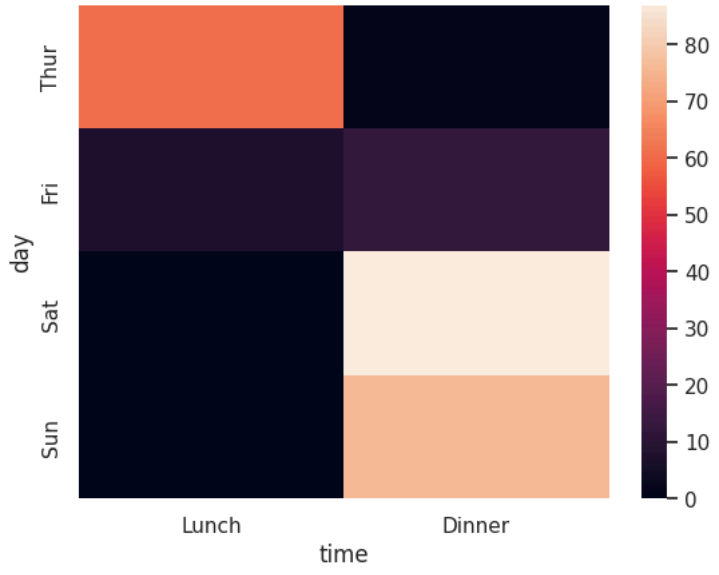



```
# heatmap(categorical to categorical data ke liye crosstab)
p=pd.crosstab(tips['day'],tips['time'])
p
```

time	Lunch	Dinner
day		
Thur	61	1
Fri	7	12
Sat	0	87
Sun	0	76

```
sns.heatmap(p)
```

```
<Axes: xlabel='time', ylabel='day'>
```




```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("/content/Attrition - Attrition (1)d.csv")
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Emplo
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	

5 rows × 35 columns

```
import plotly.graph_objects  as go
import plotly.io as px
px.templates.default='plotly_white'
```

```
df.isnull().sum()
```

	0
Age	0
Attrition	0
BusinessTravel	0
DailyRate	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EmployeeCount	0
EmployeeNumber	0
EnvironmentSatisfaction	0
Gender	0
HourlyRate	0
JobInvolvement	0
JobLevel	0
JobRole	0
JobSatisfaction	0
MaritalStatus	0
MonthlyIncome	0
MonthlyRate	0
NumCompaniesWorked	0
Over18	0
OverTime	0
PercentSalaryHike	0
PerformanceRating	0
RelationshipSatisfaction	0
StandardHours	0
StockOptionLevel	0
TotalWorkingYears	0
TrainingTimesLastYear	0
WorkLifeBalance	0
YearsAtCompany	0
YearsInCurrentRole	0
YearsSinceLastPromotion	0
YearsWithCurrManager	0

dtype: int64

```
# check percentage of attrition by department
```

```
# filter the data to show only 'yes' value in the 'attrition' column
```

```
attr_df=df[df['Attrition']=='Yes']
```

```
attr_df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Em
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	
14	28	Yes	Travel_Rarely	103	Research & Development	24	3	Life Sciences	1	
21	36	Yes	Travel_Rarely	1218	Sales	9	4	Life Sciences	1	
24	34	Yes	Travel_Rarely	699	Research & Development	6	1	Medical	1	
...	...	...	...	...	...	...	...	...	...	...
1438	23	Yes	Travel_Frequently	638	Sales	9	3	Marketing	1	
1442	29	Yes	Travel_Rarely	1092	Research & Development	1	4	Medical	1	
1444	56	Yes	Travel_Rarely	310	Research & Development	7	2	Technical Degree	1	
1452	50	Yes	Travel_Frequently	878	Sales	1	4	Life Sciences	1	
1461	50	Yes	Travel_Rarely	410	Sales	28	3	Marketing	1	

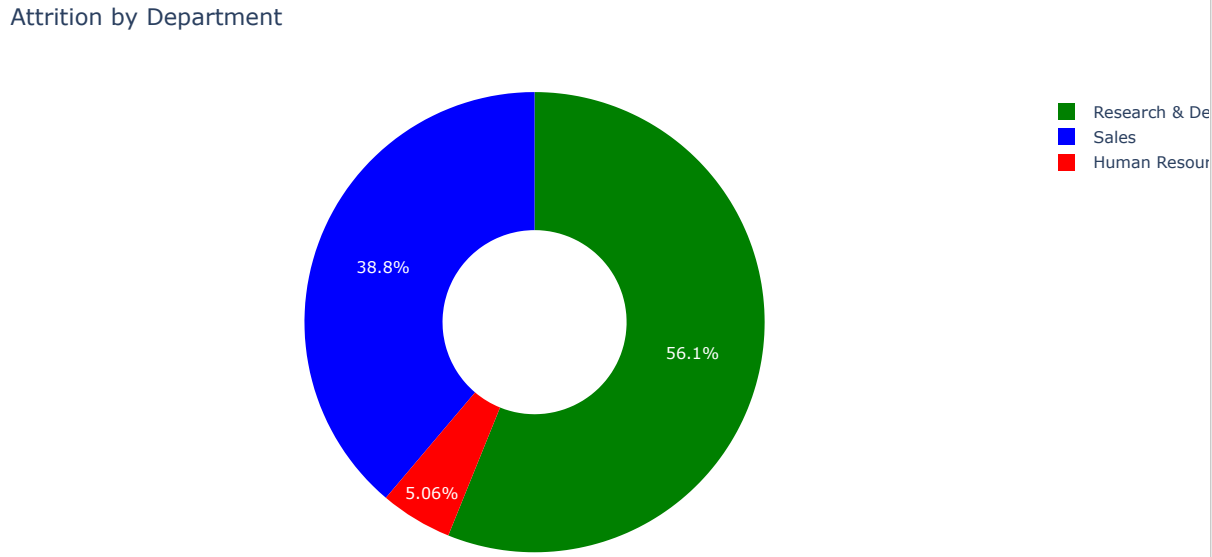
237 rows × 35 columns

```
# calculate the attrition by department
```

```
from os import name
attrition_by_dept=attr_df.groupby('Department').size().reset_index(name='count')
attrition_by_dept
```

	Department	count
0	Human Resources	12
1	Research & Development	133
2	Sales	92

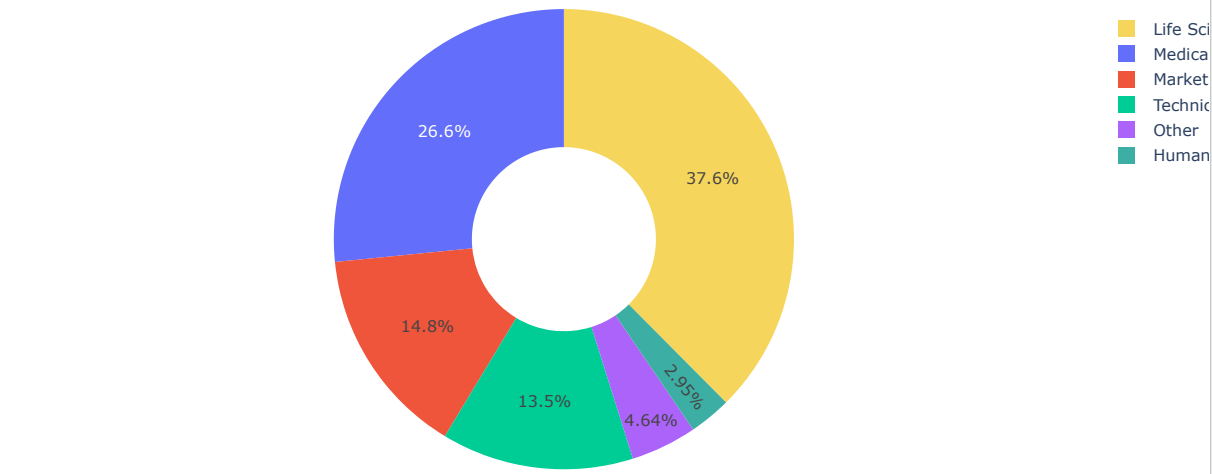
```
# create a donut chart
fig = go.Figure(data=[go.Pie(
    labels=attrition_by_dept['Department'],
    values=attrition_by_dept['count'],
    hole=0.4,
    marker= dict(colors=['red','green', 'blue']),
    textposition='inside'
)])

# update the layout
fig.update_layout(title='Attrition by Department')

# show the chart
fig.show()
```




```
attrition_by= attr_df.groupby('EducationField').size().reset_index(name='count')
attrition_by
```

	EducationField	count
0	Human Resources	7
1	Life Sciences	89
2	Marketing	35
3	Medical	63
4	Other	11
5	Technical Degree	32

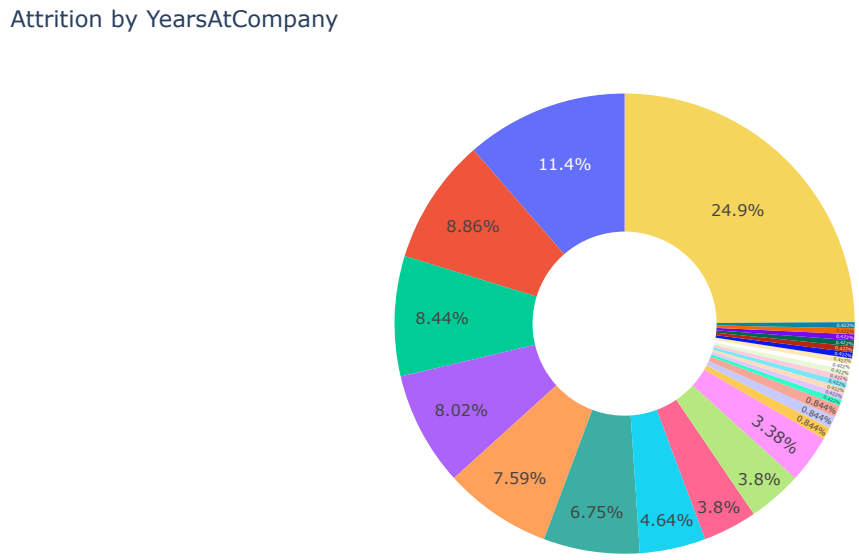
```
attrition_by_dept=attr_df.groupby('EducationField').size().reset_index(name='count')

# create a donut chart
fig = go.Figure(data=[go.Pie(
    labels=attrition_by_dept['EducationField'],
    values=attrition_by_dept['count'],
    hole=0.4,
    marker= dict(colors=['#3CAEA3','#F6D55C']),
    textposition='inside'
)])

# update the layout
fig.update_layout(title='Attrition by EducationField')

# show the chart
fig.show()
```




```
attrition_by_dept=attr_df.groupby('YearsAtCompany').size().reset_index(name='count')
attrition_by_dept
```

	YearsAtCompany	count
0	0	16
1	1	59
2	2	27
3	3	20
4	4	19
5	5	21
6	6	9
7	7	11
8	8	9
9	9	8
10	10	18
11	11	2
12	13	2
13	14	2
14	15	1
15	16	1
16	17	1
17	18	1
18	19	1
19	20	1
20	21	1
21	22	1
22	23	1
23	24	1
24	31	1
25	32	1
26	33	1
27	40	1

```
attrition_by_dept = attr_df.groupby('YearsAtCompany').size().reset_index(name='count')
fig = go.Figure(data=[go.Pie(
    labels=attrition_by_dept['YearsAtCompany'],
    values=attrition_by_dept['count'],
    hole=0.4,
    marker= dict(colors=['#3CAEA3','#F6D55C']),
    textposition='inside'
)])

fig.update_layout(title='Attrition by YearsAtCompany')

fig.show()
```



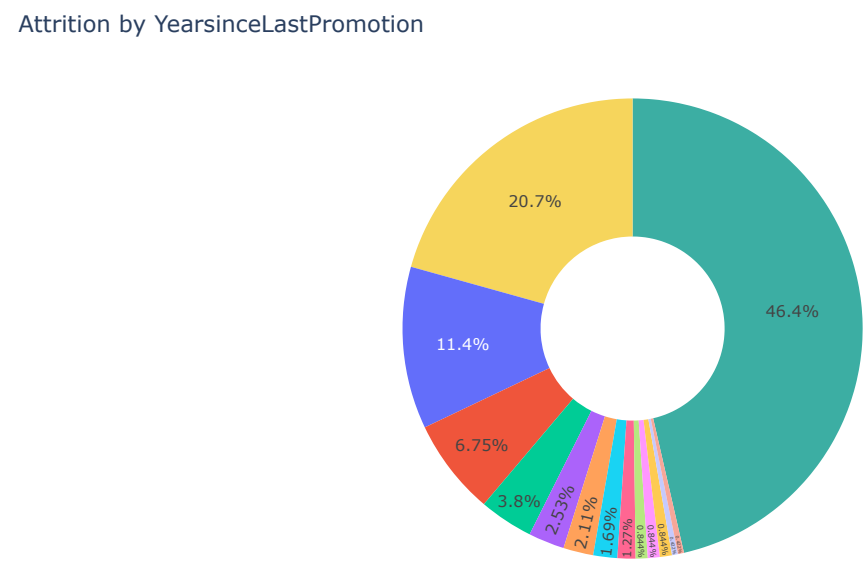

```
# we can see that most of the employees leave the organization after completing
```

```
attrition_by_dept=attr_df.groupby('YearsSinceLastPromotion').size().reset_index(name='count')
attrition_by_dept
```

	YearsSinceLastPromotion	count
0	0	110
1	1	49
2	2	27
3	3	9
4	4	5
5	5	2
6	6	6
7	7	16
8	9	4
9	10	1
10	11	2
11	13	2
12	14	1
13	15	3

```
fig = go.Figure(data=[go.Pie(
    labels=attrition_by_dept['YearsSinceLastPromotion'],
    values=attrition_by_dept['count'],
    hole=0.4,
    marker= dict(colors=['#3CAEA3','#F6D55C']),
    textposition='inside'
)])
```

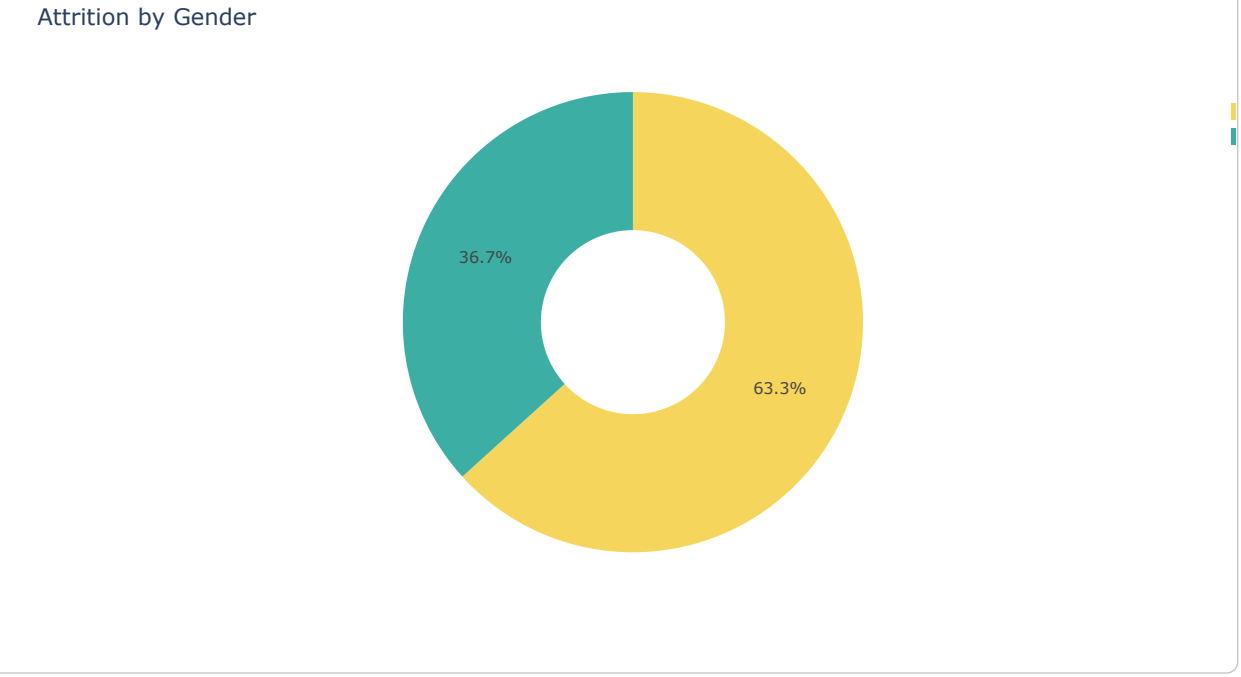
```
fig.update_layout(title='Attrition by YearsinceLastPromotion')

fig.show()
```




```
attrition_by_dept=attr_df.groupby('Gender').size().reset_index(name='count')
attrition_by_dept
```

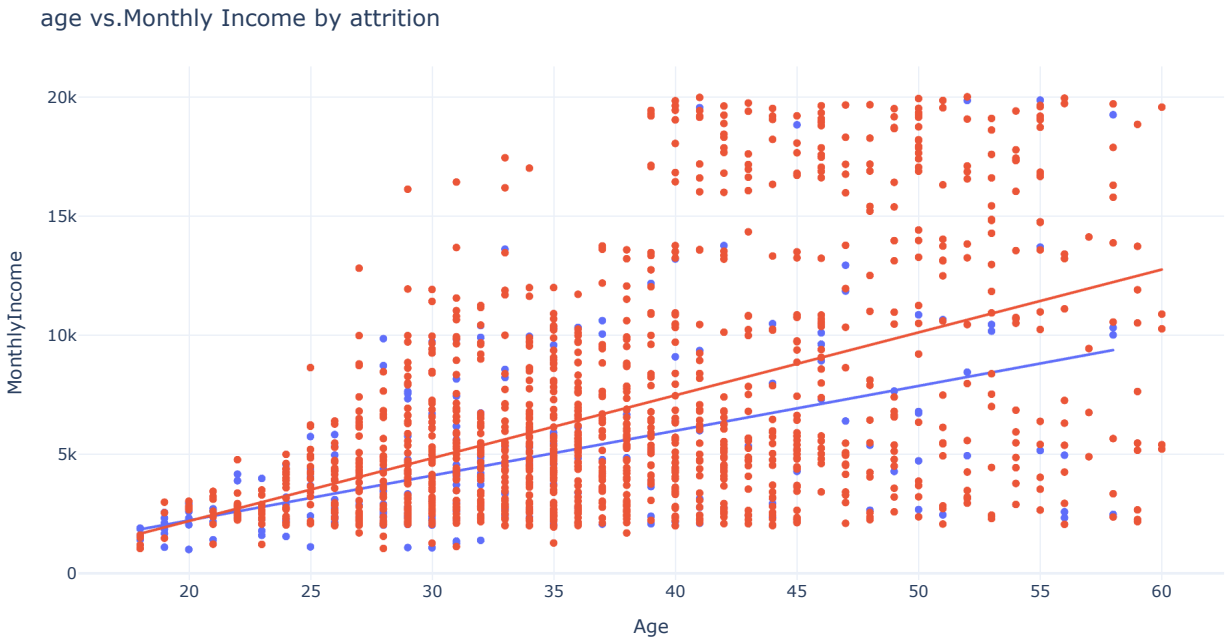
	Gender	count
0	Female	87
1	Male	150

```
# create a donut chart
# Re-calculate attrition by Gender to ensure correct data is used
attrition_by_dept=attr_df.groupby('Gender').size().reset_index(name='count')

fig = go.Figure(data=[go.Pie(
    labels=attrition_by_dept['Gender'],
    values=attrition_by_dept['count'],
    hole=0.4,
    marker= dict(colors=['#3CAEA3','#F6D55C']),
    textposition='inside'
)])

# update the layout
fig.update_layout(title='Attrition by Gender')

# show the chart
fig.show()
```




```
import plotly.express as px
```

```
import plotly.express as px
fig= px.scatter(df,x='Age',y='MonthlyIncome',color='Attrition',
                trendline='ols')
fig.update_layout(title="age vs.Monthly Income by attrition")
fig.show()
```




```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv("/content/supply_chain - supply_chain.d.csv")
df.head()
```

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Order quantities	...	Location	Lea tim
0	haircare	SKU0	69.808006	55	802	8661.996792	Non-binary	58	7	96	...	Mumbai	2
1	skincare	SKU1	14.843523	95	736	7460.900065	Female	53	30	37	...	Mumbai	2
2	haircare	SKU2	11.319683	34	8	9577.749626	Unknown	1	10	88	...	Mumbai	1
3	skincare	SKU3	61.163343	68	83	7766.836426	Non-binary	23	13	59	...	Kolkata	2
4	skincare	SKU4	4.805496	26	871	2686.505152	Non-binary	5	3	56	...	Delhi	

5 rows × 24 columns

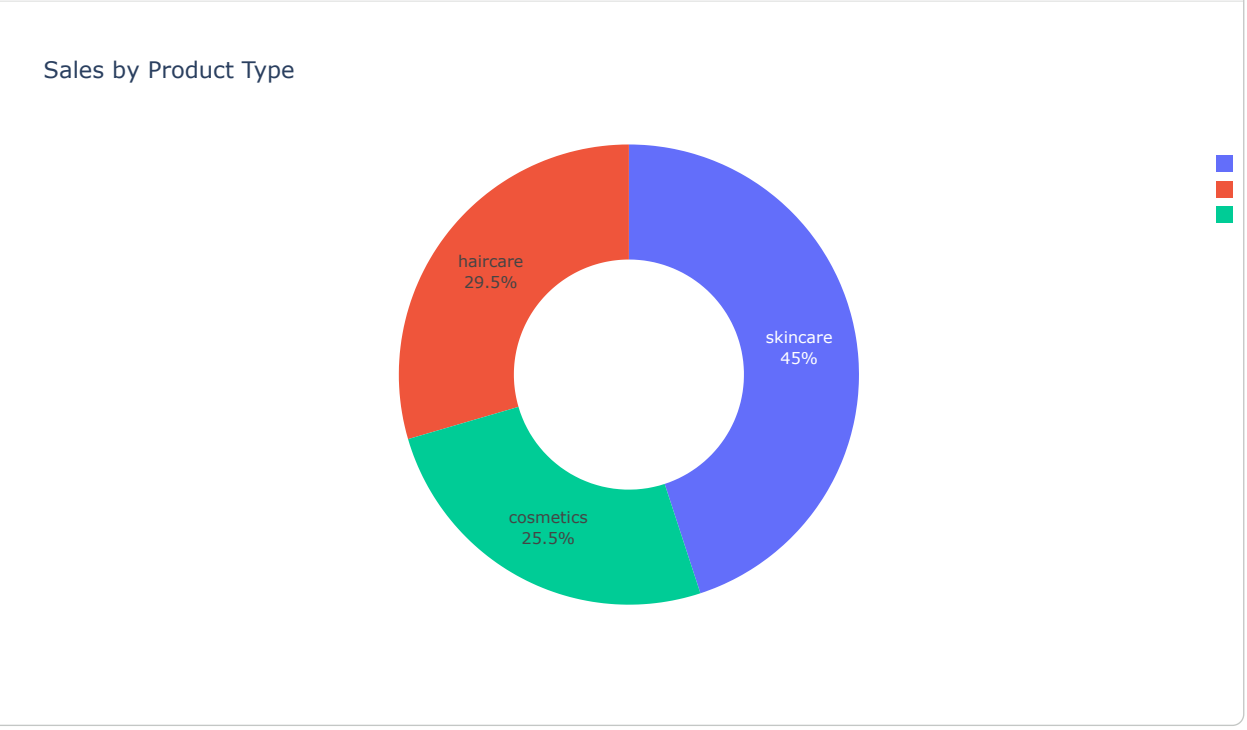
```
import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go
pio.templates.default ='plotly_white'
```

```
df.describe()
```

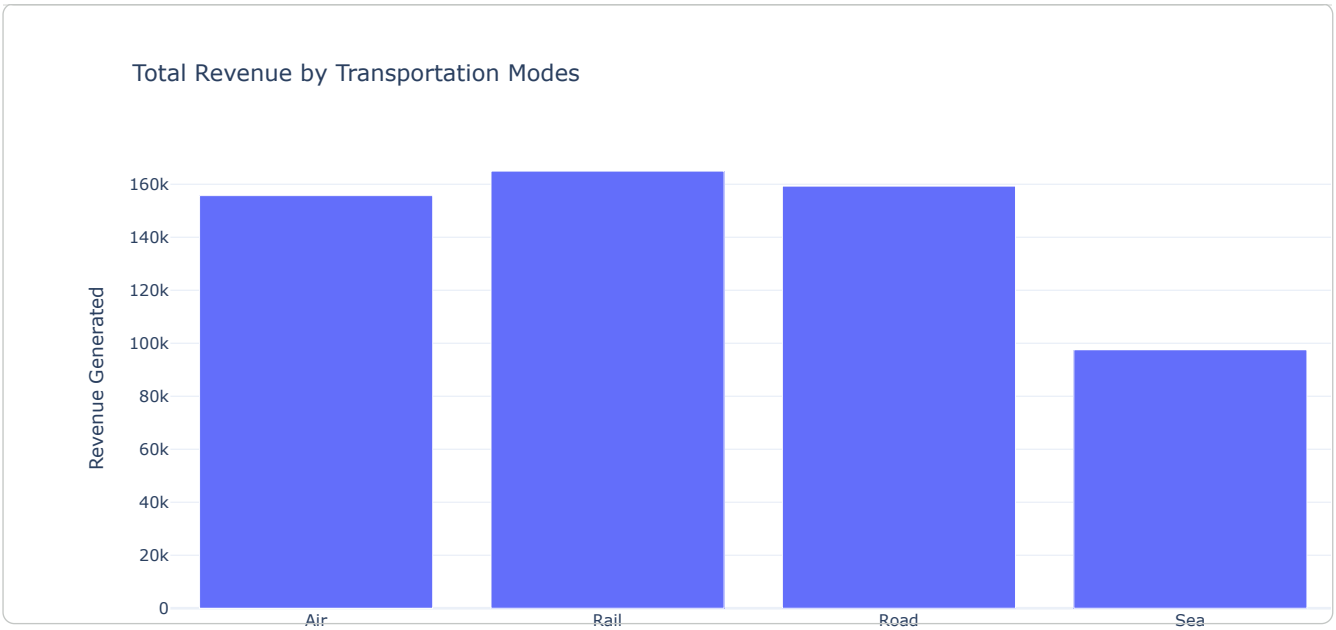
	Price	Availability	Number of products sold	Revenue generated	Stock levels	Lead times	Order quantities	Shipping times	Shipping costs	Lead time
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	49.462461	48.400000	460.990000	5776.048187	47.770000	15.960000	49.220000	5.750000	5.548149	17.080000
std	31.168193	30.743317	303.780074	2732.841744	31.369372	8.785801	26.784429	2.724283	2.651376	8.846251
min	1.699976	1.000000	8.000000	1061.618523	0.000000	1.000000	1.000000	1.000000	1.013487	1.000000
25%	19.597823	22.750000	184.250000	2812.847151	16.750000	8.000000	26.000000	3.750000	3.540248	10.000000
50%	51.239830	43.500000	392.500000	6006.352023	47.500000	17.000000	52.000000	6.000000	5.320534	18.000000
75%	77.198228	75.000000	704.250000	8253.976920	73.000000	24.000000	71.250000	8.000000	7.601695	25.000000
max	99.171329	100.000000	996.000000	9866.465458	100.000000	30.000000	96.000000	10.000000	9.929816	30.000000

```
fig = px.scatter(df,x='Price',
                 y='Revenue generated',
                 color='Product type',
                 hover_data=['Number of products sold'],
                 trendline='ols')
fig.show()
```


```
sales_data = df.groupby('Product type')['Number of products sold'].sum().reset_index()

pie_chart = px.pie(sales_data, values='Number of products sold',
                   names='Product type',
                   title='Sales by Product Type',
                   hover_data=['Number of products sold'],
                   hole=0.5,
                   color_discrete_sequence=px.colors.qualitative.Plotly)
pie_chart.update_traces(textposition='inside', textinfo='percent+label')
pie_chart.show()
```




```
total_revenue =df.groupby("Transportation modes")['Revenue generated'].sum().reset_index()
fig=go.Figure()
fig.add_trace(go.Bar(x=total_revenue['Transportation modes'],
                     y=total_revenue['Revenue generated']))
fig.update_layout(title='Total Revenue by Transportation Modes',
                  xaxis_title='Transportation Modes',
                  yaxis_title='Revenue Generated')
fig.show()
```




```
avg_lead_time=df.groupby('Product type')['Lead time'].mean().reset_index()

avg_manufacturing_cost=df.groupby('Product type')['Manufacturing costs'].mean().reset_index()
result = pd.merge(avg_lead_time,avg_manufacturing_cost,on='Product type')
result.rename(columns={'Lead time':'average lead time' ,
                       'Manufacturing costs':'average manufacturing cost'},
              inplace=True)
print(result)
```

```
  Product type  average lead time  average manufacturing cost
0    cosmetics          13.538462                   43.052740
1     haircare          18.705882                   48.457993
2     skincare          18.000000                   48.993157
```

```
revenue_chart=px.line(df,x='SKU',
                      y='Revenue generated',
                      title='revenue generated by sku')
revenue_chart.show()
```