



RAJALAKSHMI ENGINEERING COLLEGE

Approved by AICTE | Affiliated to Anna University | Accredited by NAAC

Department of Computer Science and Engineering

CS23334 Fundamentals of Data Science Lab

III semester II Year (2023R)

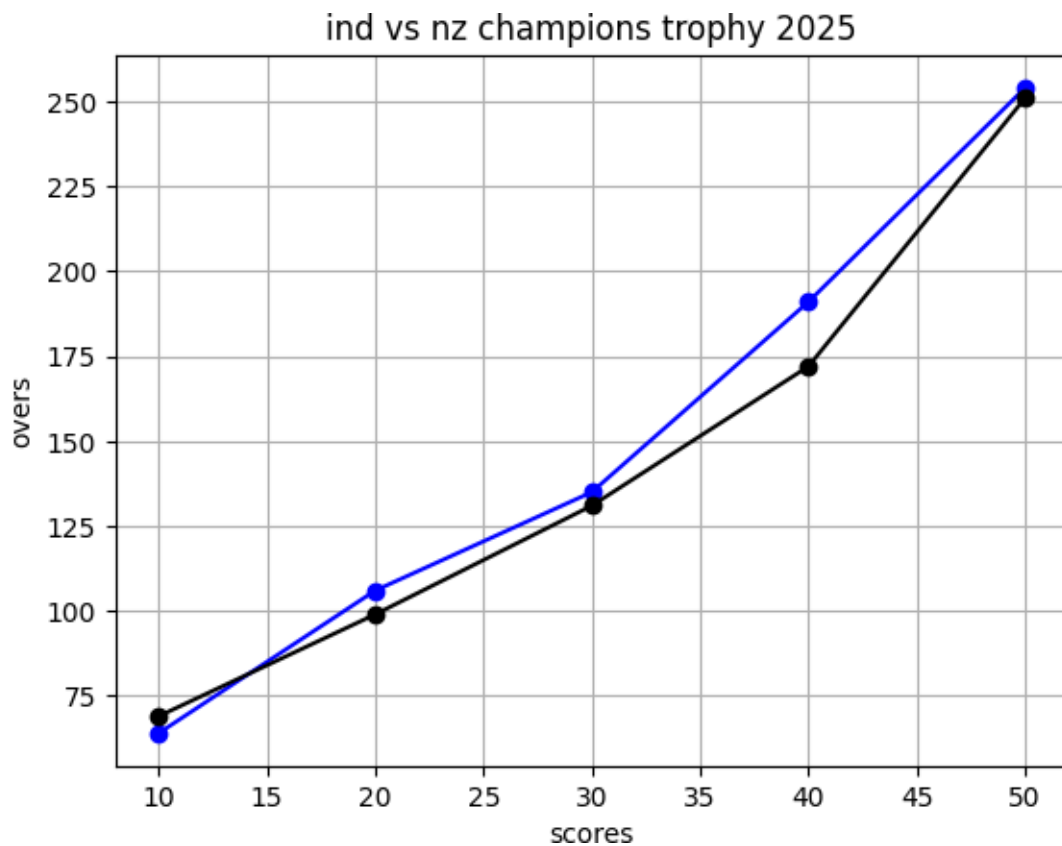
Name of the Student : SARVESH R

Register Number : 240701477

nnjushrr

November 18, 2025

```
[8]: from matplotlib import pyplot as plt
overs=[10,20,30,40,50]
india=[64,106,135,191,254]
nz=[69,99,131,172,251]
plt.plot(overs,india,color='b',marker='o')
plt.plot(overs,nz,color='black',marker='o')
plt.xlabel("scores")
plt.ylabel("overs")
plt.title("ind vs nz champions trophy 2025")
plt.grid(True)
plt.show()
```

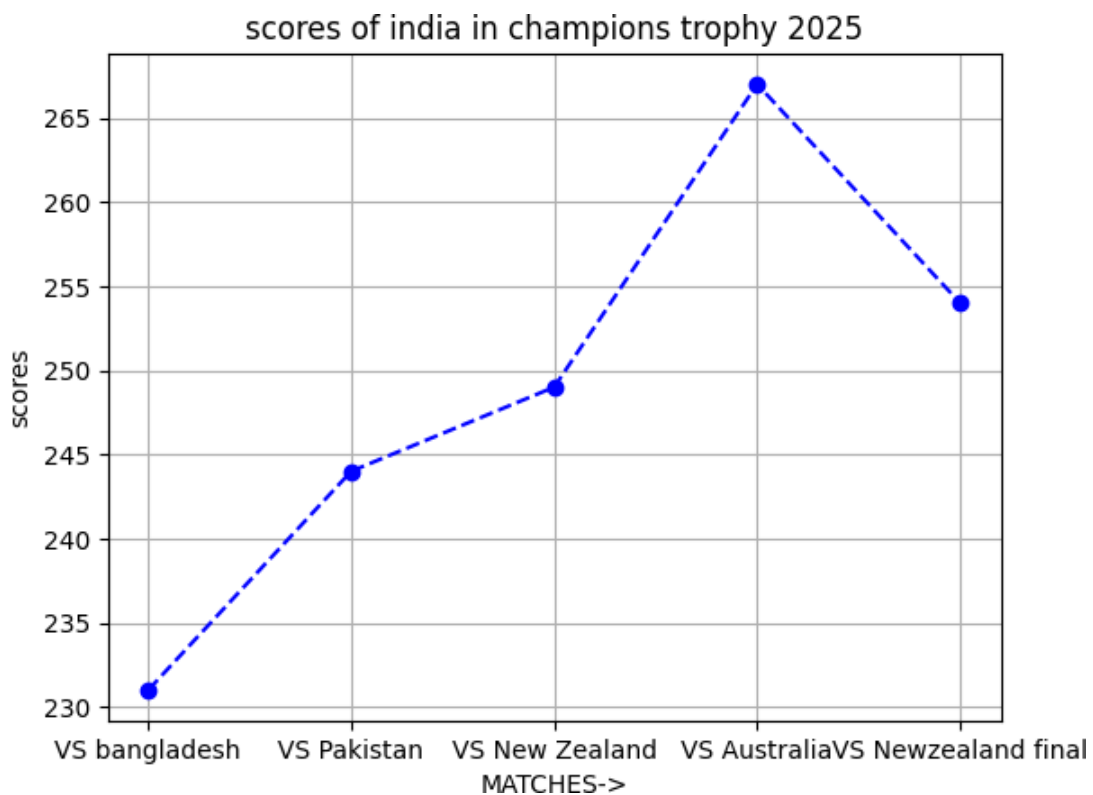


[]:

ipaczigq7

November 18, 2025

```
[15]: from matplotlib import pyplot as plt
matches=['VS bangladesh','VS Pakistan','VS New Zealand','VS Australia','VS_
sNewzealand final']
scores=[231,244,249,267,254]
plt.plot(matches,scores,color='b',marker='o',linestyle='--')
plt.title("scores of india in champions trophy 2025")
plt.xlabel("MATCHES->")
plt.ylabel("scores")
plt.grid(True)
plt.show()
```

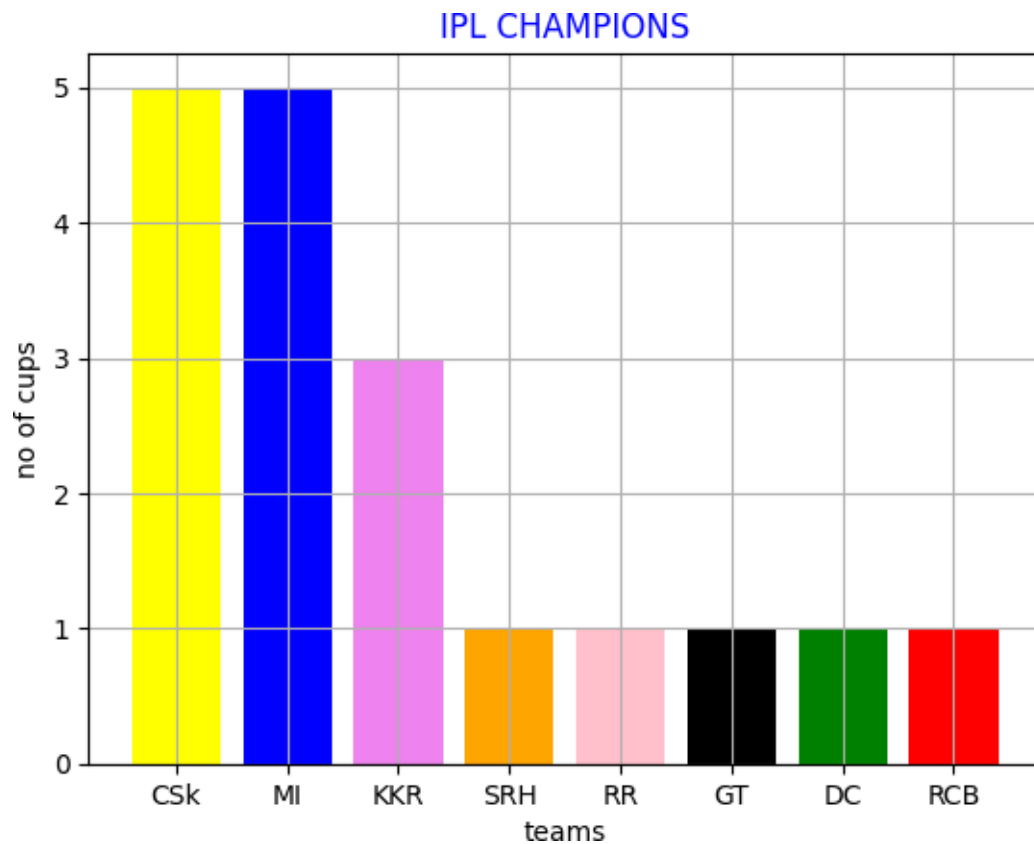


[]:

exp1c

November 18, 2025

```
[3]: from matplotlib import pyplot as plt
ipl={'CSK':5,'MI':5,'KKR':3,'SRH':1,'RR':1,'GT':1,'DC':1,'RCB':1}
cups=list(ipl.values())
teams=list(ipl.keys())
colors=['yellow','blue','violet','orange','pink','black','green','red']
plt.bar(teams,cups,color=colors)
plt.title("IPL CHAMPIONS",color='b')
plt.xlabel("teams")
plt.ylabel("no of cups")
plt.grid(True)
plt.show()
```



[]:

exp1d

November 18, 2025

```
[8]: from matplotlib import pyplot as plt

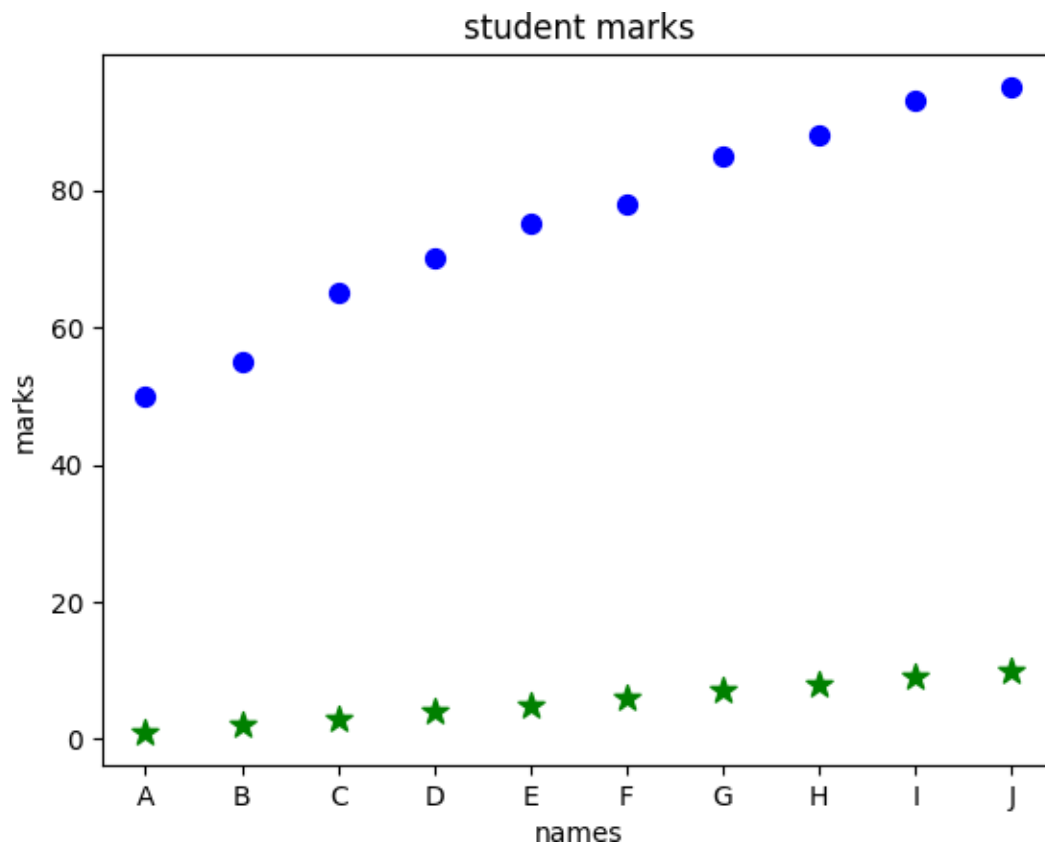
student_data = {
    'A': (1, 50),
    'B': (2, 55),
    'C': (3, 65),
    'D': (4, 70),
    'E': (5, 75),
    'F': (6, 78),
    'G': (7, 85),
    'H': (8, 88),
    'I': (9, 93),
    'J': (10, 95)
}
students=list(student_data.keys())
hours=[value[0] for value in student_data.values()]
marks=[value[1] for value in student_data.values()]

plt.scatter(students,hours,color='g',marker='*',s=100)
plt.scatter(students,marks,color='b',marker='.',s=200)

plt.xlabel("names")

plt.ylabel("marks")
plt.title("student marks")

plt.show()
```

[]:

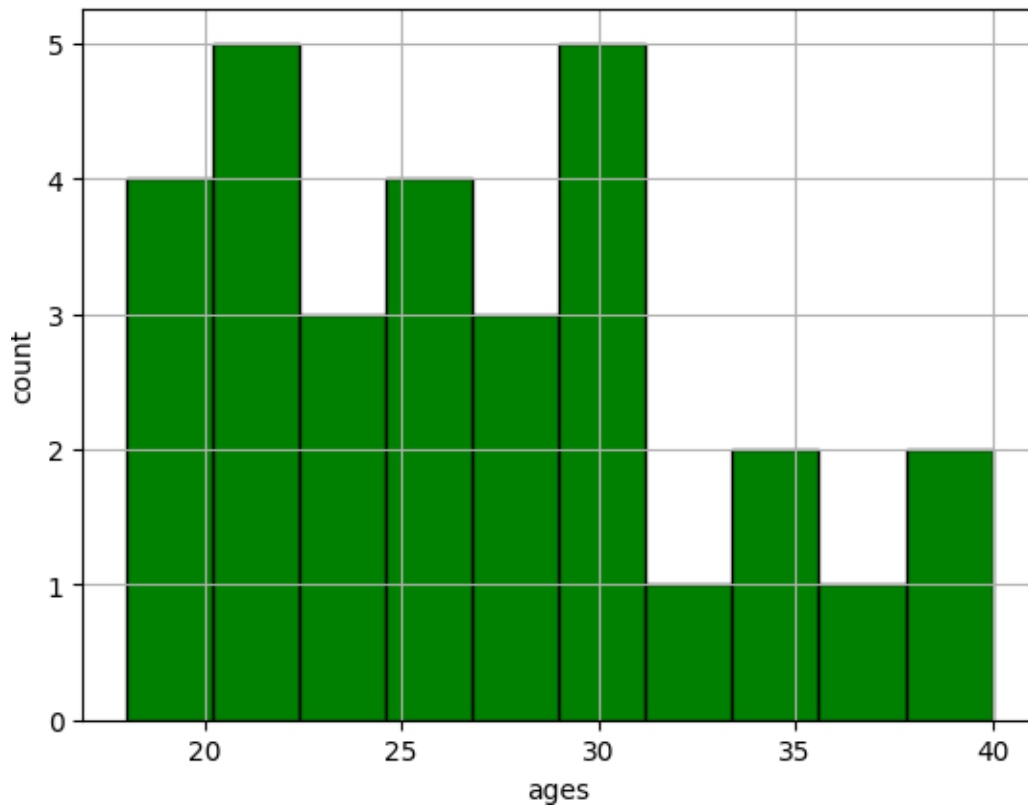
exp1e

November 18, 2025

```
[11]: from matplotlib import pyplot as plt

ages = [18, 19, 20, 20, 21, 21, 22, 22, 22, 23,
        23, 24, 25, 25, 25, 26, 27, 28, 28, 29,
        30, 30, 30, 31, 32, 34, 35, 36, 38, 40]

plt.hist(ages,color='g',bins=10,edgecolor='black')
plt.xlabel("ages")
plt.ylabel("count")
plt.grid(True)
plt.show()
```



[]:

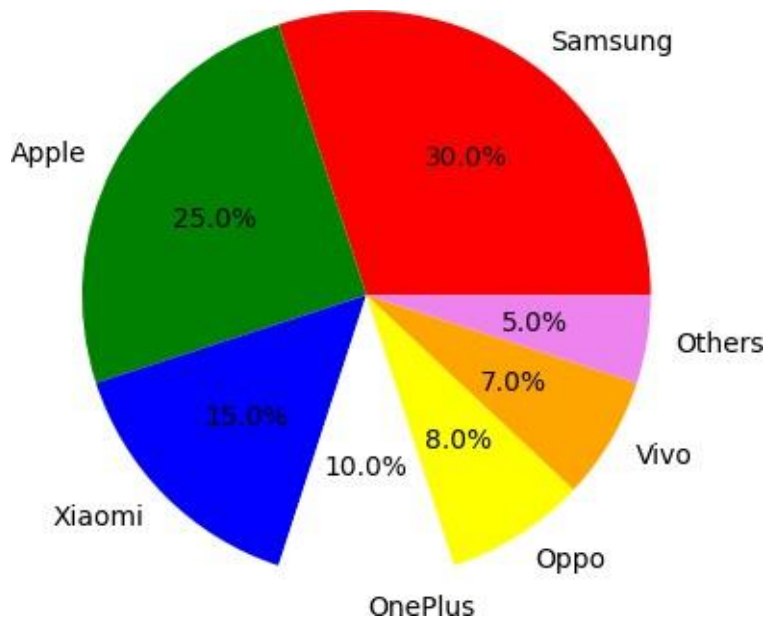
exp1f

November 18, 2025

```
[17]: from matplotlib import pyplot as plt

brands = ['Samsung', 'Apple', 'Xiaomi', 'OnePlus', 'Oppo', 'Vivo', 'Others']
market_share = [30, 25, 15, 10, 8, 7, 5]

plt.pie(market_share, labels=brands, autopct='%0.1f%%', colors=['red', 'green', 'b', 'w', 'yellow', 'orange', 'violet'])
plt.show()
```



[]:

exp2

November 18, 2025

```
[4]: import os
import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

os.chdir("D:\\pandas")

df=pd.read_csv("sales_data.csv")
df.head()
```

```
[4]:
```

	Date	Product	Sales	Quantity	Region
0	01-01-2023	Product A	200	4	North
1	02-01-2023	Product B	150	3	South
2	03-01-2023	Product A	220	5	North
3	04-01-2023	Product C	300	6	East
4	05-01-2023	Product B	180	4	West

```
[7]: df.isnull().sum()
```

```
[7]: Date      0
Product    0
Sales      0
Quantity   0
Region     0
dtype: int64
```

```
[13]: df['Sales'].fillna(df['Sales'].mean())
```

```
[13]: 0    200
1    150
2    220
3    300
4    180
5    210
6    320
```

```

7    160
8    230
9    310
10   190
11   240
12   330
13   170
14   250
15   340

```

Name: Sales, dtype: int64

```
[15]: df.dropna(subset=['Product','Quantity','Region'])
```

```
[15]:
```

	Date	Product	Sales	Quantity	Region
0	01-01-2023	Product A	200	4	North
1	02-01-2023	Product B	150	3	South
2	03-01-2023	Product A	220	5	North
3	04-01-2023	Product C	300	6	East
4	05-01-2023	Product B	180	4	West
5	06-01-2023	Product A	210	5	North
6	07-01-2023	Product C	320	7	East
7	08-01-2023	Product B	160	3	South
8	09-01-2023	Product A	230	6	North
9	10-01-2023	Product C	310	7	East
10	11-01-2023	Product B	190	4	West
11	12-01-2023	Product A	240	6	North
12	13-01-2023	Product C	330	8	East
13	14-01-2023	Product B	170	3	South
14	15-01-2023	Product A	250	7	North
15	16-01-2023	Product C	340	8	East

```
[16]: df.describe()
```

```
[16]:
```

	Sales	Quantity
count	16.000000	16.000000
mean	237.500000	5.375000
std	64.031242	1.746425
min	150.000000	3.000000
25%	187.500000	4.000000
50%	225.000000	5.500000
75%	302.500000	7.000000
max	340.000000	8.000000

```
[18]: product_summary=df.groupby('Product').agg({
        'Sales':'sum',
        'Quantity':'sum'
    }).reset_index()
```

```
product_summary
```

```
[18]:
```

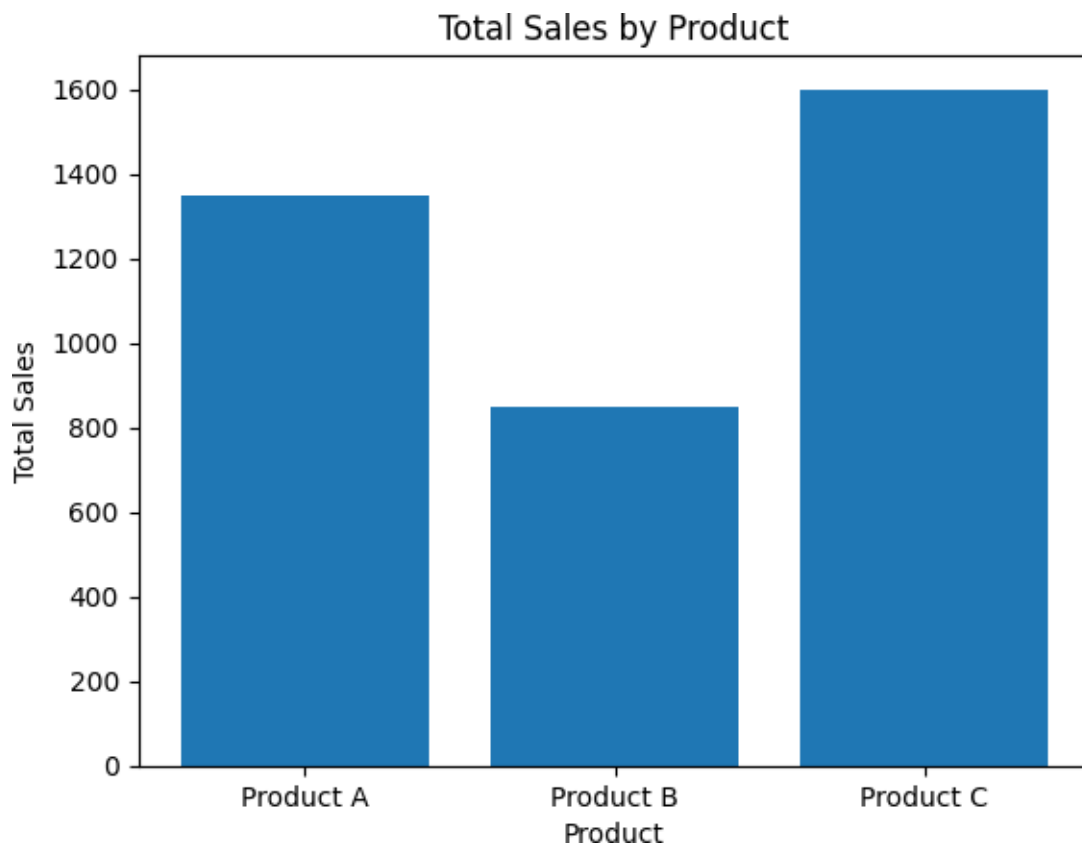
	Product	Sales	Quantity
0	Product A	1350	33
1	Product B	850	17
2	Product C	1600	36

```
[19]: plt.figure(figsize=(10,6))
```

```
[19]: <Figure size 1000x600 with 0 Axes>
```

```
<Figure size 1000x600 with 0 Axes>
```

```
[20]: plt.bar(product_summary['Product'], product_summary['Sales'])  
plt.xlabel('Product')  
plt.ylabel('Total Sales')  
plt.title('Total Sales by Product')  
plt.show()
```



```
[22]: sales_over_time = df.groupby('Date').agg({'Sales': 'sum'}).reset_index()
plt.figure(figsize=(10, 6))
plt.plot(sales_over_time['Date'],sales_over_time['Sales'])
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('SalesOver Time')
plt.show()
```



```
[29]: pivot_table = df.pivot_table(values='Sales', index='Region', columns='Product',
aggfunc='sum',fill_value=0)
```

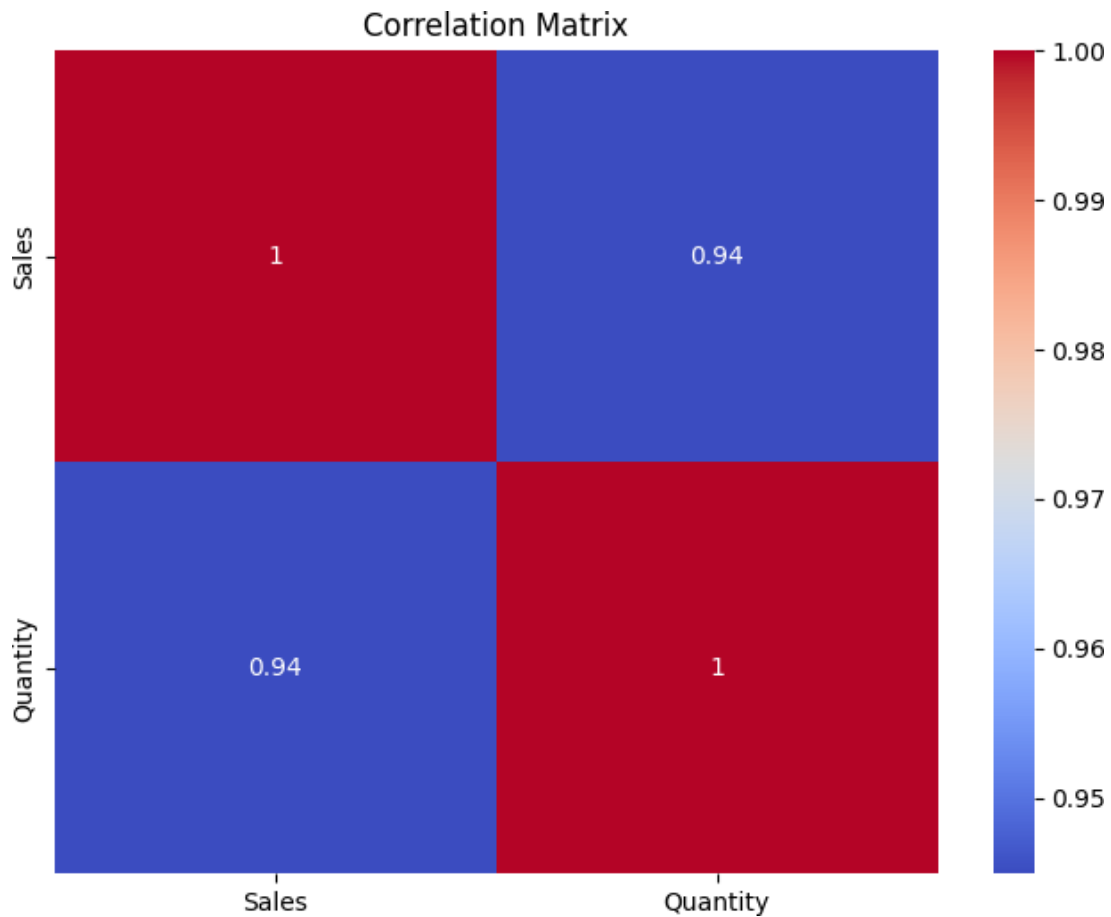
```
pivot_table
```

```
[29]: Product    Product A    Product B    Product C
Region
East           0           0          1600
North        1350           0           0
South           0          480           0
West           0          370           0
```

```
[37]: df['Date'] = pd.to_datetime(df['Date'],dayfirst=True)
correlation_matrix = df.select_dtypes(include=[float, int]).corr()
print(correlation_matrix)
```


	Sales	Quantity
Sales	1.000000	0.944922
Quantity	0.944922	1.000000

```
[38]: import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



```
[ ]:
```


exp3

November 18, 2025

```
[2]: import pandas as pd

import numpy as np

import os

os.chdir("D:\\pandas")

df=pd.read_csv("pre_process_datasample.csv")

df
```

```
[2]:
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[3]: df["Country"].mode()
```

```
[3]: 0    France
      Name: Country, dtype: object
```

```
[4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Country     10 non-null    object
1   Age         9 non-null     float64
```

```
2   Salary      9 non-null    float64
3   Purchased  10 non-null    object
dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

```
[5]: df["Country"].mode()[0]
```

```
[5]: 'France'
```

```
[6]: type(df["Country"].mode()[0])
```

```
[6]: str
```

```
[8]: df["Country"].fillna(df["Country"].mode()[0])
```

```
[8]: 0    France
1    Spain
2    Germany
3    Spain
4    Germany
5    France
6    Spain
7    France
8    Germany
9    France
Name: Country, dtype: object
```

```
[10]: df["Age"].fillna(df["Age"].median())
```

```
[10]: 0    44.0
1    27.0
2    30.0
3    38.0
4    40.0
5    35.0
6    38.0
7    48.0
8    50.0
9    37.0
Name: Age, dtype: float64
```

```
[48]: df["Salary"].fillna(round(df["Salary"].mean()))
```

```
[48]: 0    72000.0
1    48000.0
2    54000.0
3    61000.0
```

```

4    63778.0
5    58000.0
6    52000.0
7    79000.0
8    83000.0
9    67000.0

```

Name: Salary, dtype: float64

```
[49]: pd.get_dummies(df.Country)
```

```

[49]:   France  Germany  Spain
0     True     False  False
1     False     False   True
2     False     True   False
3     False     False   True
4     False     True   False
5     True     False  False
6     False     False   True
7     True     False  False
8     False     True   False
9     True     False  False

```

```

[50]: updated_dataset = pd.concat([pd.get_dummies(df.Country), df.iloc[:, [1, 2, 3]]], axis=1)
      updated_dataset

```

```

[50]:   France  Germany  Spain  Age  Salary  Purchased
0     True     False  False  44.0  72000.0         No
1     False     False   True  27.0  48000.0         Yes
2     False     True   False  30.0  54000.0         No
3     False     False   True  38.0  61000.0         No
4     False     True   False  40.0  63778.0         Yes
5     True     False  False  35.0  58000.0         Yes
6     False     False   True   NaN  52000.0         No
7     True     False  False  48.0  79000.0         Yes
8     False     True   False  50.0  83000.0         No
9     True     False  False  37.0  67000.0         Yes

```

```

[52]: updated_dataset["Purchased"] = updated_dataset["Purchased"].
      .replace(['No','Yes'], [0,1]).astype(int)

```

```
[53]: updated_dataset
```

```

[53]:   France  Germany  Spain  Age  Salary  Purchased
0     True     False  False  44.0  72000.0         0
1     False     False   True  27.0  48000.0         1
2     False     True   False  30.0  54000.0         0

```

3	False	False	True	38.0	61000.0	0
4	False	True	False	40.0	63778.0	1
5	True	False	False	35.0	58000.0	1
6	False	False	True	NaN	52000.0	0
7	True	False	False	48.0	79000.0	1
8	False	True	False	50.0	83000.0	0
9	True	False	False	37.0	67000.0	1

[]:

exp4

November 18, 2025

```
[45]: import numpy as np

import pandas as pd

import os

os.chdir("D:\\pandas")

df=pd.read_csv("Hotel_Dataset.csv")
df
```

```
[45]:
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	\
0	1	20-25	4	Ibis	veg	1300	
1	2	30-35	5	LemonTree	Non-Veg	2000	
2	3	25-30	6	RedFox	Veg	1322	
3	4	20-25	-1	LemonTree	Veg	1234	
4	5	35+	3	Ibis	Vegetarian	989	
5	6	35+	3	Ibys	Non-Veg	1909	
6	7	35+	4	RedFox	Vegetarian	1000	
7	8	20-25	7	LemonTree	Veg	2999	
8	9	25-30	2	Ibis	Non-Veg	3456	
9	9	25-30	2	Ibis	Non-Veg	3456	
10	10	30-35	5	RedFox	non-Veg	-6755	

	NoOfPax	EstimatedSalary	Age_Group.1
0	2	40000	20-25
1	3	59000	30-35
2	2	30000	25-30
3	2	120000	20-25
4	2	45000	35+
5	2	122220	35+
6	-1	21122	35+
7	-10	345673	20-25
8	3	-99999	25-30
9	3	-99999	25-30
10	4	87777	30-35

```
[46]: df.duplicated()
```

```
[46] : 0    False
      1    False
      2    False
      3    False
      4    False
      5    False
      6    False
      7    False
      8    False
      9     True
     10    False
      dtype: bool
```

```
[47] : df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   CustomerID            11 non-null    int64
1   Age_Group              11 non-null    object
2   Rating(1-5)           11 non-null    int64
3   Hotel                  11 non-null    object
4   FoodPreference         11 non-null    object
5   Bill                   11 non-null    int64
6   NoOfPax                11 non-null    int64
7   EstimatedSalary        11 non-null    int64
8   Age_Group.1            11 non-null    object
dtypes: int64(5), object(4)
memory usage: 924.0+ bytes
```

```
[48] : df.drop_duplicates(inplace=True)
      df
```

```
[48]:
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	\
0	1	20-25	4	Ibis	veg	1300	
1	2	30-35	5	LemonTree	Non-Veg	2000	
2	3	25-30	6	RedFox	Veg	1322	
3	4	20-25	-1	LemonTree	Veg	1234	
4	5	35+	3	Ibis	Vegetarian	989	
5	6	35+	3	Ibys	Non-Veg	1909	
6	7	35+	4	RedFox	Vegetarian	1000	
7	8	20-25	7	LemonTree	Veg	2999	
8	9	25-30	2	Ibis	Non-Veg	3456	

10	10	30-35	5	RedFox	non-Veg	-6755
----	----	-------	---	--------	---------	-------

	NoOfPax	EstimatedSalary	Age_Group.1
0	2	40000	20-25
1	3	59000	30-35
2	2	30000	25-30
3	2	120000	20-25
4	2	45000	35+
5	2	122220	35+
6	-1	21122	35+
7	-10	345673	20-25
8	3	-99999	25-30
10	4	87777	30-35

```
[49]: len(df)
```

```
[49]: 10
```

```
[50]: index=np.array(list(range(0,len(df))))
df.set_index(index)
```

```
[50]:
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	\
0	1	20-25	4	Ibis	veg	1300	2	
1	2	30-35	5	LemonTree	Non-Veg	2000	3	
2	3	25-30	6	RedFox	Veg	1322	2	
3	4	20-25	-1	LemonTree	Veg	1234	2	
4	5	35+	3	Ibis	Vegetarian	989	2	
5	6	35+	3	Ibys	Non-Veg	1909	2	
6	7	35+	4	RedFox	Vegetarian	1000	-1	
7	8	20-25	7	LemonTree	Veg	2999	-10	
8	9	25-30	2	Ibis	Non-Veg	3456	3	
9	10	30-35	5	RedFox	non-Veg	-6755	4	

	EstimatedSalary	Age_Group.1
0	40000	20-25
1	59000	30-35
2	30000	25-30
3	120000	20-25
4	45000	35+
5	122220	35+
6	21122	35+
7	345673	20-25
8	-99999	25-30
9	87777	30-35

```
[51]: df=df.drop(['Age_Group.1'],axis=1)
df
```

```
[51]: CustomerID Age_Group Rating(1-5) Hotel FoodPreference Bill \
0 1 20-25 4 Ibis veg 1300
1 2 30-35 5 LemonTree Non-Veg 2000
2 3 25-30 6 RedFox Veg 1322
3 4 20-25 -1 LemonTree Veg 1234
4 5 35+ 3 Ibis Vegetarian 989
5 6 35+ 3 Ibys Non-Veg 1909
6 7 35+ 4 RedFox Vegetarian 1000
7 8 20-25 7 LemonTree Veg 2999
8 9 25-30 2 Ibis Non-Veg 3456
10 10 30-35 5 RedFox non-Veg -6755
```

```
NoOfPax EstimatedSalary
0 2 40000
1 3 59000
2 2 30000
3 2 120000
4 2 45000
5 2 122220
6 -1 21122
7 -10 345673
8 3 -99999
10 4 87777
```

```
[52]: df.loc[df["CustomerID"] < 0, "CustomerID"] = np.nan
df.loc[df["Bill"] < 0, "Bill"] = np.nan
df.loc[df["EstimatedSalary"] < 0, "EstimatedSalary"] = np.nan
df.loc[df["Rating(1-5)"] < 0, "Rating(1-5)"] = np.nan
```

```
df
```

```
[52]: CustomerID Age_Group Rating(1-5) Hotel FoodPreference Bill \
0 1.0 20-25 4.0 Ibis veg 1300.0
1 2.0 30-35 5.0 LemonTree Non-Veg 2000.0
2 3.0 25-30 6.0 RedFox Veg 1322.0
3 4.0 20-25 NaN LemonTree Veg 1234.0
4 5.0 35+ 3.0 Ibis Vegetarian 989.0
5 6.0 35+ 3.0 Ibys Non-Veg 1909.0
6 7.0 35+ 4.0 RedFox Vegetarian 1000.0
7 8.0 20-25 7.0 LemonTree Veg 2999.0
8 9.0 25-30 2.0 Ibis Non-Veg 3456.0
10 10.0 30-35 5.0 RedFox non-Veg NaN
```

	NoOfPax	EstimatedSalary
0	2	40000.0
1	3	59000.0
2	2	30000.0
3	2	120000.0
4	2	45000.0
5	2	122220.0
6	-1	21122.0
7	-10	345673.0
8	3	NaN
10	4	87777.0

```
[53]: df.Age_Group.unique()
```

```
[53]: array(['20-25', '30-35', '25-30', '35+'], dtype=object)
```

```
[54]: df["Hotel"].unique()
```

```
[54]: array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'], dtype=object)
```

```
[55]: df.Hotel.replace(['Ibys'],'Ibis')
```

```
[55]: 0      Ibis
      1  LemonTree
      2    RedFox
      3  LemonTree
      4      Ibis
      5      Ibis
      6    RedFox
      7  LemonTree
      8      Ibis
     10    RedFox
      Name: Hotel, dtype: object
```

```
[56]: df.FoodPreference.replace(['Vegetarian','veg'],'Veg')
```

```
[56]: 0      Veg
      1 Non-Veg
      2      Veg
      3      Veg
      4      Veg
      5 Non-Veg
      6      Veg
      7      Veg
      8 Non-Veg
     10 non-Veg
      Name: FoodPreference, dtype: object
```

```
[57]: df.FoodPreference.replace(['non-Veg'],'Non-Veg')
```

```
[57]: 0      veg
      1    Non-Veg
      2      Veg
      3      Veg
      4  Vegetarian
      5    Non-Veg
      6  Vegetarian
      7      Veg
      8    Non-Veg
     10    Non-Veg
      Name: FoodPreference, dtype: object
```

```
[58]: df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()))
```

```
[58]: 0      40000.0
      1      59000.0
      2      30000.0
      3     120000.0
      4      45000.0
      5     122220.0
      6      21122.0
      7     345673.0
      8      96755.0
     10      87777.0
      Name: EstimatedSalary, dtype: float64
```

```
[59]: df.NoOfPax.fillna(round(df.NoOfPax.median()))
      df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()))
      df.Bill.fillna(round(df.Bill.mean()))
      df
```

```
[59]: CustomerID Age_Group Rating(1-5) Hotel FoodPreference Bill \
0      1.0      20-25      4.0      Ibis      veg 1300.0
1      2.0      30-35      5.0  LemonTree    Non-Veg 2000.0
2      3.0      25-30      6.0      RedFox      Veg 1322.0
3      4.0      20-25      NaN  LemonTree      Veg 1234.0
4      5.0       35+      3.0      Ibis  Vegetarian  989.0
5      6.0       35+      3.0      Ibys    Non-Veg 1909.0
6      7.0       35+      4.0      RedFox  Vegetarian 1000.0
7      8.0      20-25      7.0  LemonTree      Veg 2999.0
8      9.0      25-30      2.0      Ibis    Non-Veg 3456.0
10     10.0      30-35      5.0      RedFox    non-Veg    NaN

      NoOfPax EstimatedSalary
0           2      40000.0
```

1	3	59000.0
2	2	30000.0
3	2	120000.0
4	2	45000.0
5	2	122220.0
6	-1	21122.0
7	-10	345673.0
8	3	NaN
10	4	87777.0

[]:

exp6

November 18, 2025

```
[1]: import numpy as np
```

```
[2]: import pandas as pd
```

```
[7]: import os
```

```
os.chdir("D:\\pandas")  
df=pd.read_csv("pre_process_datasample.csv")  
df
```

```
[7]:
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[5]: df.head()
```

```
[5]:
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```
[6]: df.Country.fillna(df.Country.mode()[0])
```

```
[6]:
```

0	France
1	Spain
2	Germany
3	Spain

```
4    Germany
5    France
6    Spain
7    France
8    Germany
9    France
```

Name: Country, dtype: object

```
[8]: features=df.iloc[:, :-1].values
```

```
[ ]: label=df.iloc[:, -1].values
```

```
[10]: from sklearn.impute import SimpleImputer
      age=SimpleImputer(strategy="mean",missing_values=np.nan)
      Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
```

```
[11]: age.fit(features[:,[1]])
```

```
[11]: SimpleImputer()
```

```
[12]: Salary.fit(features[:,[2]])
```

```
[12]: SimpleImputer()
```

```
[13]: SimpleImputer()
```

```
[13]: SimpleImputer()
```

```
[14]: features[:,[1]]=age.transform(features[:,[1]])
      features[:,[2]]=Salary.transform(features[:,[2]])
      features
```

```
[14]: array([['France', 44.0, 72000.0],
             ['Spain', 27.0, 48000.0],
             ['Germany', 30.0, 54000.0],
             ['Spain', 38.0, 61000.0],
             ['Germany', 40.0, 63777.77777777778],
             ['France', 35.0, 58000.0],
             ['Spain', 38.77777777777778, 52000.0],
             ['France', 48.0, 79000.0],
             ['Germany', 50.0, 83000.0],
             ['France', 37.0, 67000.0]], dtype=object)
```

```
[15]: from sklearn.preprocessing import OneHotEncoder
      oh = OneHotEncoder(sparse_output=False)
```



```
Country=oh.fit_transform(features[:,[0]])
Country
```

```
[15]: array([[1., 0., 0.],
            [0., 0., 1.],
            [0., 1., 0.],
            [0., 0., 1.],
            [0., 1., 0.],
            [1., 0., 0.],
            [0., 0., 1.],
            [1., 0., 0.],
            [0., 1., 0.],
            [1., 0., 0.]])
```

```
[16] : final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
final_set
```

```
[16]: array([[1.0, 0.0, 0.0, 44.0, 72000.0],
            [0.0, 0.0, 1.0, 27.0, 48000.0],
            [0.0, 1.0, 0.0, 30.0, 54000.0],
            [0.0, 0.0, 1.0, 38.0, 61000.0],
            [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
            [1.0, 0.0, 0.0, 35.0, 58000.0],
            [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
            [1.0, 0.0, 0.0, 48.0, 79000.0],
            [0.0, 1.0, 0.0, 50.0, 83000.0],
            [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

```
[17] : from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(final_set)
feat_standard_scaler=sc.transform(final_set)
feat_standard_scaler
```

```
[17]: array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
            7.58874362e-01, 7.49473254e-01],
            [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
            -1.71150388e+00, -1.43817841e+00],
            [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
            -1.27555478e+00, -8.91265492e-01],
            [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
            -1.13023841e-01, -2.53200424e-01],
            [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
            1.77608893e-01, 6.63219199e-16],
            [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
            -5.48972942e-01, -5.26656882e-01],
            [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
```

```

0.00000000e+00, -1.07356980e+00],
[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
 1.34013983e+00, 1.38753832e+00],
[-8.16496581e-01, 1.52752523e+00, -6.54653671e-01,
 1.63077256e+00, 1.75214693e+00],
[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
-2.58340208e-01, 2.93712492e-01]])

```

```

[18]: from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler(feature_range=(0,1))
mms.fit(final_set)
feat_minmax_scaler=mms.transform(final_set)
feat_minmax_scaler

```

```

[18]: array([[1.          , 0.          , 0.          , 0.73913043, 0.68571429],
 [0.          , 0.          , 1.          , 0.          , 0.          ],
 [0.          , 1.          , 0.          , 0.13043478, 0.17142857],
 [0.          , 0.          , 1.          , 0.47826087, 0.37142857],
 [0.          , 1.          , 0.          , 0.56521739, 0.45079365],
 [1.          , 0.          , 0.          , 0.34782609, 0.28571429],
 [0.          , 0.          , 1.          , 0.51207729, 0.11428571],
 [1.          , 0.          , 0.          , 0.91304348, 0.88571429],
 [0.          , 1.          , 0.          , 1.          , 1.          ],
 [1.          , 0.          , 0.          , 0.43478261, 0.54285714]])

```

```

[ ]:

```

exp7

November 18, 2025

```
[1]: import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
tips=sns.load_dataset('tips')
```

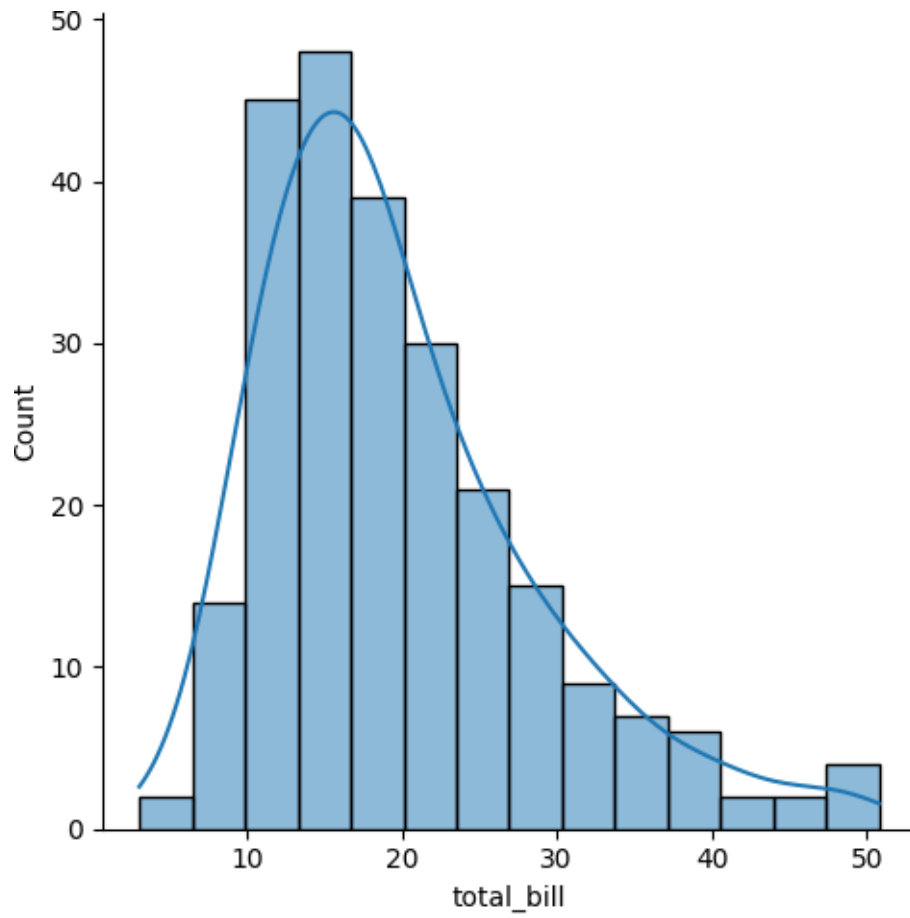
```
[2]: tips.head()
```

```
[2]:
```

	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

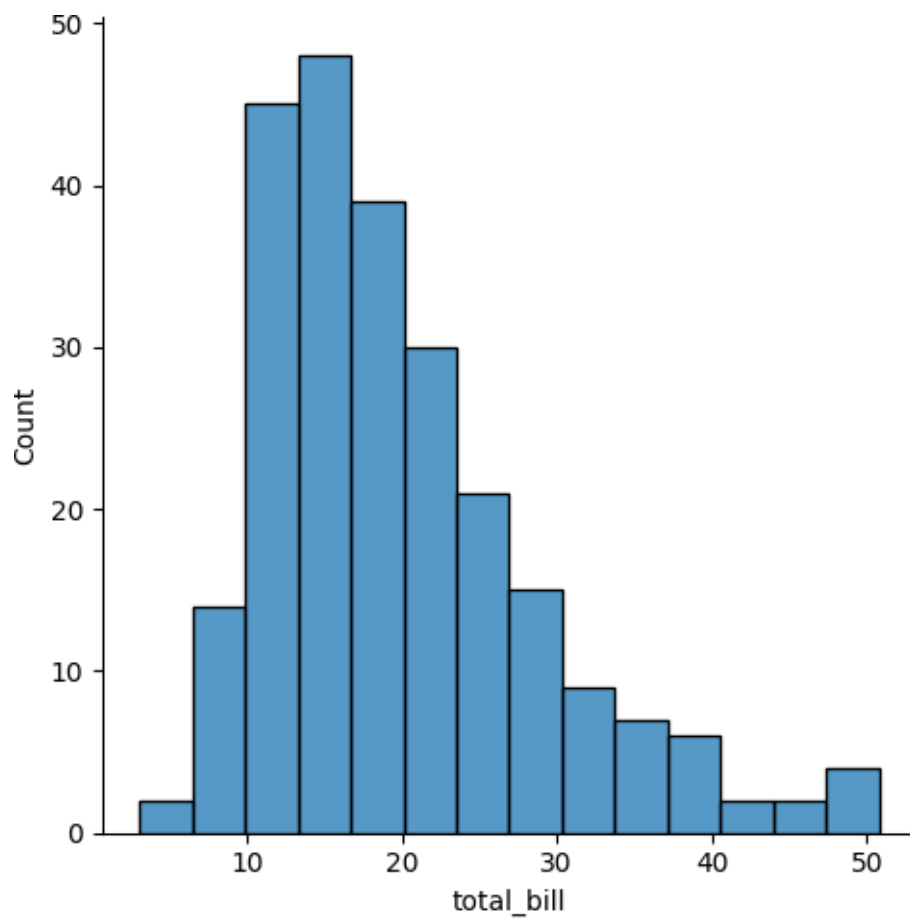
```
[3]: sns.displot(tips.total_bill,kde=True)
```

```
[3]: <seaborn.axisgrid.FacetGrid at 0x201fdd6a660>
```



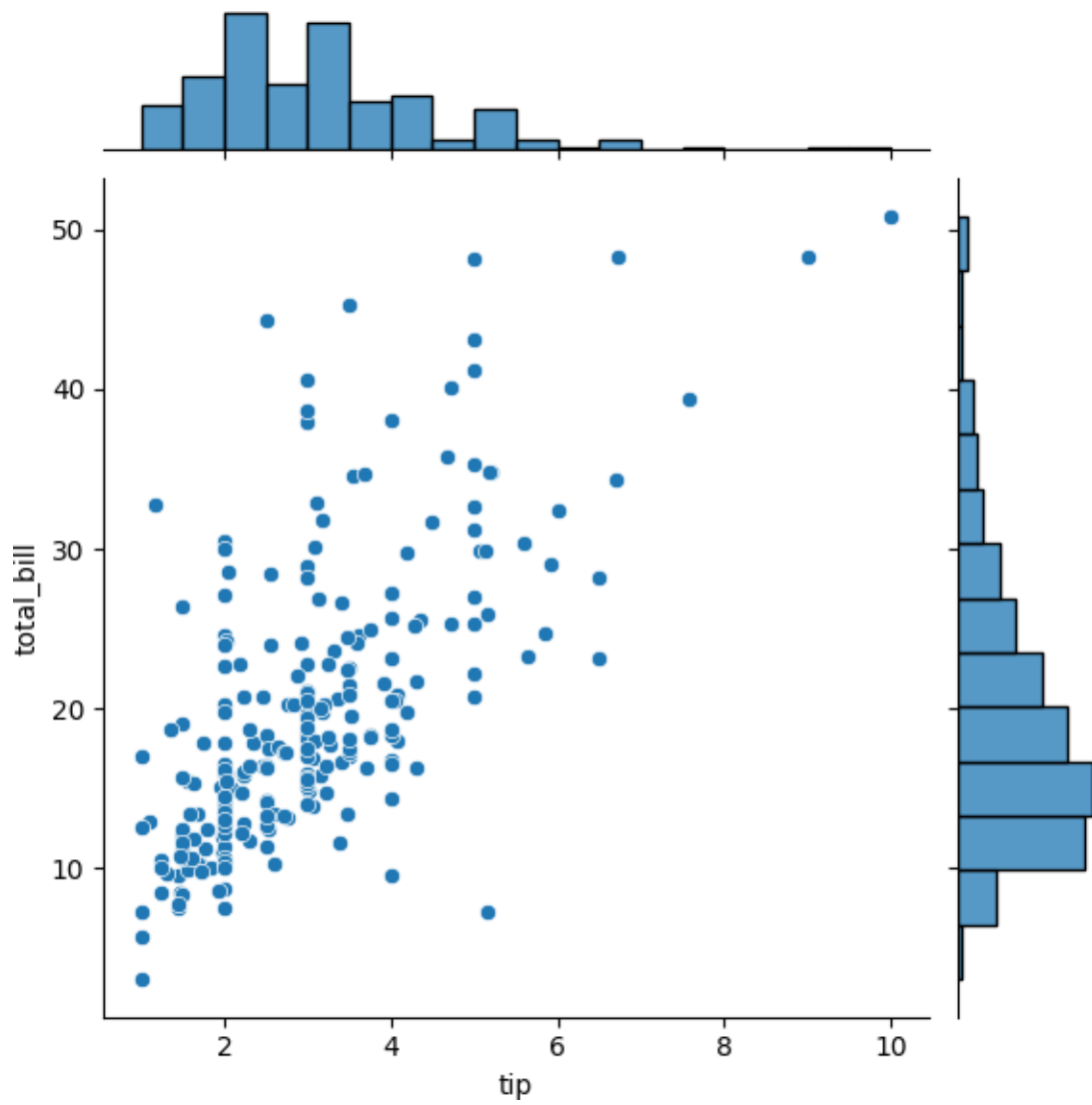
```
[4]: sns.displot(tips.total_bill,kde=False)
```

```
[4]: <seaborn.axisgrid.FacetGrid at 0x201fdf4b250>
```



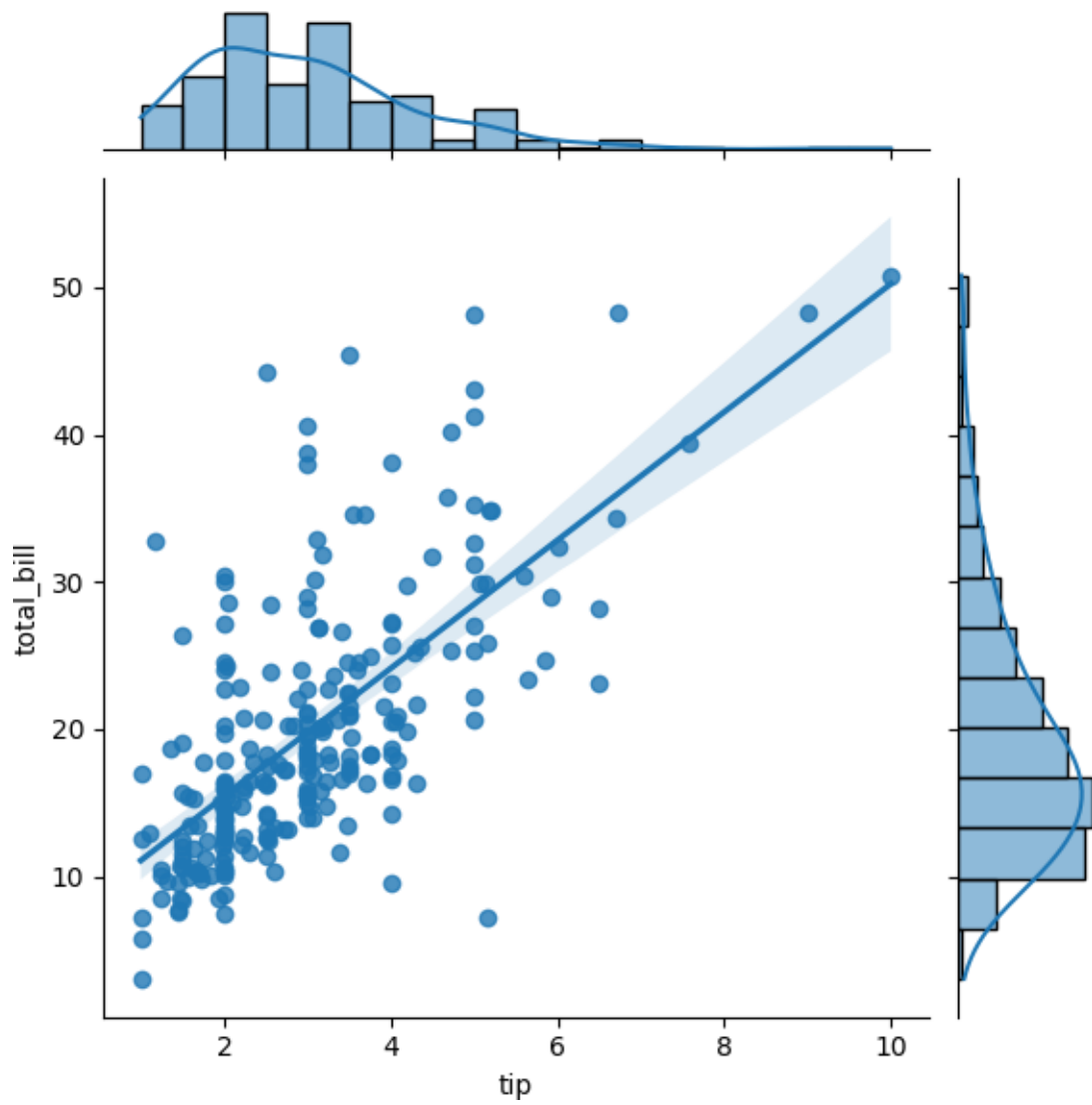
```
[5]: sns.jointplot(x=tips.tip,y=tips.total_bill)
```

```
[5]: <seaborn.axisgrid.JointGrid at 0x201fdde9fd0>
```



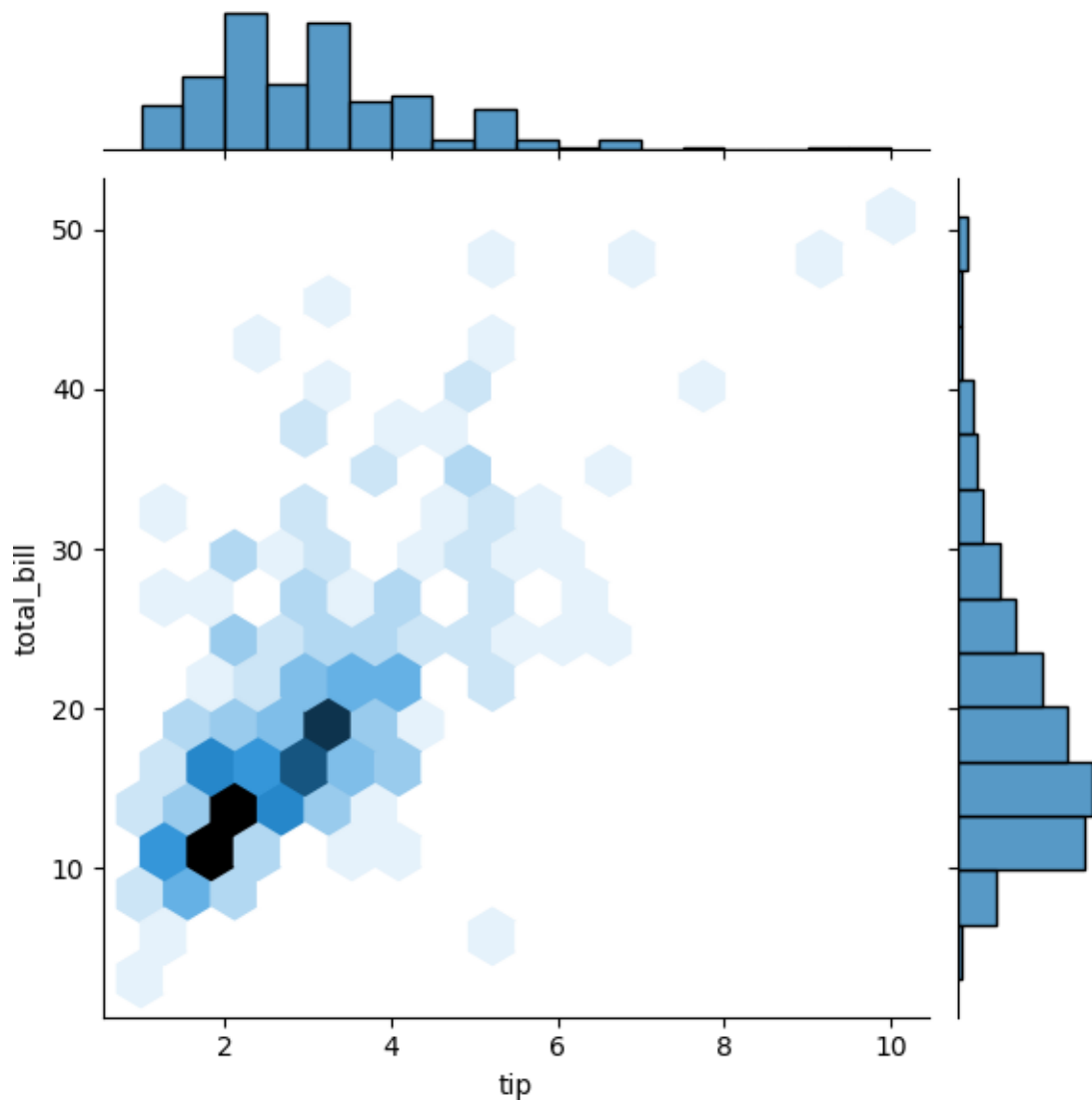
```
[6]: sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
```

```
[6]: <seaborn.axisgrid.JointGrid at 0x201ff15c550>
```



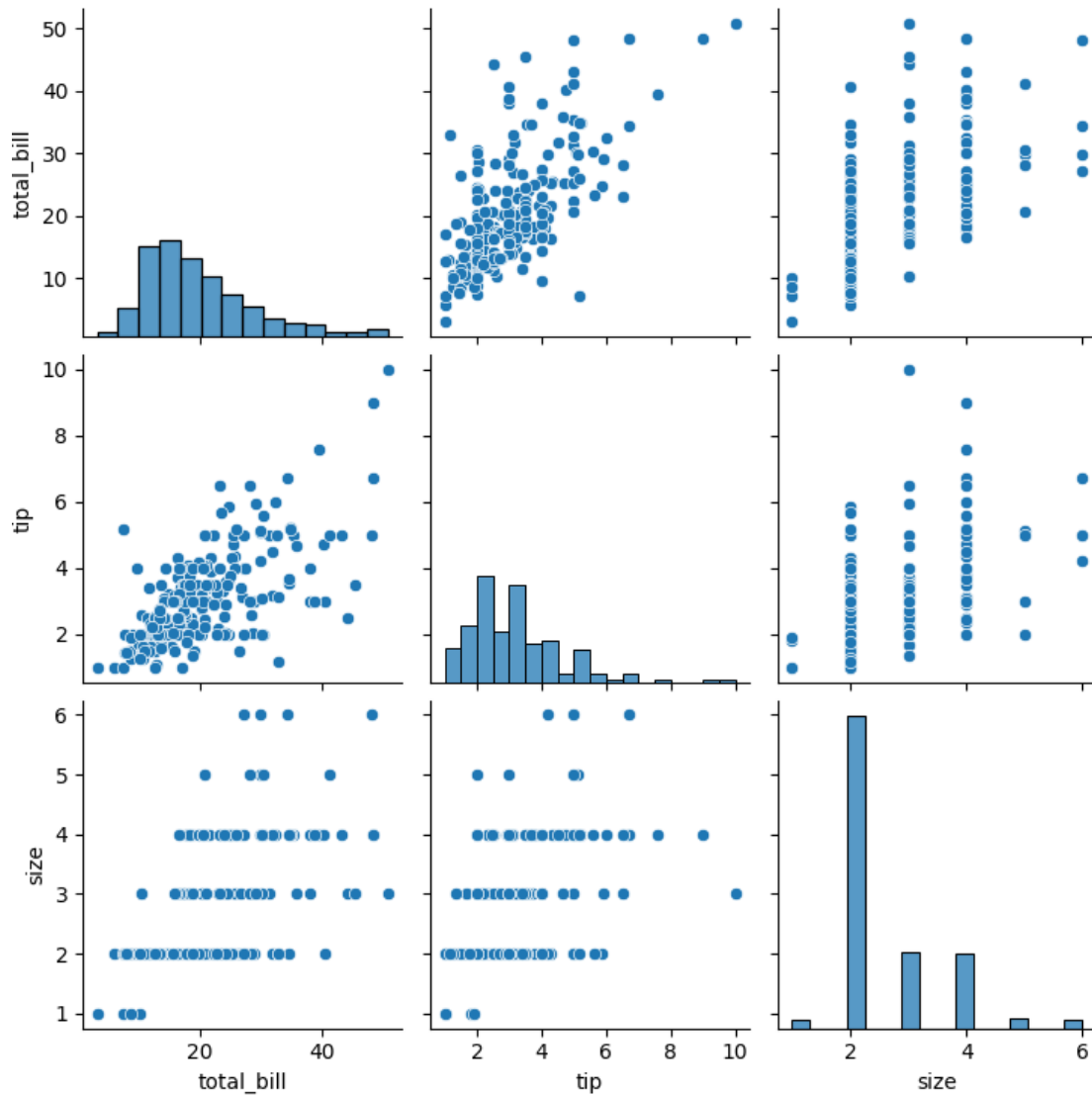
```
[7]: sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
```

```
[7]: <seaborn.axisgrid.JointGrid at 0x201ff56da90>
```



```
[8]: sns.pairplot(tips)
```

```
[8]: <seaborn.axisgrid.PairGrid at 0x201fddeaf90>
```

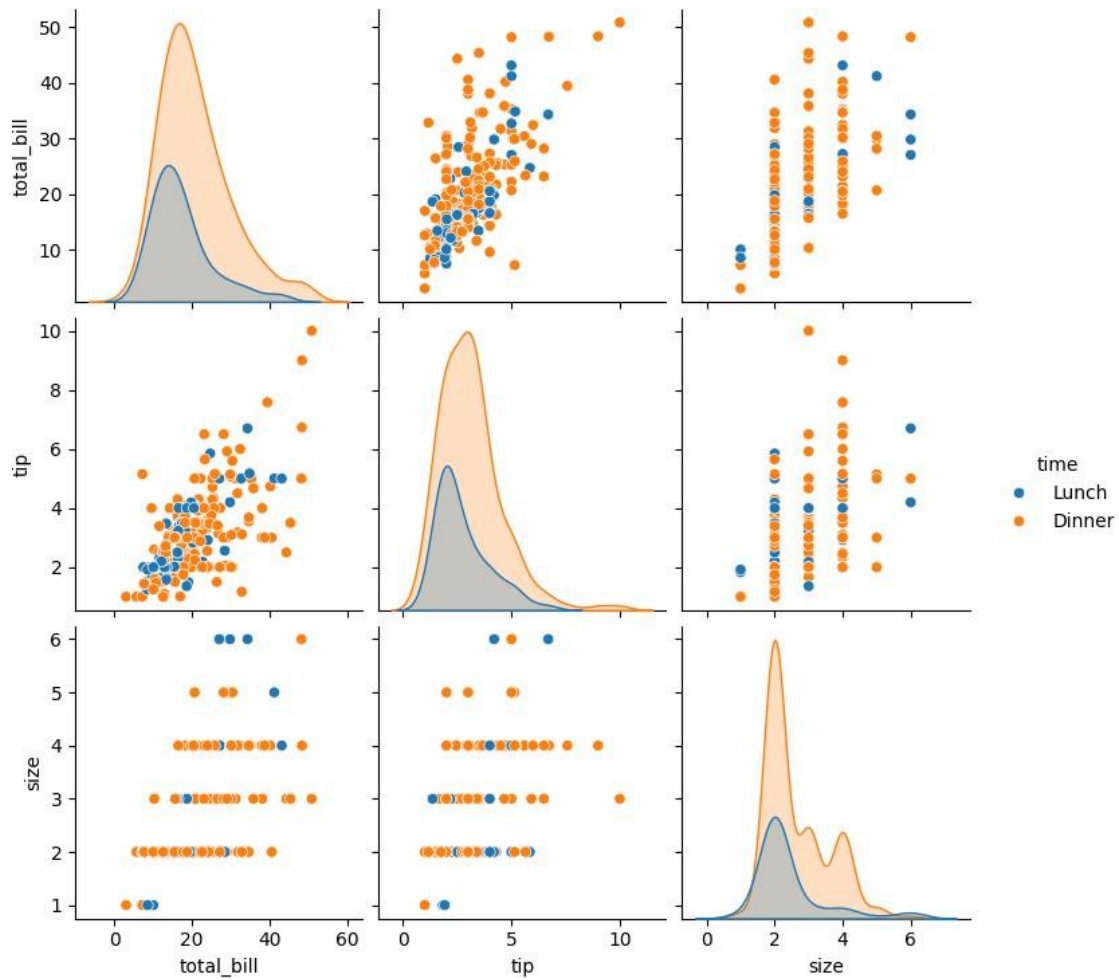



```
[9]: tips.time.value_counts()
```

```
[9]: time
Dinner    176
Lunch      68
Name: count, dtype: int64
```

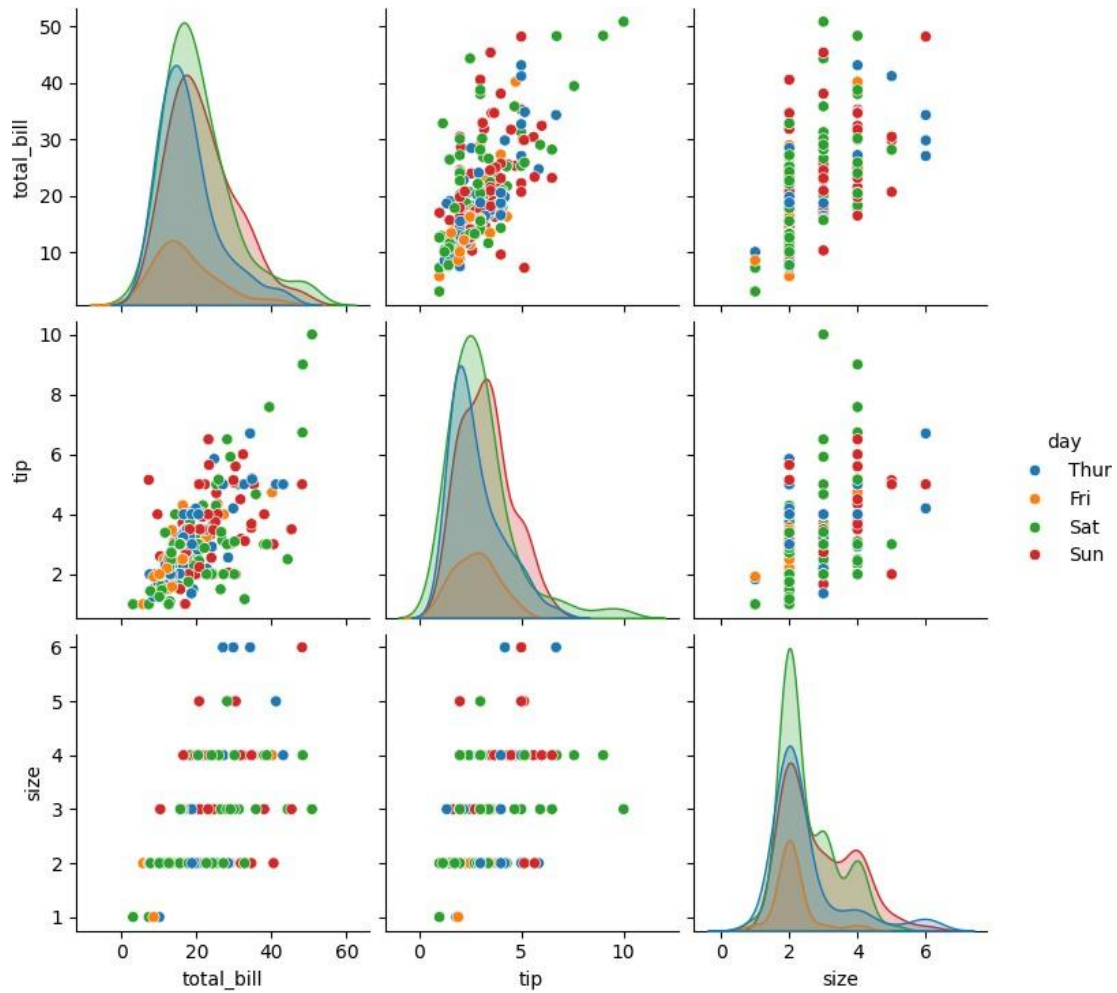
```
[10]: sns.pairplot(tips,hue='time')
```

```
[10]: <seaborn.axisgrid.PairGrid at 0x201843e9310>
```



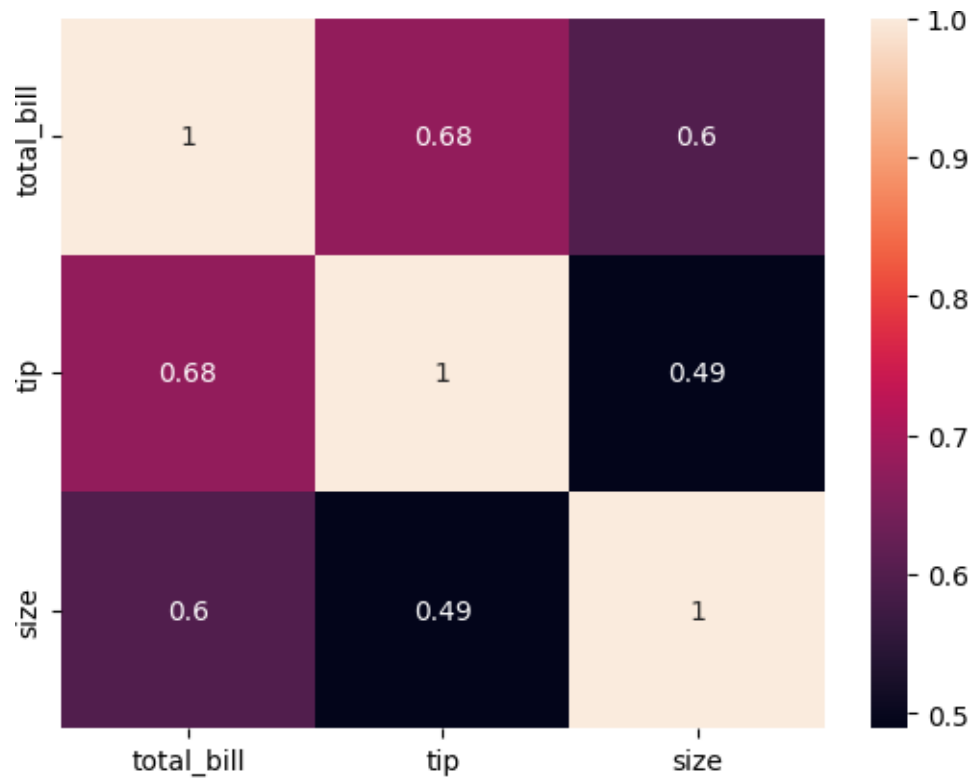
```
[11]: sns.pairplot(tips,hue='day')
```

```
[11]: <seaborn.axisgrid.PairGrid at 0x20184cbd950>
```



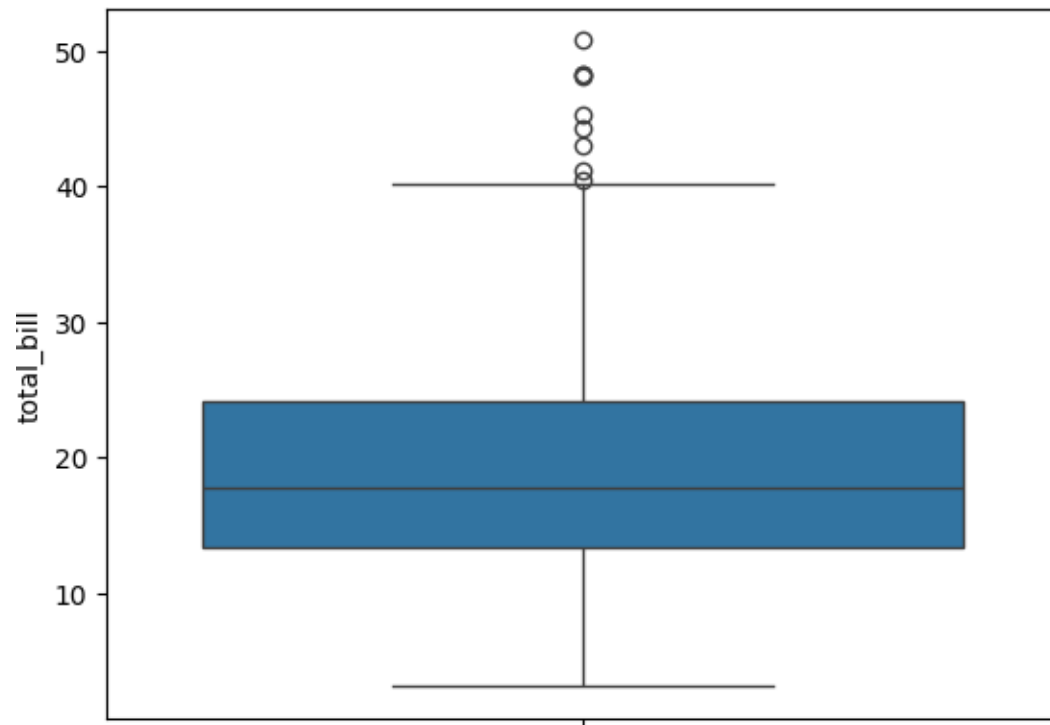
```
[12]: sns.heatmap(tips.corr(numeric_only=True),annot=True)
```

```
[12]: <Axes: >
```



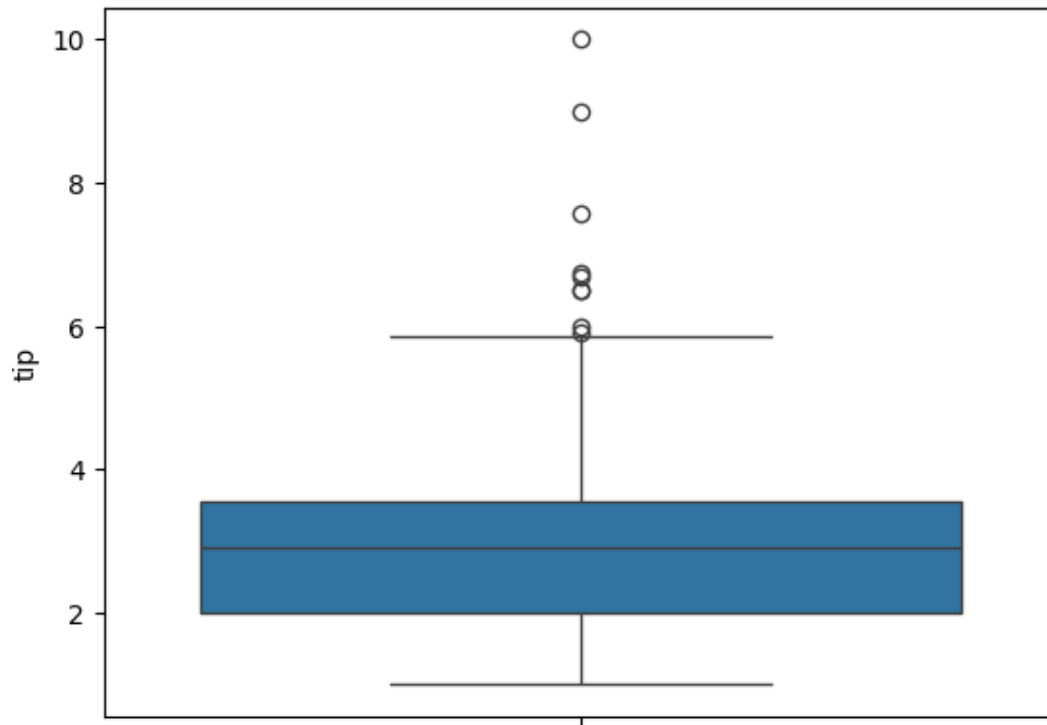
```
[13]: sns.boxplot(tips.total_bill)
```

```
[13]: <Axes: ylabel='total_bill'>
```



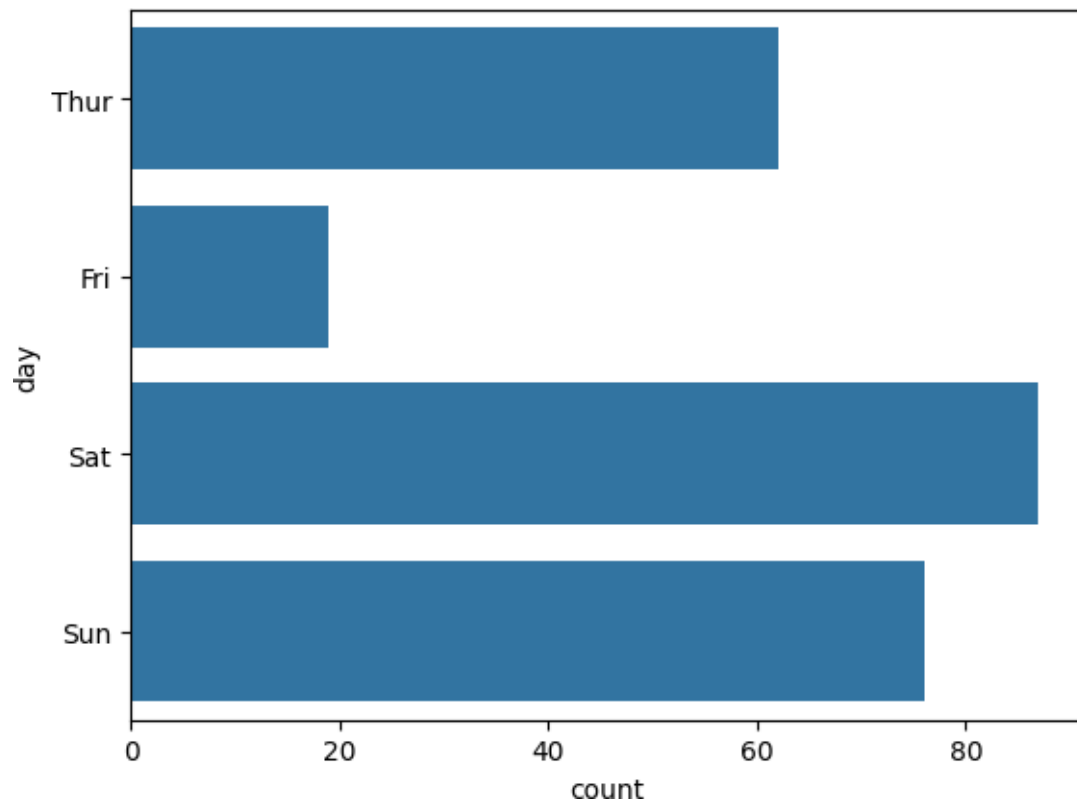
```
[14]: sns.boxplot(tips.tip)
```

```
[14]: <Axes: ylabel='tip'>
```



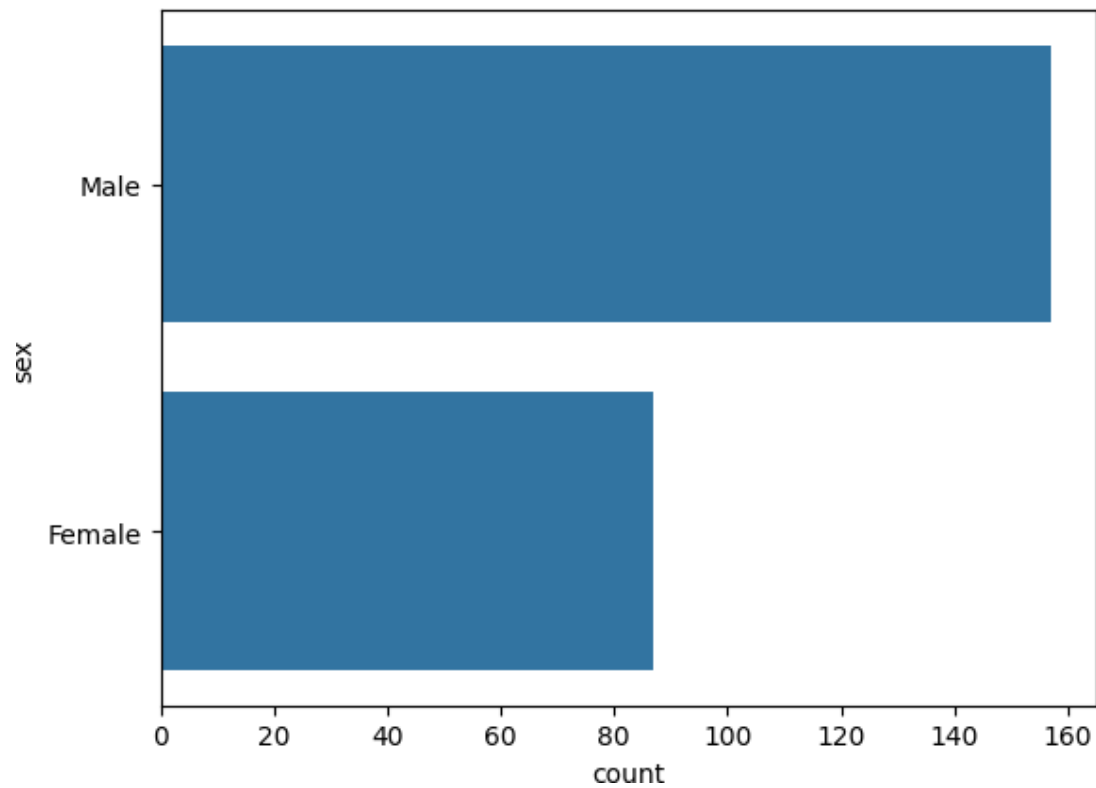
```
[15]: sns.countplot(tips.day)
```

```
[15]: <Axes: xlabel='count', ylabel='day'>
```



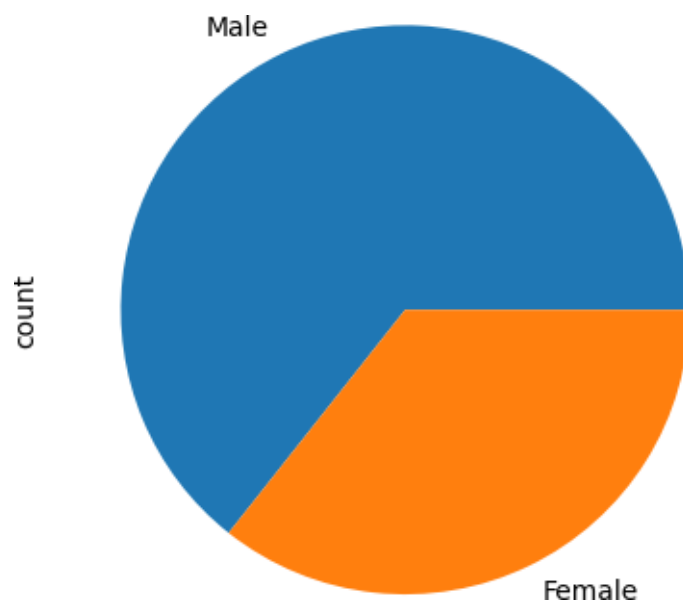
```
[16]: sns.countplot(tips.sex)
```

```
[16]: <Axes: xlabel='count', ylabel='sex'>
```



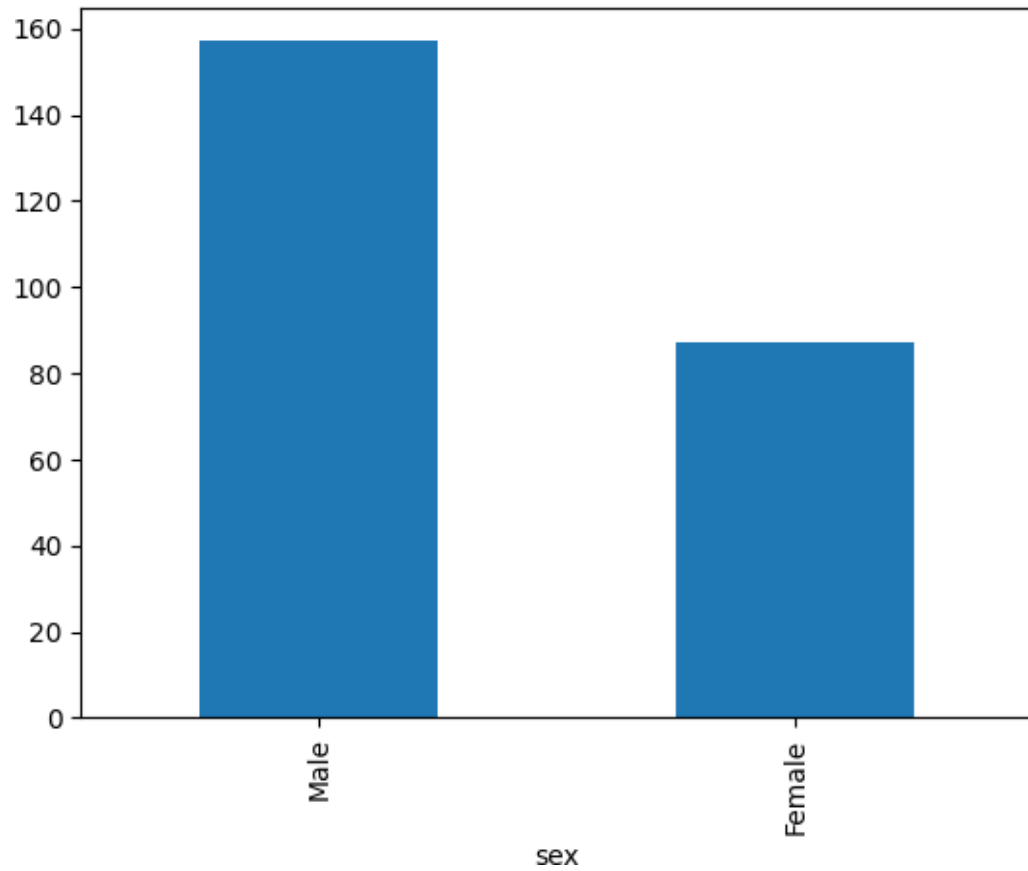
```
[17]: tips.sex.value_counts().plot(kind='pie')
```

```
[17]: <Axes: ylabel='count'>
```

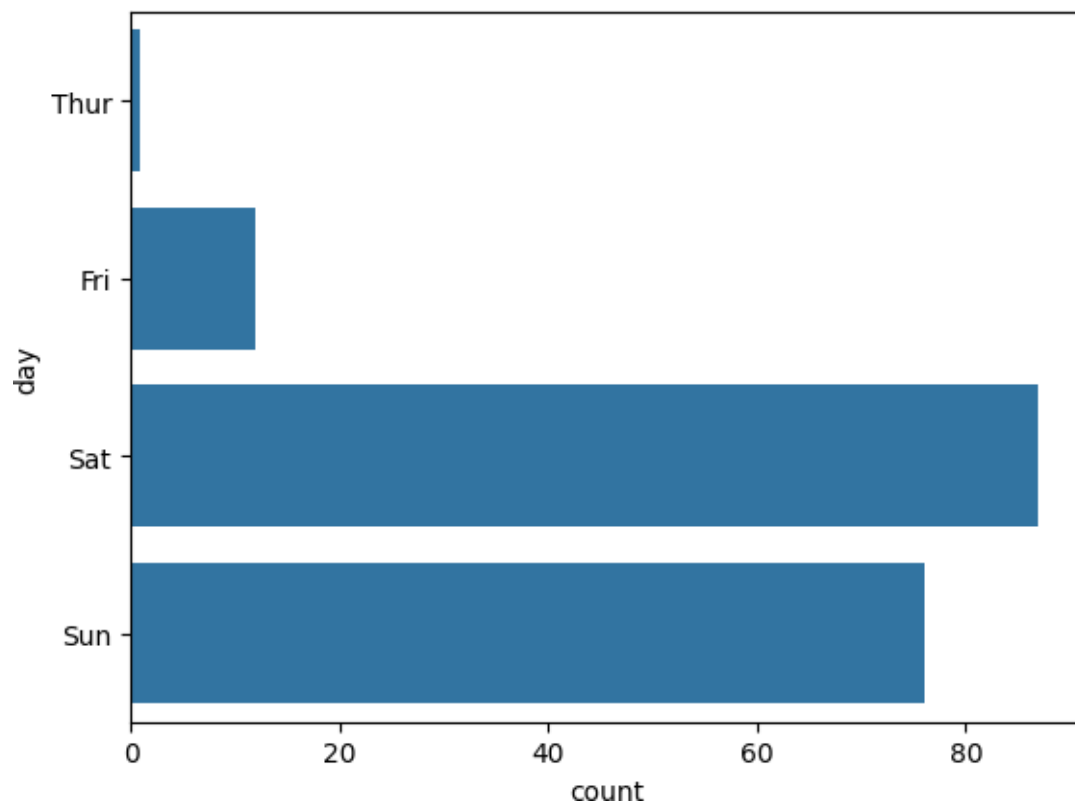
```
[18]: tips.sex.value_counts().plot(kind='bar')
```

```
[18]: <Axes: xlabel='sex'>
```



```
[19]: sns.countplot(tips[tips.time=='Dinner']['day'])
```

```
[19]: <Axes: xlabel='count', ylabel='day'>
```



[]:

```
In [ ]: import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data.csv')
df
```

```
In [19]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
#   Column             Non-Null Count  Dtype  
---  -
0   YearsExperience     30 non-null    float64
1   Salary              30 non-null    int64   
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

```
In [3]: df.dropna(inplace=True)
```

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
#   Column             Non-Null Count  Dtype  
---  -
0   YearsExperience     30 non-null    float64
1   Salary              30 non-null    int64   
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

```
In [5]: df.describe()
```

```
Out[5]:
```

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	58720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

```
In [6]: features=df.iloc[:,[0]].values
label=df.iloc[:,[1]].values
```

```
In [7]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,random_st
```

```
In [20]: from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
```

```
In [21]: model.score(x_train,y_train)
```

```
Out[21]: 0.9603182547438908
```

```
In [23]: model.score(x_test,y_test)
```

```
Out[23]: 0.9184170849214232
```

```
In [24]: model.coef_
```

```
Out[24]: array([[9281.30847068]])
```

```
In [25]: model.intercept_
```

```
Out[25]: array([27166.73682891])
```

```
In [26]: import pickle  
pickle.dump(model,open('SalaryPred.model','wb'))
```

```
In [27]: model=pickle.load(open('SalaryPred.model','rb'))
```

```
In [28]: yr_of_exp=float(input("Enter Years of Experience: "))  
yr_of_exp_NP=np.array([[yr_of_exp]])  
Salary=model.predict(yr_of_exp_NP)  
  
Enter Years of Experience: 44
```

```
In [ ]:
```

```
In [29]: print("Estimated Salary for {} years of experience is {}:".format(yr_of_exp,Salary))  
  
Estimated Salary for 44.0 years of experience is [[435544.30953887]]:
```

```
In [ ]:
```

```
In [1]: import numpy as np
import pandas as pd
df=pd.read_csv('Social_Network_Ads.csv')
df
```

```
Out[1]:
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

```
In [2]: df.head()
```

```
Out[2]:
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
In [4]: features=df.iloc[:,[2,3]].values
label=df.iloc[:,4].values
features
```

```
Out[4]: array([[ 19, 19000],  
               [ 35, 20000],  
               [ 26, 43000],  
               [ 27, 57000],  
               [ 19, 76000],  
               [ 27, 58000],  
               [ 27, 84000],  
               [ 32, 150000],  
               [ 25, 33000],  
               [ 35, 65000],  
               [ 26, 80000],  
               [ 26, 52000],  
               [ 20, 86000],  
               [ 32, 18000],  
               [ 18, 82000],  
               [ 29, 80000],  
               [ 47, 25000],  
               [ 45, 26000],  
               [ 46, 28000],  
               [ 10, 22000])
```

```
In [5]: label
```

```
Out[5]: array([[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1, 1, 0, 1], dtype=int64)
```

```
In [6]: from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
```



```
In [7]: for i in range(1,401):
        x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,
        model=LogisticRegression()
        model.fit(x_train,y_train)
        train_score=model.score(x_train,y_train)
        test_score=model.score(x_test,y_test)
        if test_score>train_score:
            print("Test {} Train{} Random State {}".format(test_score,train_score,i))
```

```
Test 0.6875 Train0.63125 Random State 3
Test 0.7375 Train0.61875 Random State 4
Test 0.6625 Train0.6375 Random State 5
Test 0.65 Train0.640625 Random State 6
Test 0.675 Train0.634375 Random State 7
Test 0.675 Train0.634375 Random State 8
Test 0.65 Train0.640625 Random State 10
Test 0.6625 Train0.6375 Random State 11
Test 0.7125 Train0.625 Random State 13
Test 0.675 Train0.634375 Random State 16
Test 0.7 Train0.628125 Random State 17
Test 0.7 Train0.628125 Random State 21
Test 0.65 Train0.640625 Random State 24
Test 0.6625 Train0.6375 Random State 25
Test 0.75 Train0.615625 Random State 26
Test 0.675 Train0.634375 Random State 27
Test 0.7 Train0.628125 Random State 28
Test 0.6875 Train0.63125 Random State 29
Test 0.6875 Train0.63125 Random State 31
```

```
In [8]: x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,
        finalModel=LogisticRegression()
        finalModel.fit(x_train,y_train)
```

Out[8]: LogisticRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [9]: print(finalModel.score(x_train,y_train))
        print(finalModel.score(x_test,y_test))
```

```
0.834375
0.9125
```

```
In [10]: from sklearn.metrics import classification_report
        print(classification_report(label,finalModel.predict(features)))
```

	precision	recall	f1-score	support
0	0.85	0.93	0.89	257
1	0.84	0.71	0.77	143


```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
In [2]: df=pd.read_csv('Mall_Customers.csv')
```

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   CustomerID                           200 non-null   int64
1   Gender                               200 non-null   object
2   Age                                   200 non-null   int64
3   Annual Income (k$)                   200 non-null   int64
4   Spending Score (1-100)                200 non-null   int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

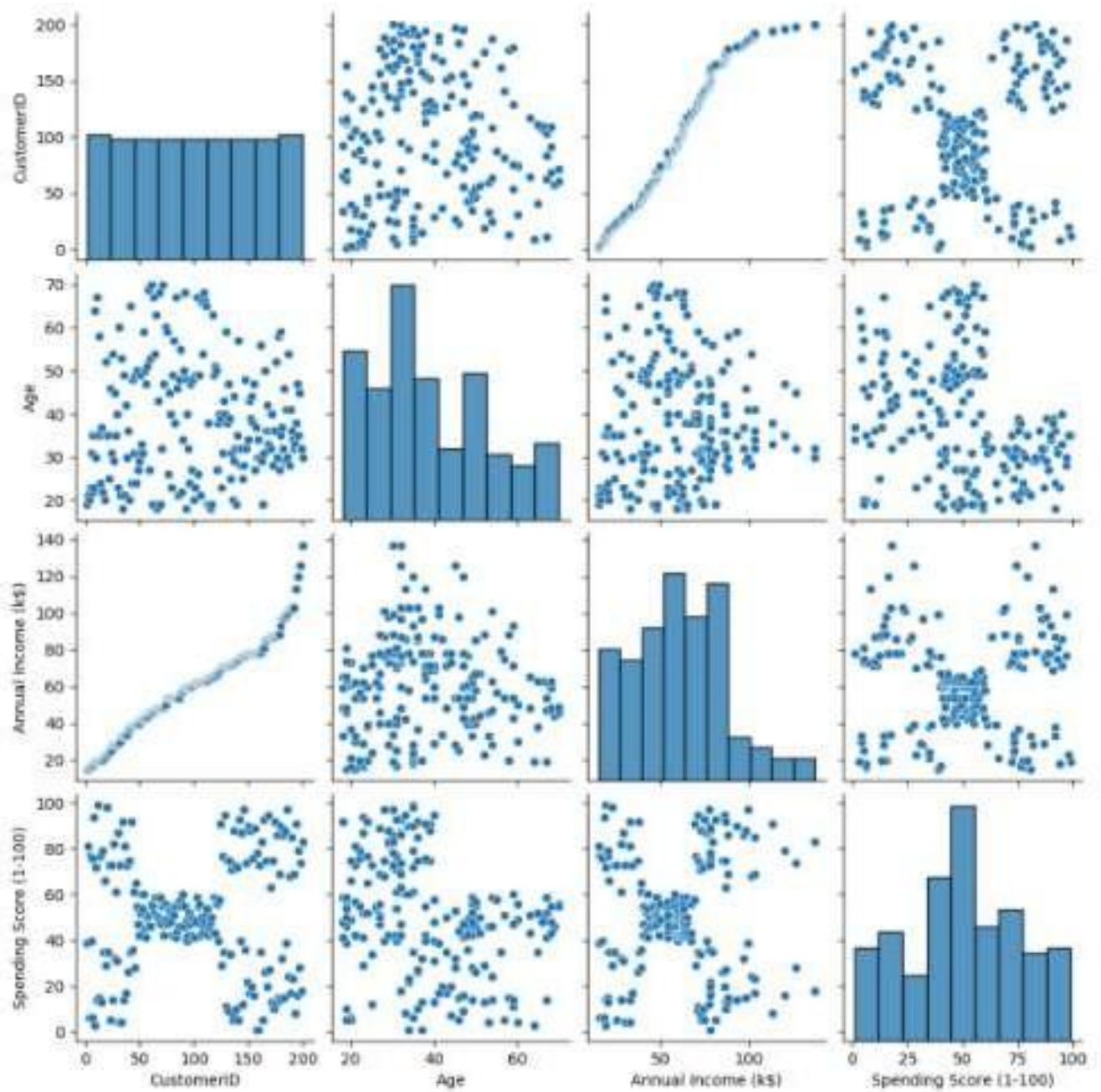
```
In [4]: df.head()
```

```
Out[4]:
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
In [5]: sns.pairplot(df)
```

```
Out[5]: <seaborn.axisgrid.PairGrid at 0x170e8e47850>
```



```
In [6]: features=df.iloc[:,[3,4]].values
```

```
In [7]: from sklearn.cluster import KMeans
model=KMeans(n_clusters=5)
model.fit(features)
KMeans(n_clusters=5)
```

C:\Users\Ayyadurai\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warning

```
warnings.warn(
```

C:\Users\Ayyadurai\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.

```
warnings.warn(
```

```
Out[7]: KMeans(n_clusters=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [8]: Final=df.iloc[:,[3,4]]
Final['label']=model.predict(features)
Final.head()
```

C:\Users\Ayyadurai\AppData\Local\Temp\ipykernel_8116\470183701.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

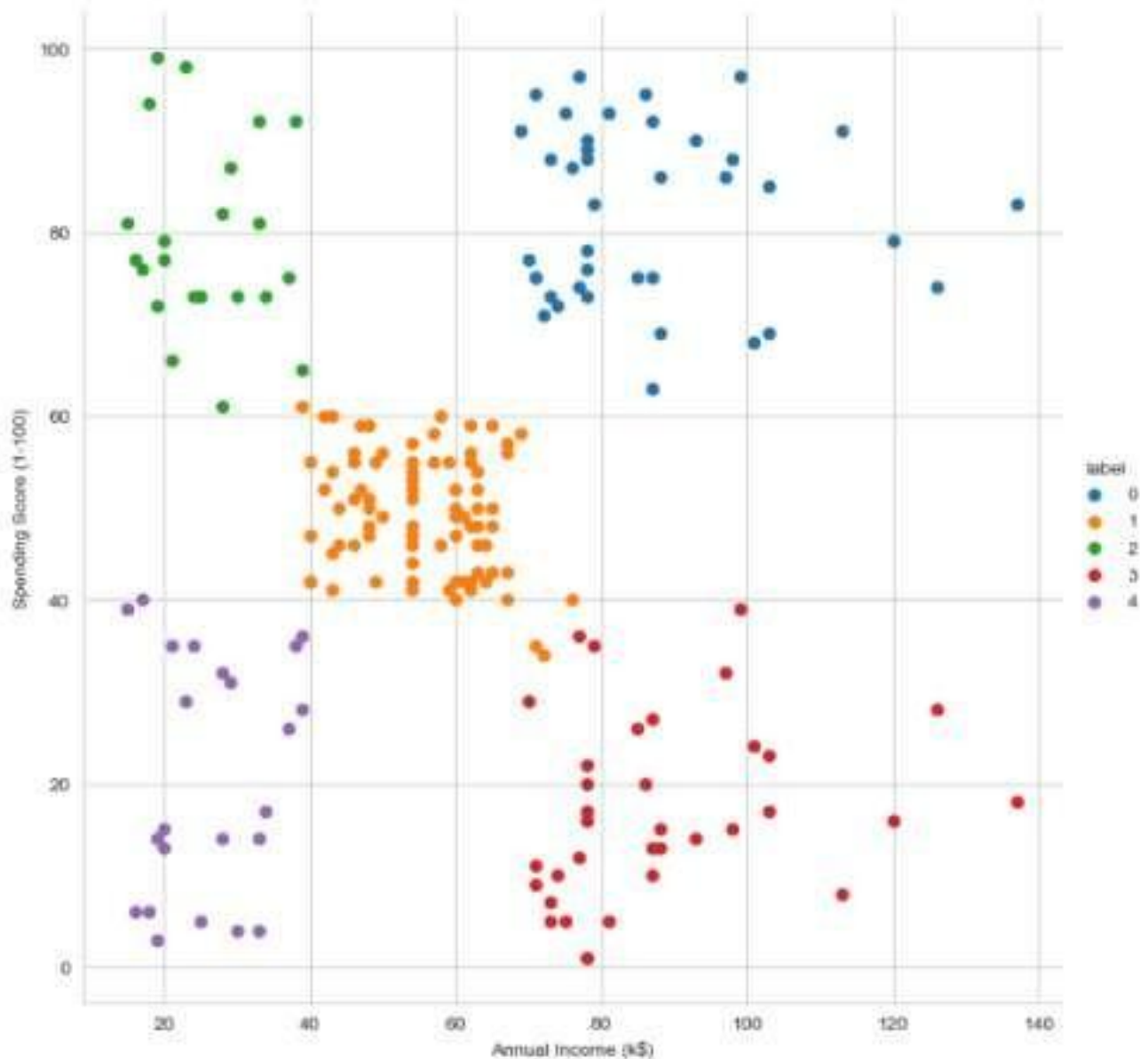
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
Final['label']=model.predict(features)
```

```
Out[8]:
```

	Annual Income (k\$)	Spending Score (1-100)	label
0	15	39	4
1	15	81	2
2	16	6	4
3	16	77	2
4	17	40	4

```
In [9]: sns.set_style("whitegrid")
sns.FacetGrid(Final, hue="label", height=8) \
.map(plt.scatter, "Annual Income (k$)", "Spending Score (1-100)") \
.add_legend();
plt.show()
```



```
In [10]: features_el=df.iloc[:,[2,3,4]].values
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,10):
    model=KMeans(n_clusters=i)
    model.fit(features_el)
    wcss.append(model.inertia_)
plt.plot(range(1,10),wcss)
```