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

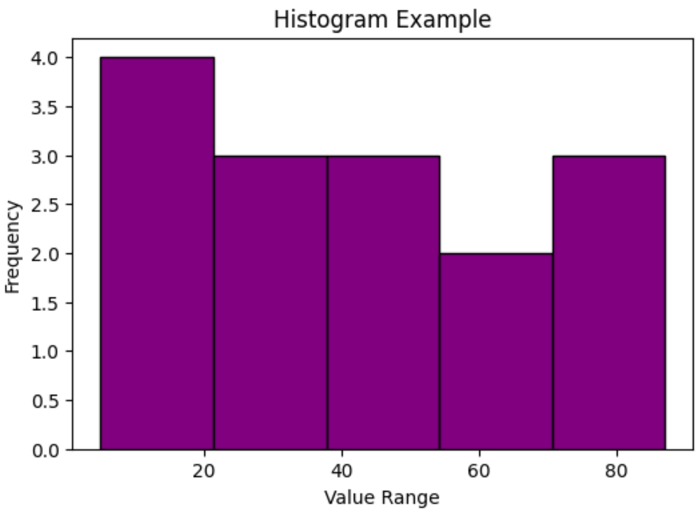
III semester II Year (2023R)

Name of the Student : Shivani R J

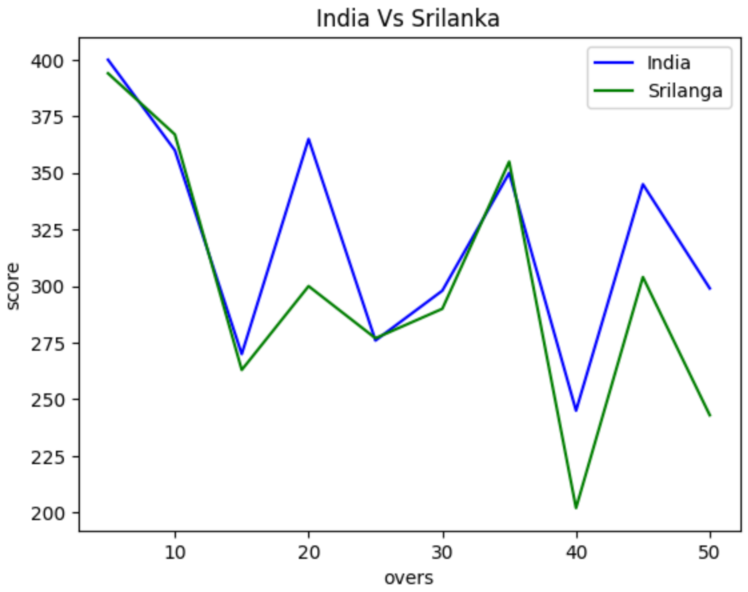
Register Number : 2116240701500

```
#Shivani R J
240701500
CSE
```

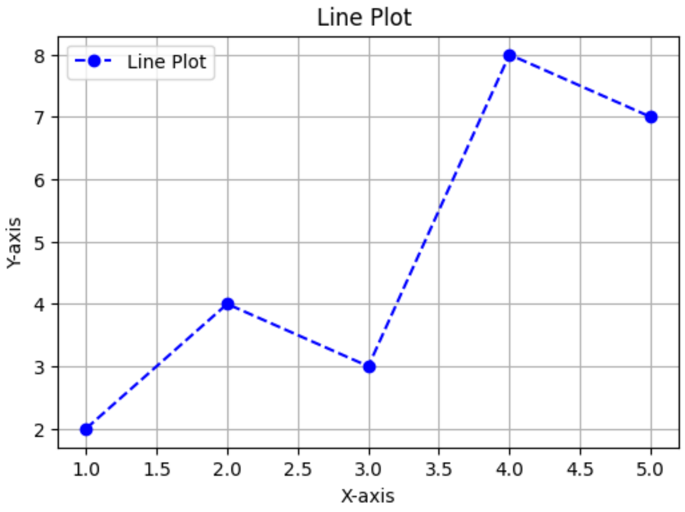
```
import matplotlib.pyplot as plt
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]
plt.figure(figsize=(6, 4))
plt.hist(data, bins=5, color='purple', edgecolor='black')
plt.title("Histogram Example")
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.show()
```



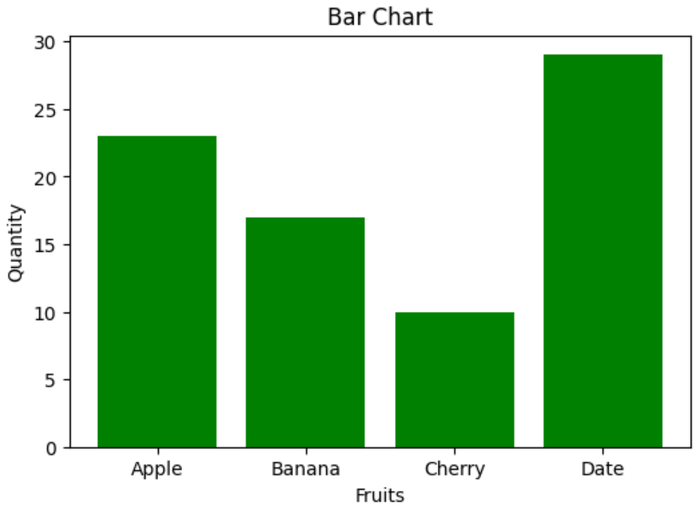
```
import matplotlib.pyplot as plt
overs=list(range(5,51,5))
India_Score=[400,360,270,365,276,298,350,245,345,299]
Srilanga_Score=[394,367,263,300,277,290,355,202,304,243]
plt.title("India Vs Srilanka")
plt.xlabel("overs")
plt.ylabel("score")
plt.plot(overs,India_Score,color="blue",label="India")
plt.plot(overs,Srilanga_Score,color="green",label="Srilanga")
plt.legend()
plt.show()
```



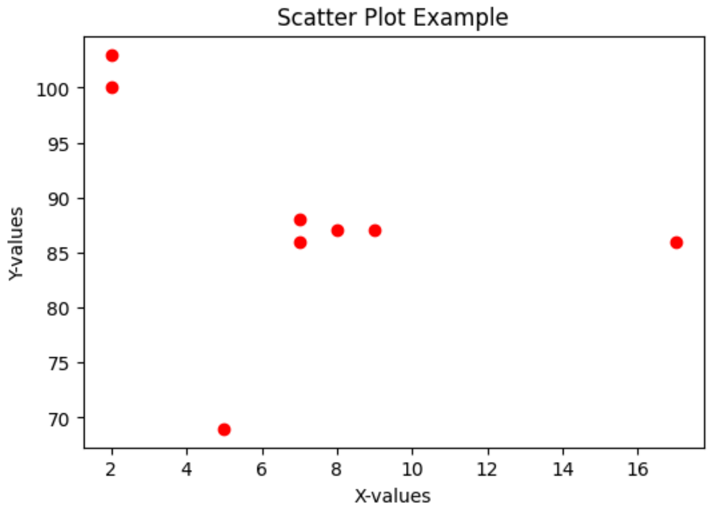
```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 3, 8, 7]
plt.figure(figsize=(6, 4))  # Set the figure size
plt.plot(x, y, color='blue', marker='o', linestyle='--', label='Line Plot')
plt.title("Line Plot ")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.grid(True)
plt.show()
```



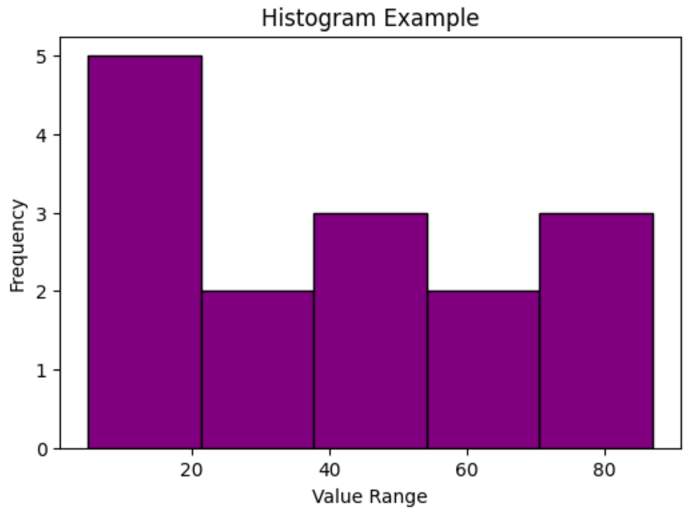
```
categories = ['Apple', 'Banana', 'Cherry', 'Date']
values = [23, 17, 10, 29]
plt.figure(figsize=(6, 4))
plt.bar(categories, values, color='green')
plt.title("Bar Chart")
plt.xlabel("Fruits")
plt.ylabel("Quantity")
plt.show()
```



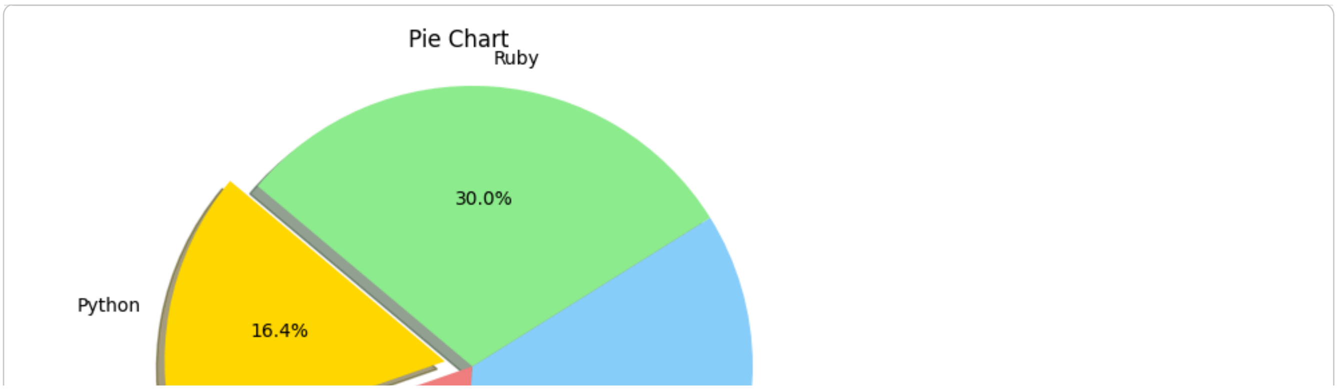
```
x_scatter = [5, 7, 8, 7, 2, 17, 2, 9]
y_scatter = [69, 86, 87, 88, 100, 86, 103, 87]
plt.figure(figsize=(6, 4))
plt.scatter(x_scatter, y_scatter, color='red')
plt.title("Scatter Plot")
plt.xlabel("X-values")
plt.ylabel("Y-values")
plt.show()
```



```
data = [12, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]
plt.figure(figsize=(6, 4))
plt.hist(data, bins=5, color='purple', edgecolor='black')
plt.title("Histogram")
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.show()
```



```
labels = ['Python', 'Java', 'C++', 'Ruby']
sizes = [115, 130, 245, 210]
colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen']
explode = (0.1, 0, 0, 0)  # Explode the 1st slice (Python)
plt.figure(figsize=(6, 6))
plt.pie(sizes, explode=explode, labels=labels, colors=colors,autopct='%1.1f%%',
shadow=True, startangle=140)
plt.title("Pie Chart")
plt.axis('equal')  # Equal aspect ratio ensures the pie is drawn as a circle.
plt.show()
```


Pie Chart
Ruby
30.0%
Python
16.4%

```
Shivani R J
240701500
CSE
```

```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving sales_data (1).csv to sales_data (1).csv
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('sales_data (1).csv')
print(df)
```

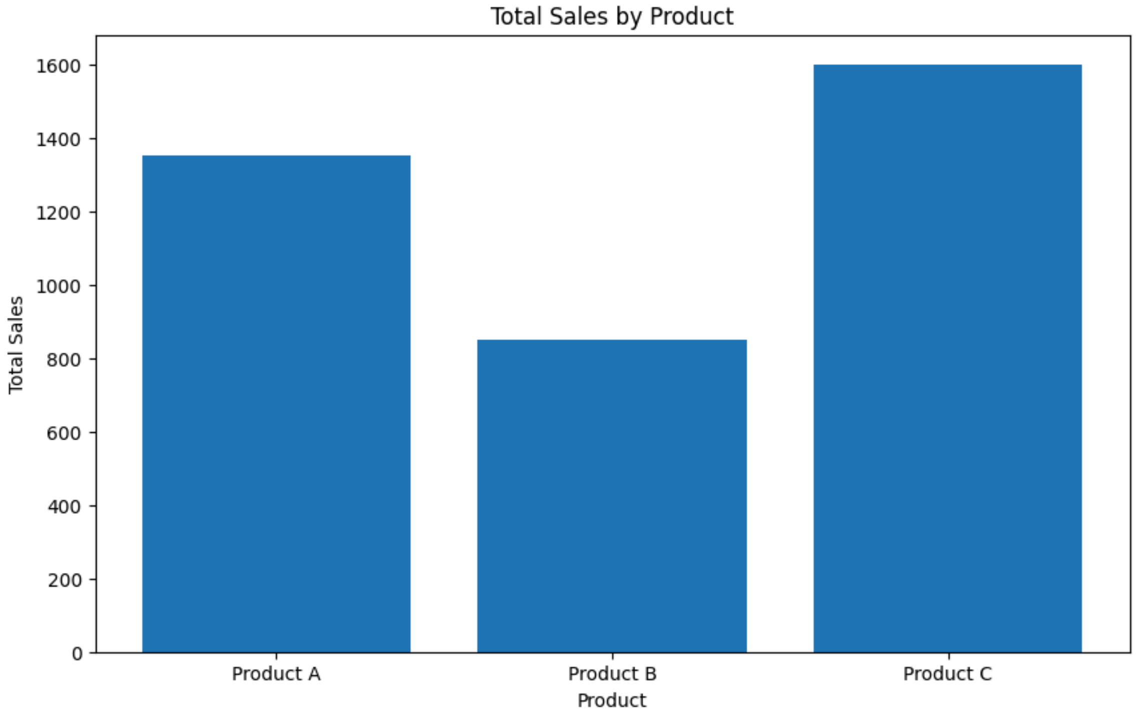
```
          Date    Product  Sales  Quantity Region
0   01-01-2023  Product A    200         4  North
1   02-01-2023  Product B    150         3  South
2   03-01-2023  Product A    220         5  North
3   04-01-2023  Product C    300         6   East
4   05-01-2023  Product B    180         4   West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

```
print(df.isnull().sum())
df['Sales'].fillna(df['Sales'].mean(),inplace=True)
df.dropna(subset=['Quantity','Region','Product'],inplace=True)
```

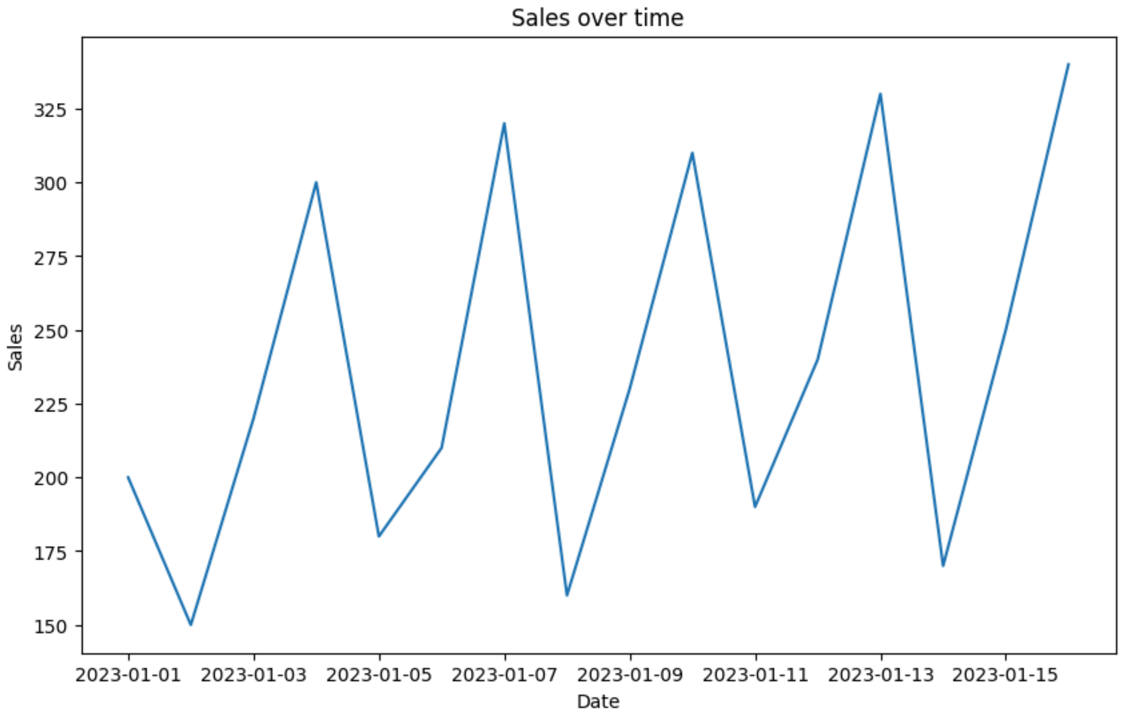
```
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64
/tmp/ipython-input-925655291.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df['Sales'].fillna(df['Sales'].mean(),inplace=True)
```

```
product_summary=df.groupby('Product'). agg({'Sales':'sum','Quantity':'sum'}).reset_index()
print(product_summary)
plt.figure(figsize=(10,6))
plt.bar(product_summary['Product'],product_summary['Sales'])
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.title('Total Sales by Product')
plt.show()
```

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



```
df['Date']=pd.to_datetime(df['Date'],dayfirst=True)
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('Sales')
plt.title('Sales over time')
plt.show()
```



```
pivot_table=df.pivot_table(values='Sales',index='Region',columns='Product',aggfunc=np.sum,fill_value=0)
print(pivot_table)
```

```
Product  Product A  Product B  Product C
Region
East             0          0       1600
North         1350          0          0
South            0        480          0
West             0        370          0
```

```
/tmp/ipython-input-3775029091.py:1: FutureWarning: The provided callable <function sum at 0x7bf42c767100> is currently using
  pivot_table=df.pivot_table(values='Sales',index='Region',columns='Product',aggfunc=np.sum,fill_value=0)
```

```
correlation_matrix=df.corr()
print(correlation_matrix)
import seaborn as sns
plt.figure(figsize=(10,6))
sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm')
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-3935011219.py in <cell line: 0>()
----> 1 correlation_matrix=df.corr()
      2 print(correlation_matrix)
      3 import seaborn as sns
      4 plt.figure(figsize=(10,6))
      5 sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm')

                                  3 frames

/usr/local/lib/python3.12/dist-packages/pandas/core/internals/managers.py in _interleave(self, dtype, na_value)
   1751             else:
   1752                 arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755 

ValueError: could not convert string to float: 'Product A'
```

```
Shivani
240701500
CSE
```

```
import numpy as np
import pandas as pd
df=pd.read_csv("pre_process_datasample.csv")
df
```

	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

Next steps: Generate code with df New interactive sheet

```
from google.colab import files
uploaded = files.upload()
```

Choose Files pre_proces…asample.csv

pre_process_datasample.csv(text/csv) - 226 bytes, last modified: 8/12/2025 - 100% done

```
Saving pre_process_datasample.csv to pre_process_datasample.csv
```

```
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
```

```
df.Country.mode()
```

	Country
0	France

dtype: object

```
df.Country.mode()[0]
```

```
'France'
```

```
type(df.Country.mode())
```

pandas.core.series.Series

```
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None,
fastpath: bool | lib.NoDefault=lib.no_default) -> None
```

/usr/local/lib/python3.12/dist-packages/pandas/core/series.py

```
One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
methods for performing operations involving the index. Statistical
```

```
df.Country.fillna(df.Country.mode()[0],inplace=True)
df.Age.fillna(df.Age.median(),inplace=True)
df.Salary.fillna(round(df.Salary.mean()),inplace=True)
df
```

```
/tmp/ipython-input-1020198583.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df.Age.fillna(df.Age.median(),inplace=True)
/tmp/ipython-input-1020198583.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df.Salary.fillna(round(df.Salary.mean()),inplace=True)
```

	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	63778.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	38.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

Next steps: Generate code with df New interactive sheet

```
pd.get_dummies(df.Country)
```

	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

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

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

```
updated_dataset.Purchased.replace(['No','Yes'],[0,1],inplace=True)
updated_dataset
```

```
Shivani R J
240701500
CSE
```

```
import numpy as np
import pandas as pd
df=pd.read_csv('Hotel_Dataset.csv')
df
```

```
    CustomerID Age_Group Rating(1-5)      Hotel FoodPreference  Bill NoOfPax EstimatedSalary Age_Group.1
 0           1     20-25           4       Ibis            veg  1300       2           40000       20-25
 1           2     30-35           5  LemonTree        Non-Veg  2000       3           59000       30-35
 2           3     25-30           6     RedFox            Veg  1322       2           30000       25-30
 3           4     20-25          -1  LemonTree            Veg  1234       2          120000       20-25
 4           5       35+           3       Ibis     Vegetarian   989       2           45000         35+
 5           6       35+           3       Ibys        Non-Veg  1909       2          122220         35+
 6           7       35+           4     RedFox     Vegetarian  1000      -1           21122         35+
 7           8     20-25           7  LemonTree            Veg  2999     -10          345673       20-25
 8           9     25-30           2       Ibis        Non-Veg  3456       3          -99999       25-30
 9           9     25-30           2       Ibis        Non-Veg  3456       3          -99999       25-30
10          10     30-35           5     RedFox        non-Veg -6755       4           87777       30-35
```

Next steps: Generate code with df   New interactive sheet

```
from google.colab import files
uploded=files.upload()
```

Choose Files Hotel_Dataset.csv

Hotel_Dataset.csv(text/csv) - 576 bytes, last modified: 11/3/2025 - 100% done

```
Saving Hotel_Dataset.csv to Hotel_Dataset.csv
```

```
df.duplicated()
df.drop_duplicates(inplace=True)
df
```

```
    CustomerID Age_Group Rating(1-5)      Hotel FoodPreference  Bill NoOfPax EstimatedSalary Age_Group.1
 0           1     20-25           4       Ibis            veg  1300       2           40000       20-25
 1           2     30-35           5  LemonTree        Non-Veg  2000       3           59000       30-35
 2           3     25-30           6     RedFox            Veg  1322       2           30000       25-30
 3           4     20-25          -1  LemonTree            Veg  1234       2          120000       20-25
 4           5       35+           3       Ibis     Vegetarian   989       2           45000         35+
 5           6       35+           3       Ibys        Non-Veg  1909       2          122220         35+
 6           7       35+           4     RedFox     Vegetarian  1000      -1           21122         35+
 7           8     20-25           7  LemonTree            Veg  2999     -10          345673       20-25
 8           9     25-30           2       Ibis        Non-Veg  3456       3          -99999       25-30
10          10     30-35           5     RedFox        non-Veg -6755       4           87777       30-35
```

Next steps: Generate code with df   New interactive sheet

```
index=np.array(list(range(0,len(df))))
df.set_index(index,inplace=True)
index
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
df.drop(['Age_Group.1'],axis=1,inplace=True)
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Vegetarian	989	2	45000
5	6	35+	3	Ibys	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777

Next steps: Generate code with df New interactive sheet

```
df.CustomerID.loc[df.CustomerId<0]=np.nan
df.Bill.loc[df.bill<0]=np.nan
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-3015053723.py in <cell line: 0>()
----> 1 df.CustomerID.loc[df.CustomerId<0]=np.nan
      2 df.Bill.loc[df.bill<0]=np.nan

/usr/local/lib/python3.12/dist-packages/pandas/core/generic.py in __getattr__(self, name)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)
   6300 
   6301     @final

AttributeError: 'DataFrame' object has no attribute 'CustomerId'
```

Next steps: Explain error

```
df.Hotel.unique()
```

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

```
df.Hotel.replace(['Ibys'],'ibis',inplace=True)
df
```

```
/tmp/ipython-input-1758254601.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df.Hotel.replace(['Ibys'],'ibis',inplace=True)
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Vegetarian	989	2	45000
5	6	35+	3	ibis	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777

Next steps: Generate code with df New interactive sheet

```
df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)
df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)
```

```
/tmp/ipython-input-3377581060.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)
```

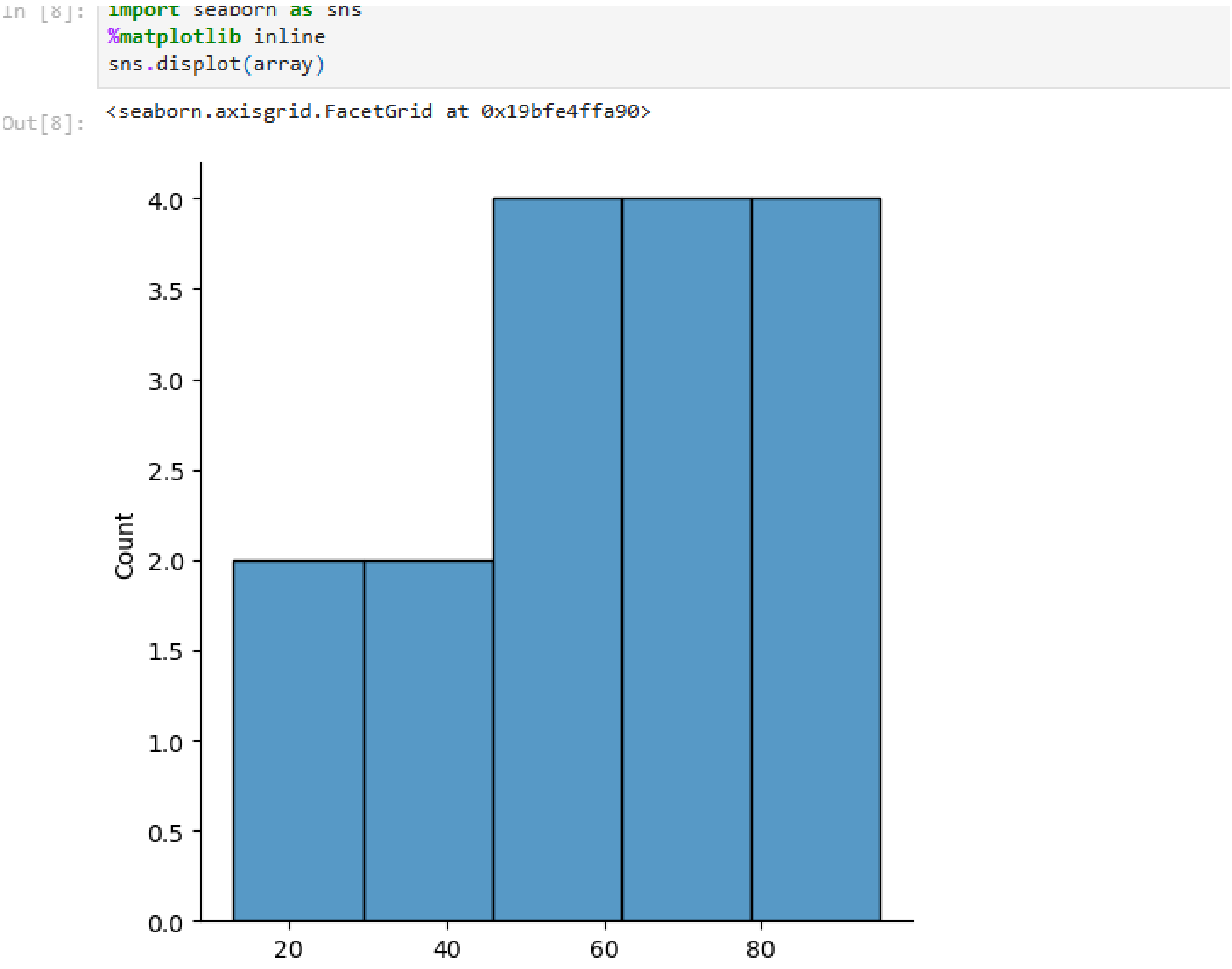
```
df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)
df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)
df.Bill.fillna(round(df.Bill.mean()),inplace=True)
df
```

```
/tmp/ipython-input-3711388855.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
/tmp/ipython-input-3711388855.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)
/tmp/ipython-input-3711388855.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)
/tmp/ipython-input-3711388855.py:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df.Bill.fillna(round(df.Bill.mean()),inplace=True)
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	Veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Veg	989	2	45000
5	6	35+	3	ibis	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Veg	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	Non-Veg	-6755	4	87777

Next steps: Generate code with df New interactive sheet

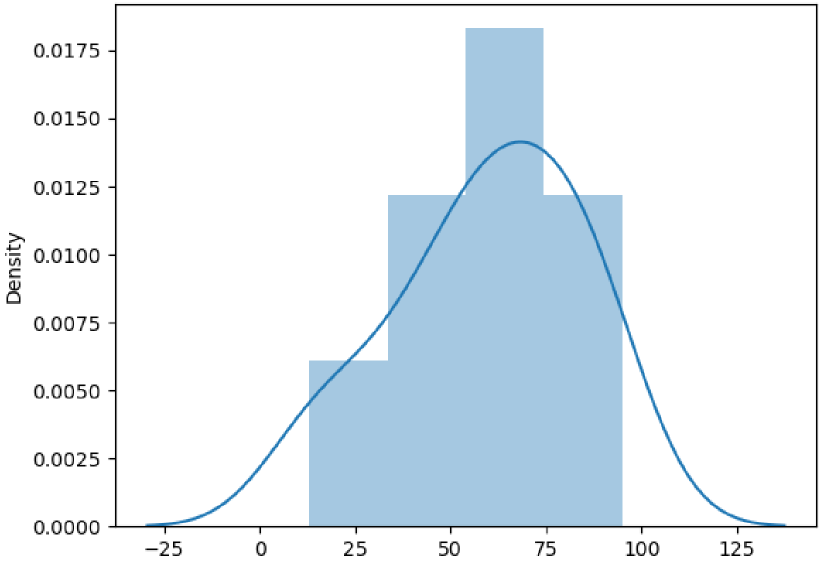
```
In [1]: import numpy as np
        array=np.random.randint(1,100,16)
        array

Out[1]: array([86, 17, 49, 95, 52, 62, 69, 74, 74, 34, 84, 13, 38, 87, 63, 56])

In [2]: array.mean()

Out[2]: 59.5625

In [3]: np.percentile(array,25)

Out[3]: 46.25

In [4]: np.percentile(array,50)

Out[4]: 62.5

In [5]: np.percentile(array,75)

Out[5]: 76.5

In [6]: np.percentile(array,100)

Out[6]: 95.0

In [7]: def outDetection(array):
            sorted(array)
            Q1,Q3=np.percentile(array,[25,75])
            IQR=Q3-Q1
            lr=Q1-(1.5*IQR)
            ur=Q3+(1.5*IQR)
            return lr,ur
        lr,ur=outDetection(array)
        lr,ur

Out[7]: (0.875, 121.875)

In [8]: import seaborn as sns
        %matplotlib inline
        sns.displot(array)

Out[8]: <seaborn.axisgrid.FacetGrid at 0x19bfe4ffa90>
```



```
In [8]: import seaborn as sns
        %matplotlib inline
        sns.displot(array)
```

```
Out[8]: <seaborn.axisgrid.FacetGrid at 0x19bfe4ffa90>
```

```
ut[9]: <Axes: ylabel='Density'>
```



```
[10]: new_array=array[(array>lr)&(array<ur)]
      new_array
```

```
t[10]: array([86, 17, 49, 95, 52, 62, 69, 74, 74, 34, 84, 13, 38, 87, 63, 56])
```

```
[11]: sns.displot(new_array)
```

```
t[11]: <seaborn.axisgrid.FacetGrid at 0x19bf8e54f90>
```

```
]: lr1,ur1=outDetection(new_array)
   lr1,ur1
```

```
]: (0.875, 121.875)
```

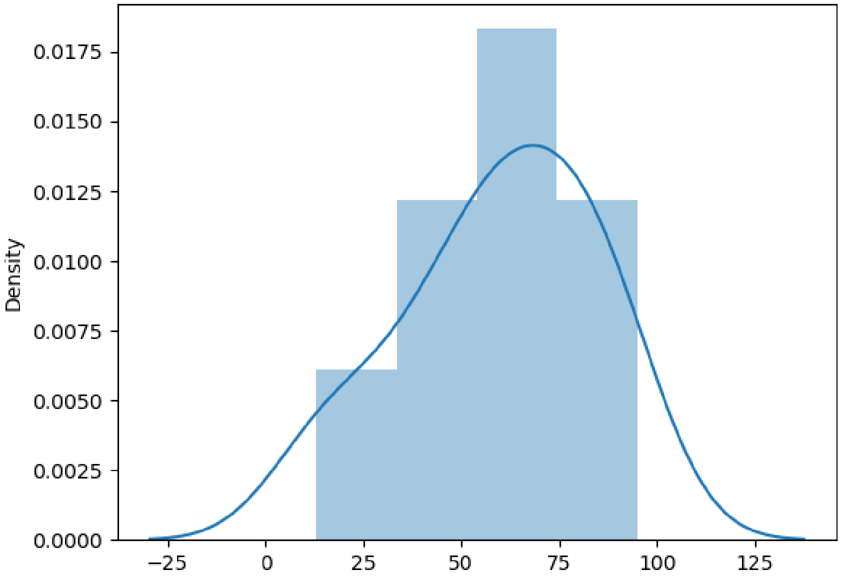
```
]: final_array=new_array[(new_array>lr1)&(new_array<ur1)]
   final_array
```

```
]: array([86, 17, 49, 95, 52, 62, 69, 74, 74, 34, 84, 13, 38, 87, 63, 56])
```

```
]: sns.distplot(final_array)
```

```
  sns.distplot(final_array)
```

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

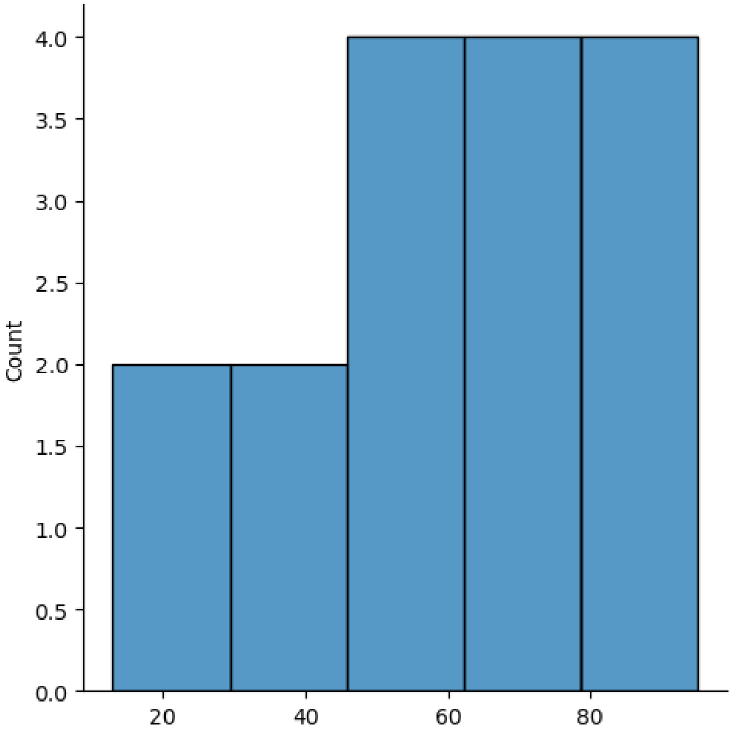


```
[10]: new_array=array[(array>lr)&(array<ur)]
      new_array
```

```
[10]: array([86, 17, 49, 95, 52, 62, 69, 74, 74, 34, 84, 13, 38, 87, 63, 56])
```

```
[11]: sns.displot(new_array)
```

```
[11]: <seaborn.axisgrid.FacetGrid at 0x19bf8e54f90>
```



In [1]:

```python
import numpy as np
import pandas as pd
df=pd.read_csv('Downloads\pre_process_datasample.csv')
df
```

Out[1]:

	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

In [2]:

```python
df.head()
```

Out[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

In [5]:

```python
df.Country.fillna(df.Country.mode()[0],inplace=True)
features=df.iloc[:,:-1].values
df.Country.fillna(df.Country.mode()[0],inplace=True)
label=df.iloc[:,-1].values
```

In [6]:

```python
from sklearn.impute import SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np.nan)
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
age.fit(features[:,[1]])
```

Out[6]:

```
▾ SimpleImputer
SimpleImputer()
```

```
In [7]: Salary.fit(features[:,[2]])
```

```
Out[7]: ▾ SimpleImputer
        SimpleImputer()
```

```
In [8]: SimpleImputer()
```

```
Out[8]: ▾ SimpleImputer
        SimpleImputer()
```

```
In [9]: features[:,[1]]=age.transform(features[:,[1]])
        features[:,[2]]=Salary.transform(features[:,[2]])
        features
```

```
Out[9]: 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)
```

```
In [10]: from sklearn.preprocessing import OneHotEncoder
         oh = OneHotEncoder(sparse_output=False)
         Country=oh.fit_transform(features[:,[0]])
         Country
```

```
Out[10]: 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.]])
```

```
In [11]: final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
         final_set
```

```
Out[11]: 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)
```

```
final_set
```

```
Out[11]: 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)
```

```
In [12]: from sklearn.preprocessing import StandardScaler
         sc=StandardScaler()
         sc.fit(final_set)
         feat_standard_scaler=sc.transform(final_set)
         feat_standard_scaler
```

```
Out[12]: 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]])
```

```
In [13]: 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
```

```
Out[13]: 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]])
```

```
In [ ]:
```

In [19]:

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

Out[19]:

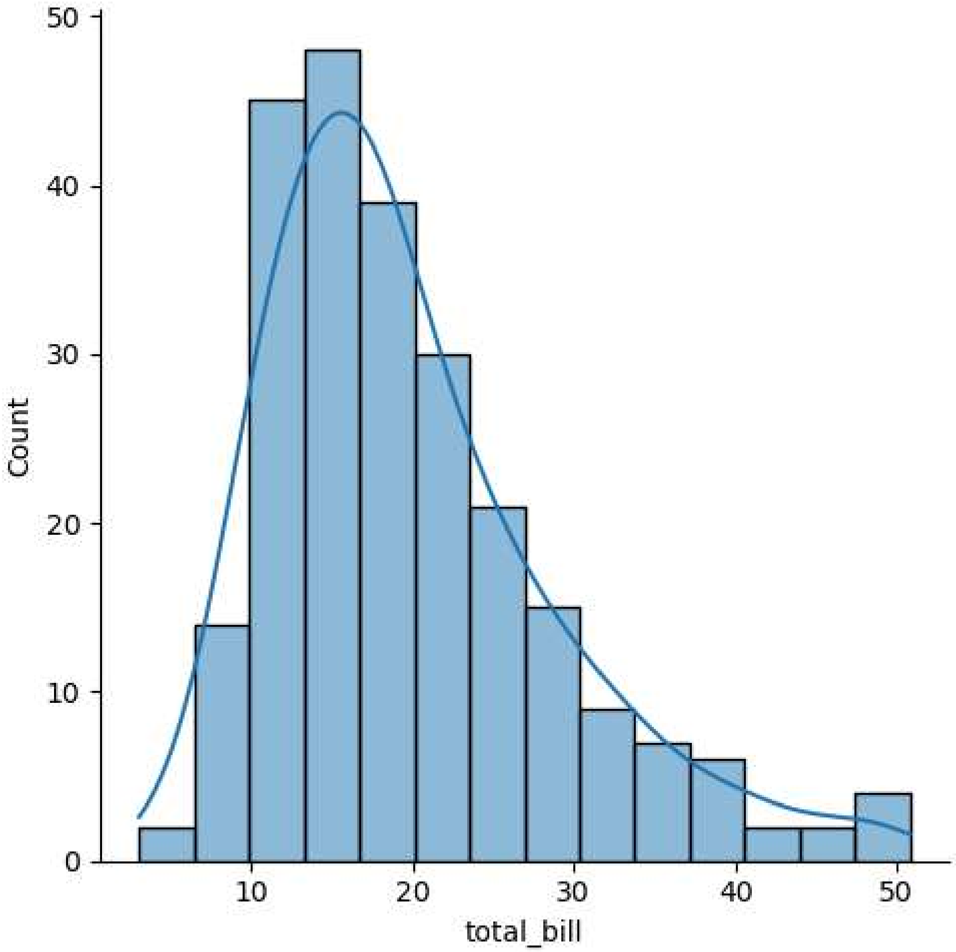
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [6]:

```
sns.displot(tips.total_bill,kde=True)
```

Out[6]:

```
<seaborn.axisgrid.FacetGrid at 0x1ecc6eef3d0>
```

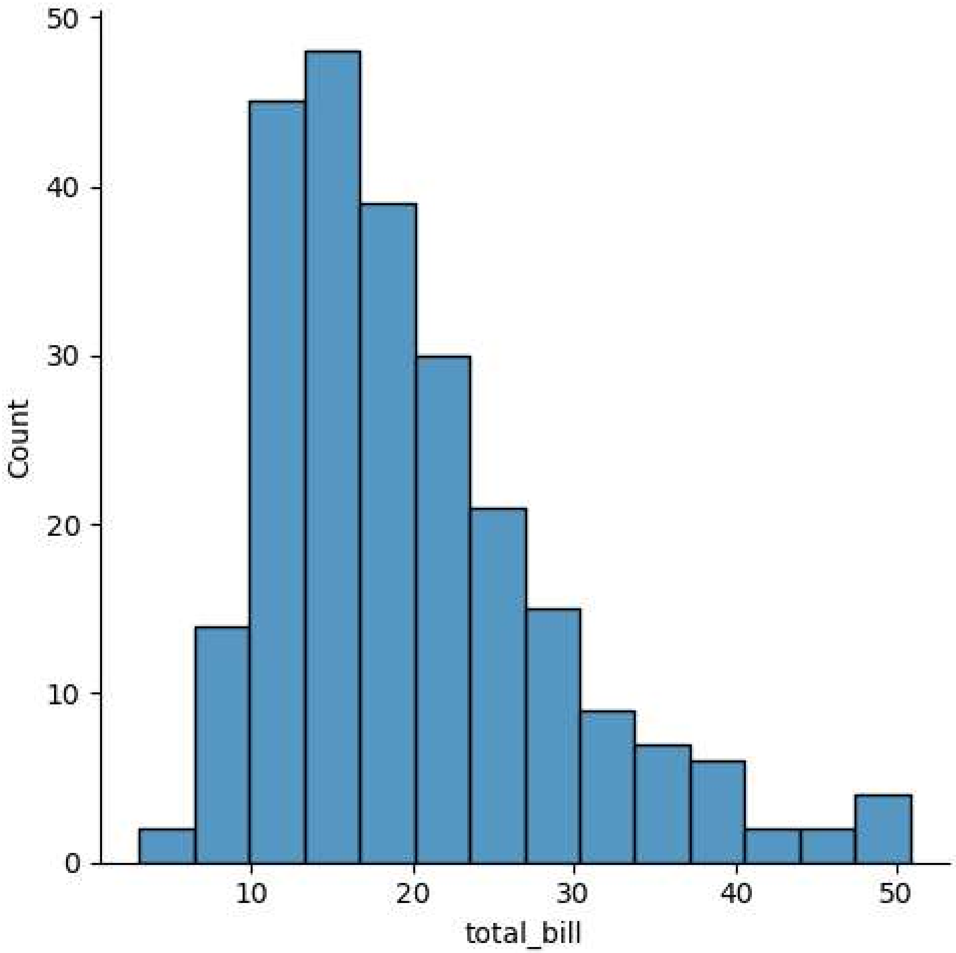


In [7]:

```
sns.displot(tips.total_bill,kde=False)
```

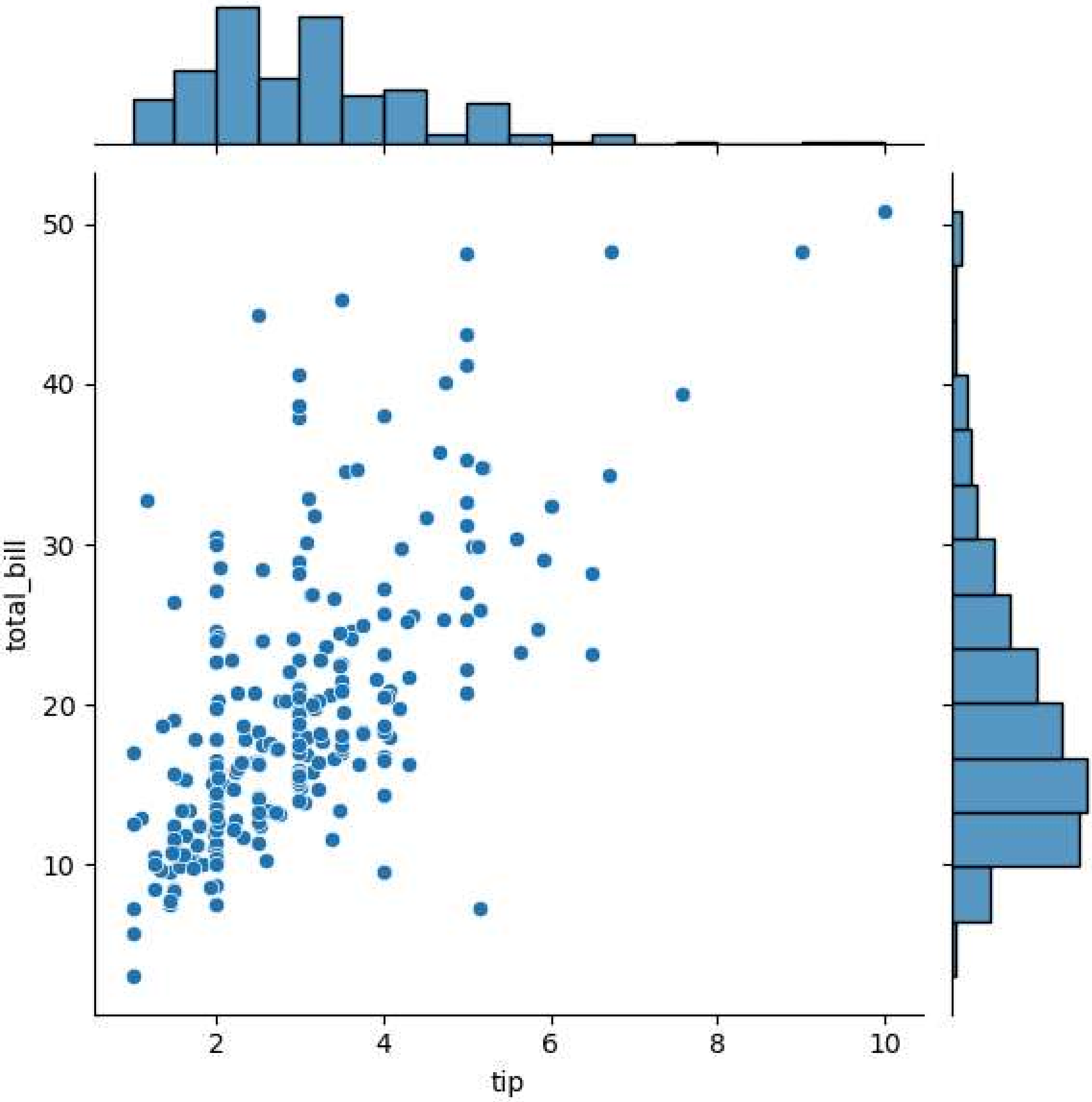
Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x1ecc6fc7410>
```



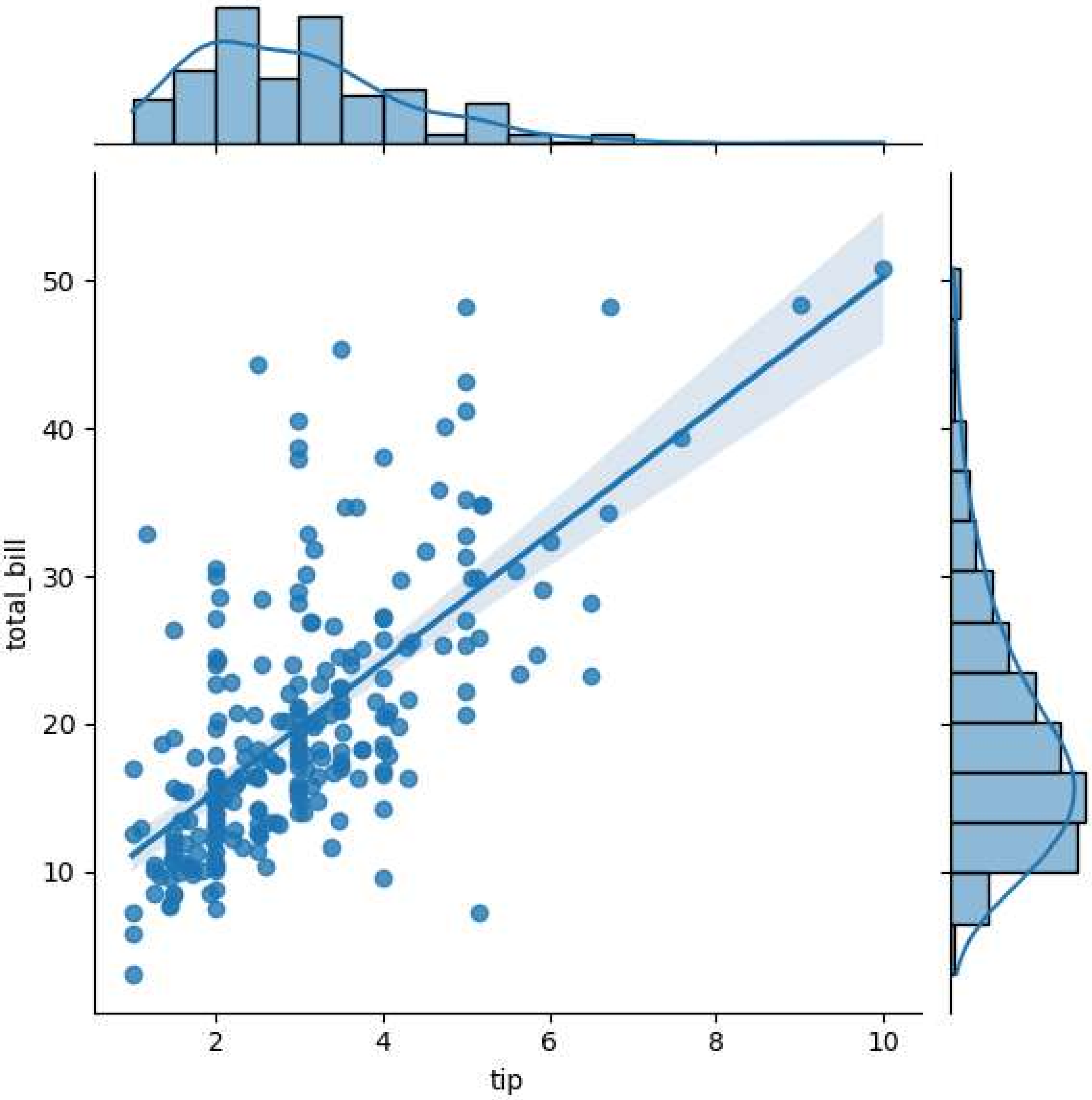
In [8]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill)
```

Out[8]:
```
<seaborn.axisgrid.JointGrid at 0x1ecc703bd10>
```



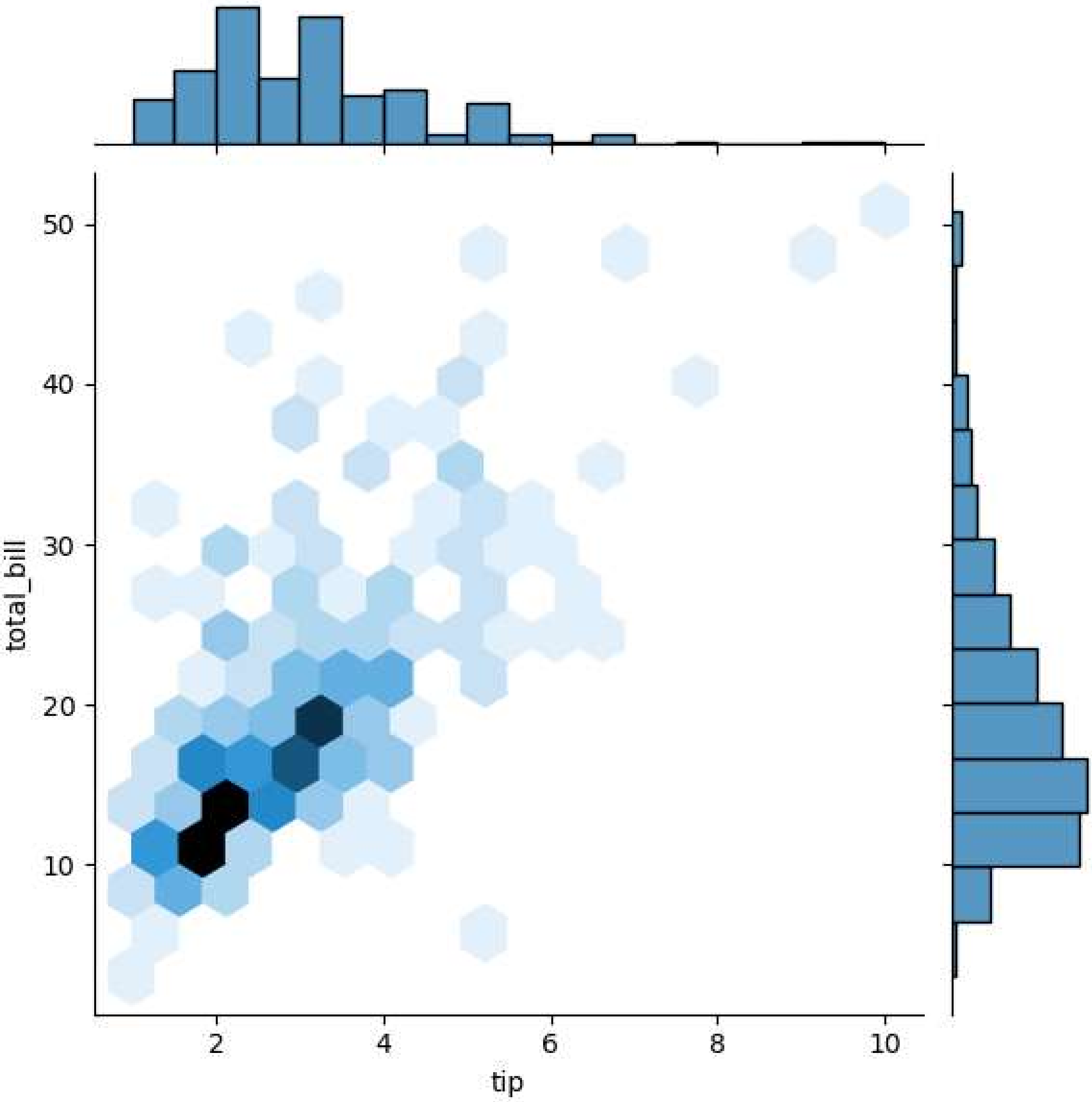
```
In [9]: sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
```

```
Out[9]: <seaborn.axisgrid.JointGrid at 0x1ecc6f34b90>
```



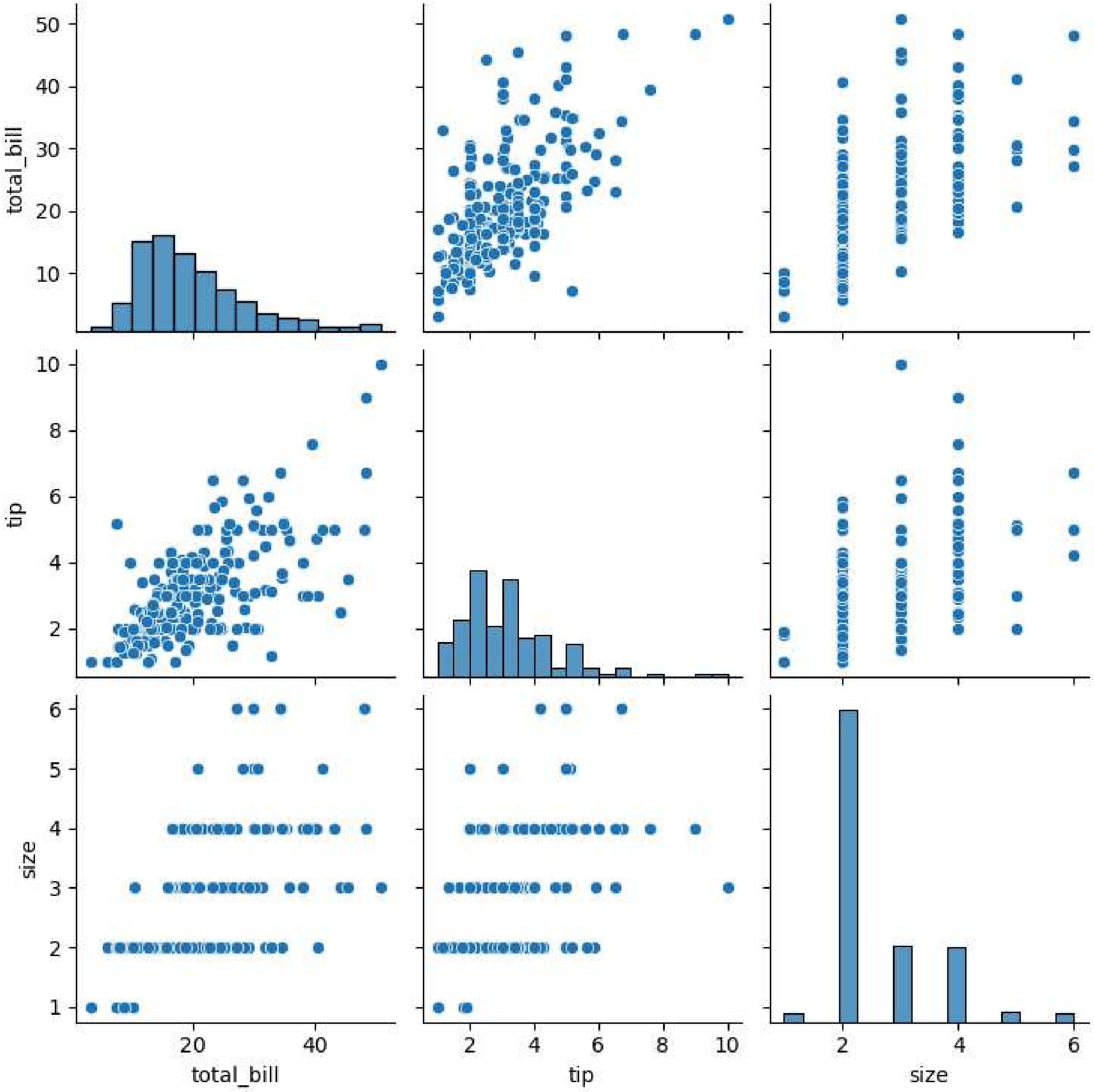
```
In [10]: sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
```

```
Out[10]: <seaborn.axisgrid.JointGrid at 0x1ecc81fe5d0>
```



In [11]:
```
sns.pairplot(tips)
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x1ecc7e1cad0>
```

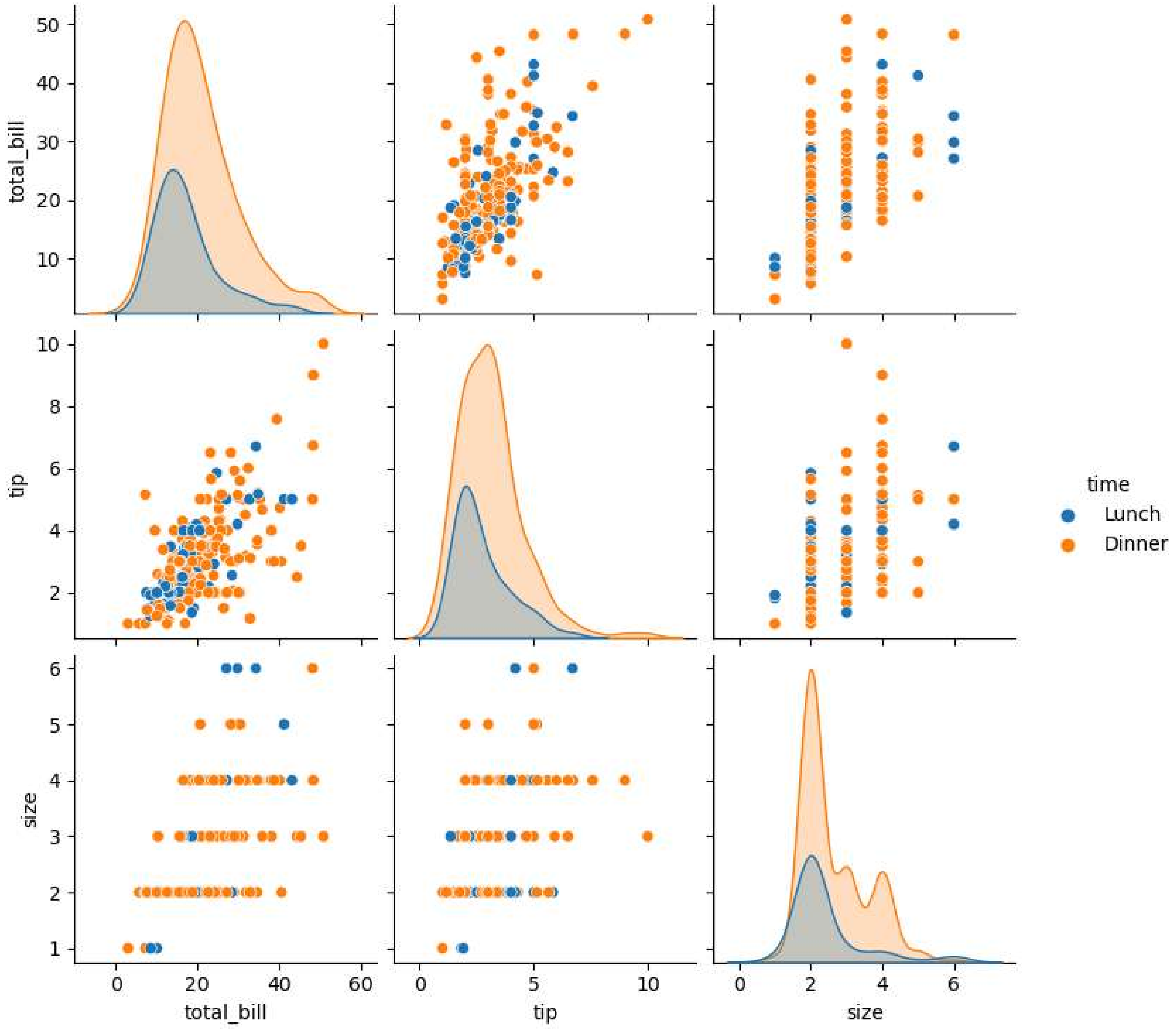


In [12]:
```
tips.time.value_counts()
```

Out[12]:
```
Dinner    176
Lunch      68
Name: time, dtype: int64
```

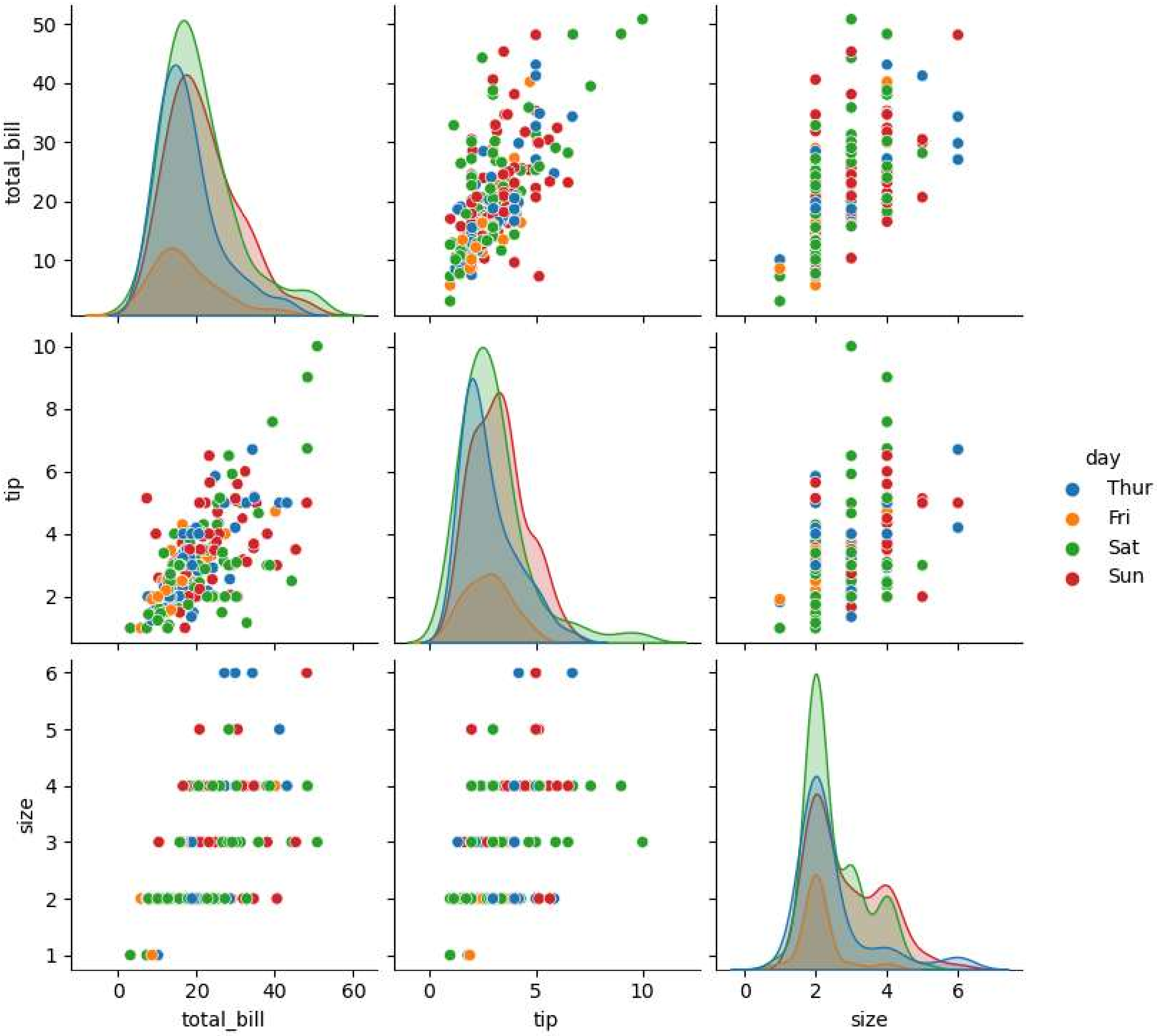
In [13]:
```
sns.pairplot(tips,hue='time')
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x1ecc8594ad0>
```



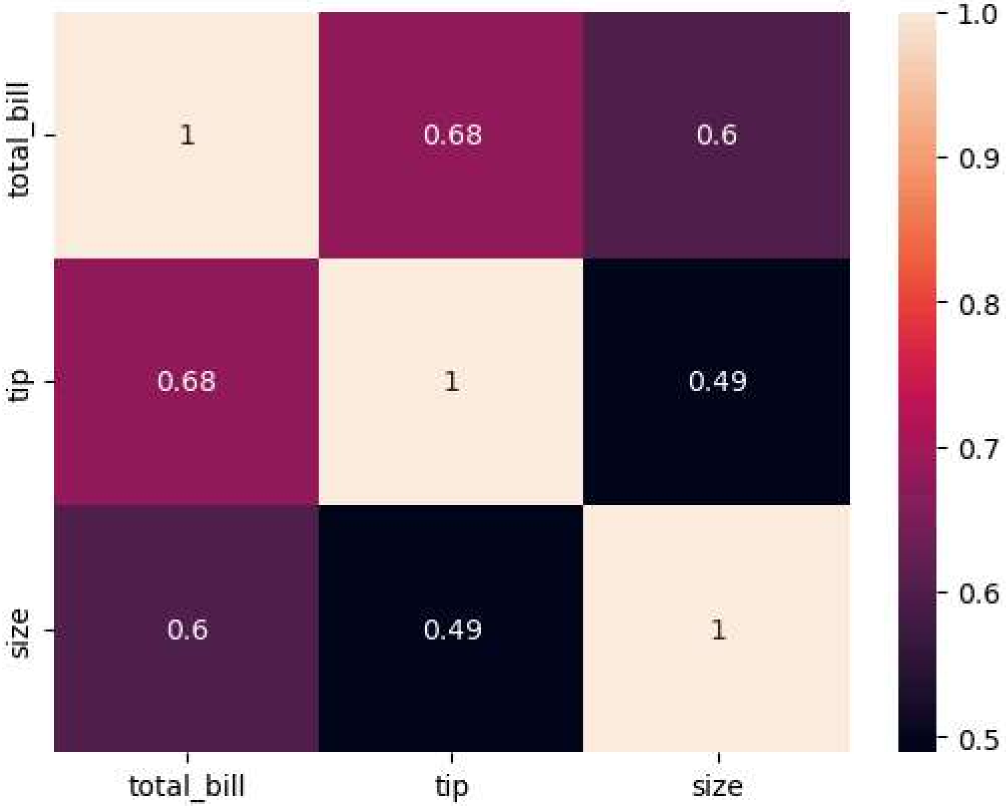
In [14]:
```
sns.pairplot(tips,hue='day')
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x1ecc6ee76d0>
```



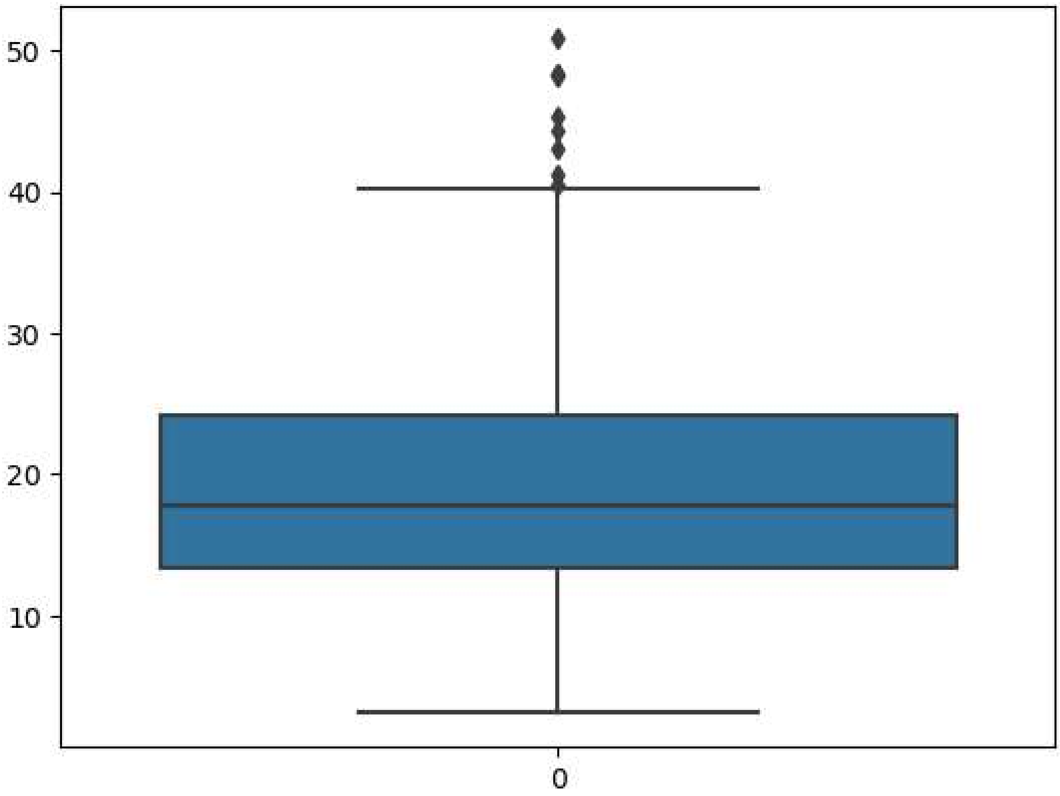
```
In [15]: sns.heatmap(tips.corr(numeric_only=True),annot=True)
```

```
Out[15]: <Axes: >
```



In [16]:
```
sns.boxplot(tips.total_bill)
```

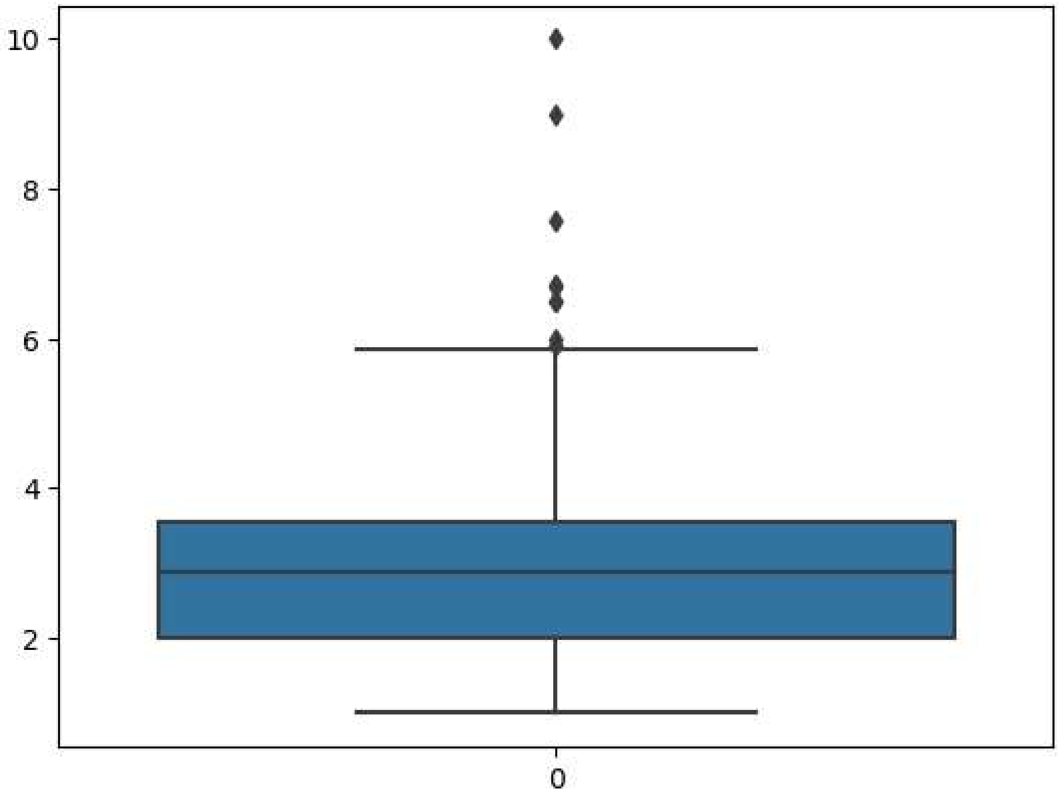
Out[16]:
```
<Axes: >
```



In [17]:
```
sns.boxplot(tips.tip)
```

Out[17]:

```
<Axes: >
```

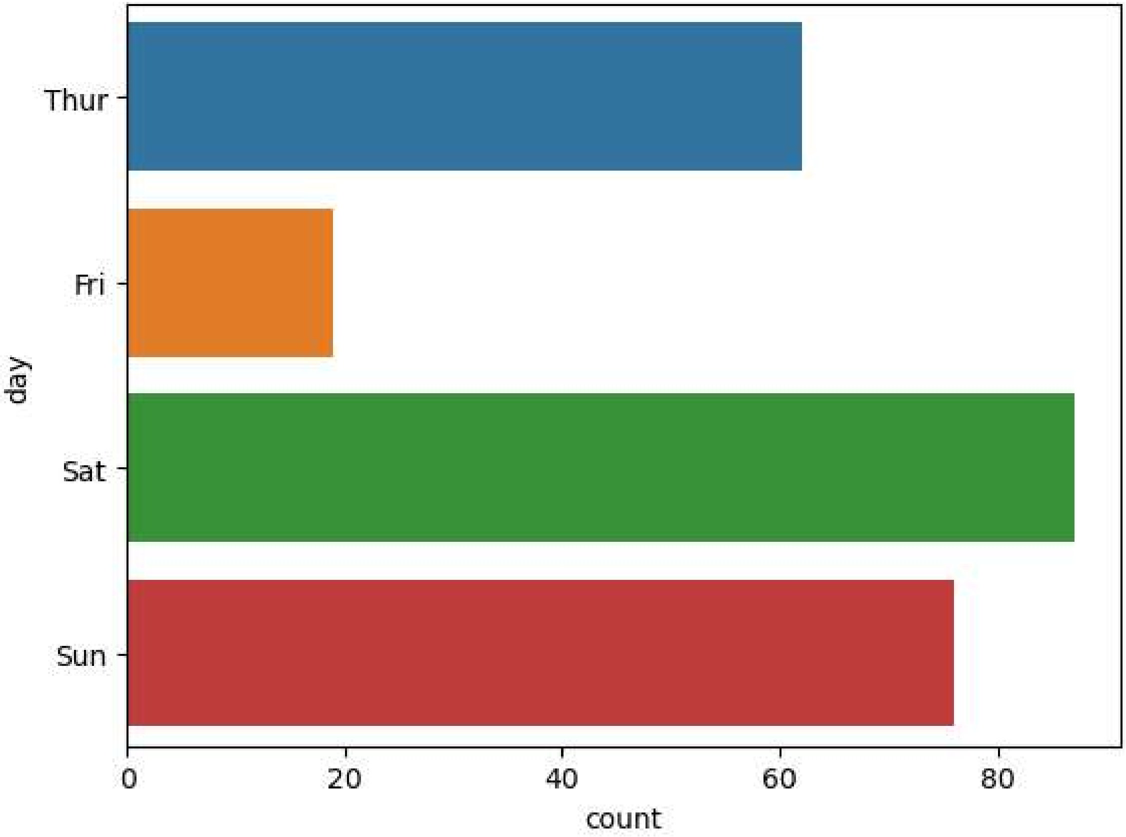


In [21]:

```
sns.countplot(y=tips.day)
```

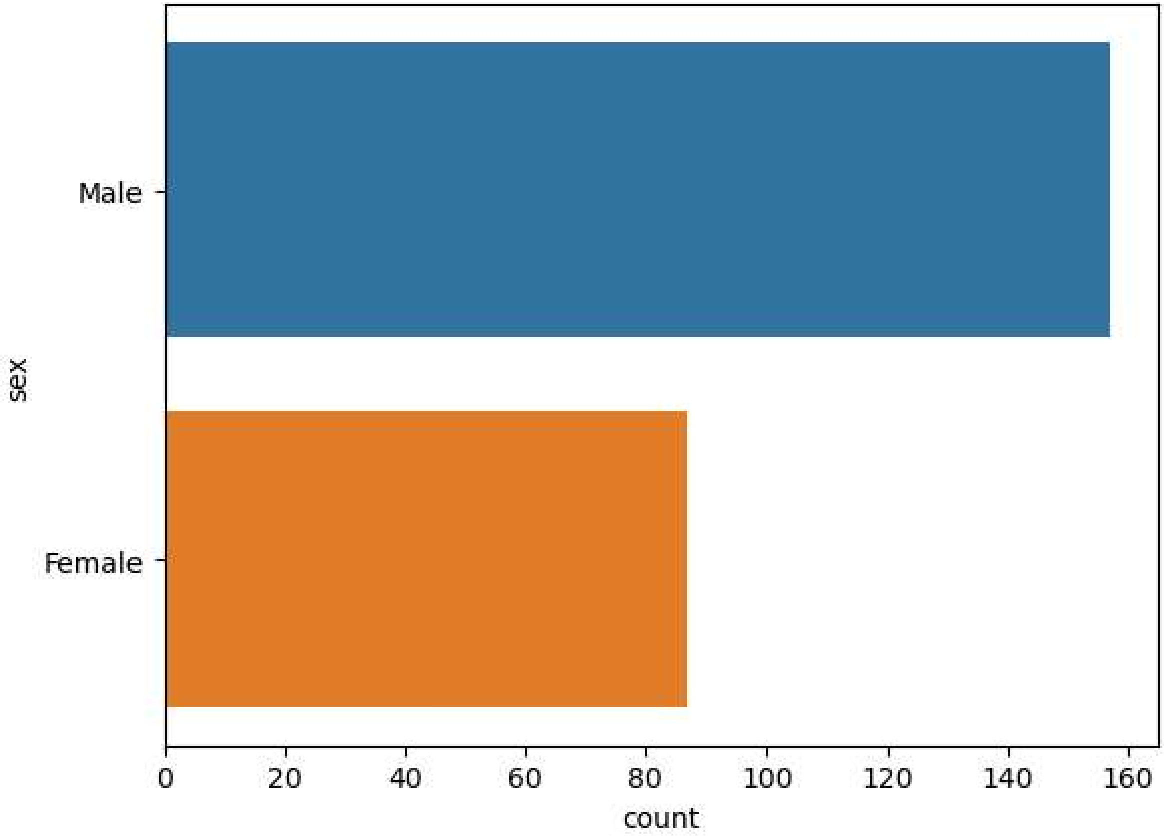
Out[21]:

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



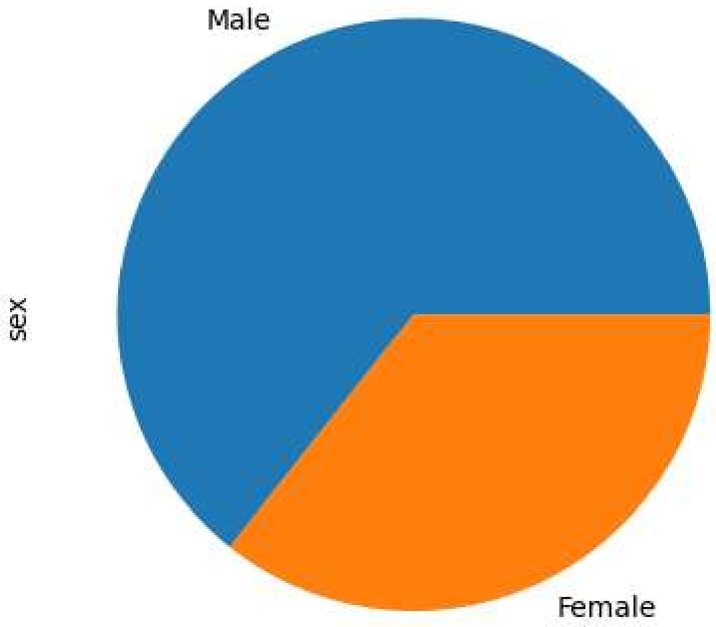
```
In [22]: sns.countplot(y=tips.sex)
```

```
Out[22]: <Axes: xlabel='count', ylabel='sex'>
```



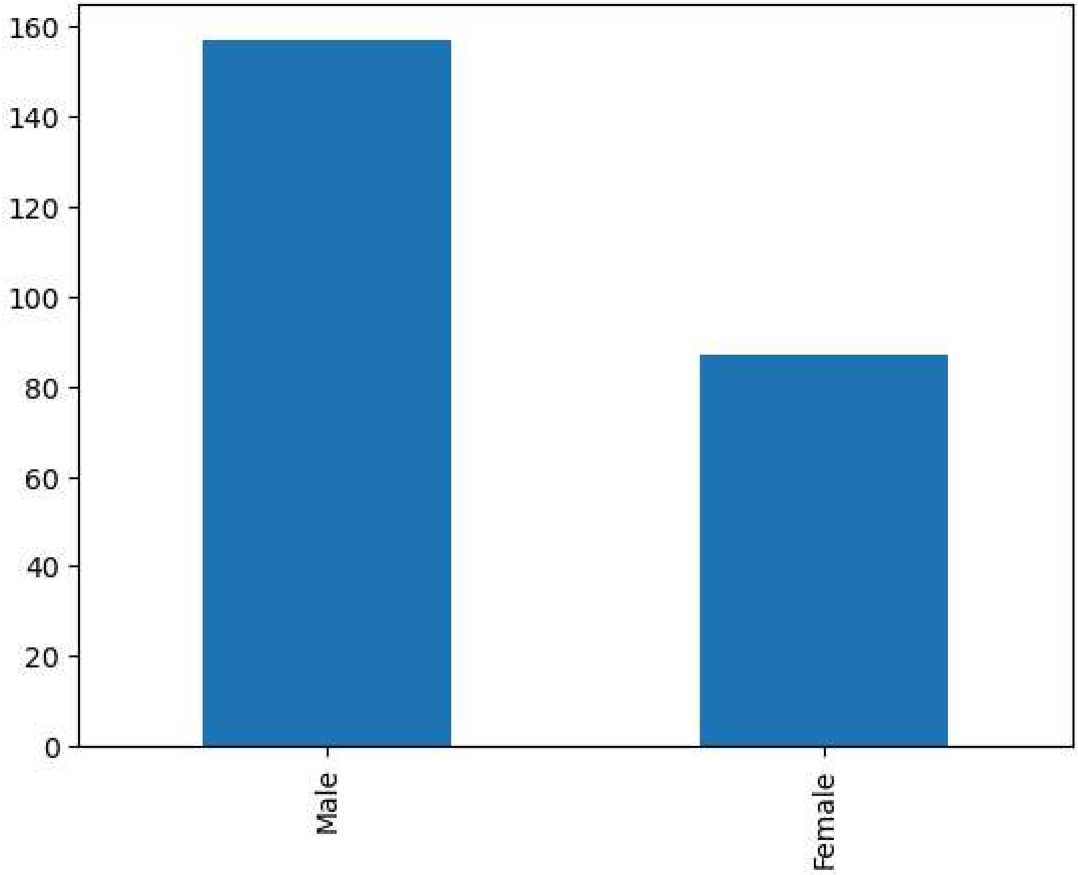
```
In [23]: tips.sex.value_counts().plot(kind='pie')
```

```
Out[23]: <Axes: ylabel='sex'>
```



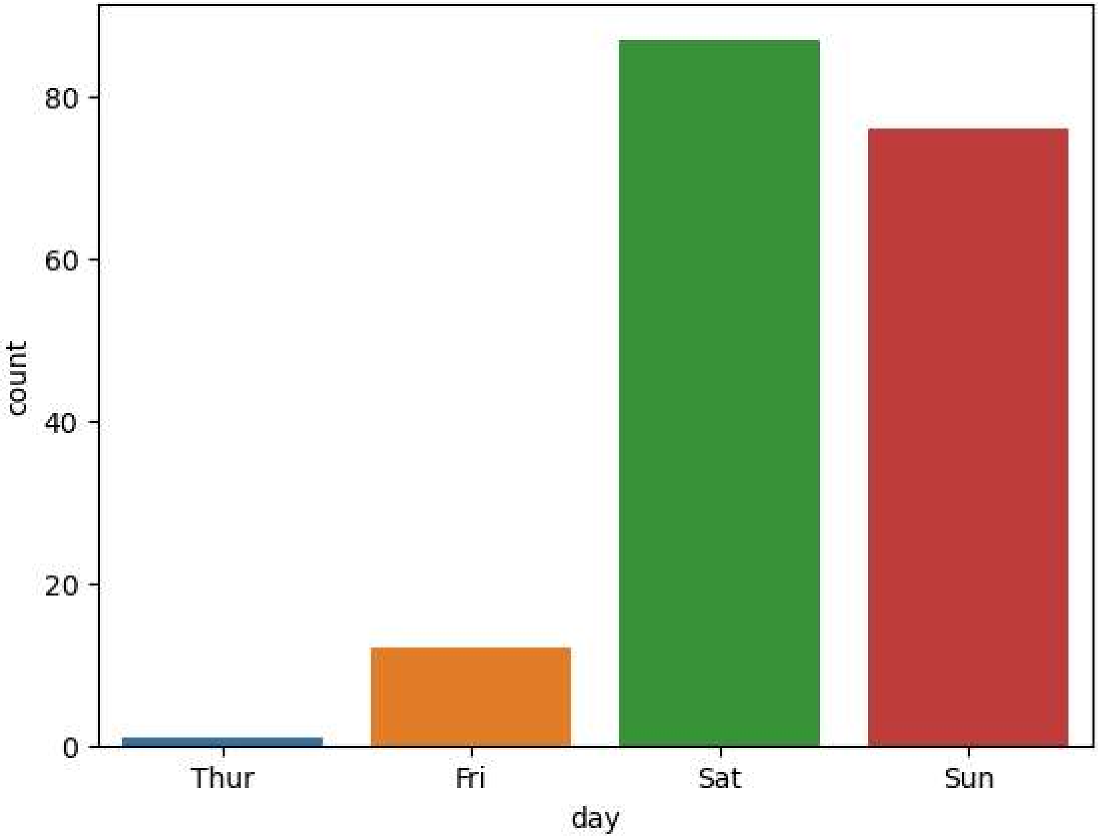
```
In [24]: tips.sex.value_counts().plot(kind='bar')
```

```
Out[24]: <Axes: >
```



```
In [29]: sns.countplot(x='day', data=tips[tips.time=='Dinner'])
```

```
Out[29]: <Axes: xlabel='day', ylabel='count'>
```



```
In [ ]:
```

In [1]:

```
import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data.csv')
df
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: 608.0 bytes
```

In [2]:

```
df.dropna(inplace=True)
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: 608.0 bytes
```

In [3]:

```
df.describe()
```

Out[3]:

	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	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [4]:

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

In [6]:

```
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_
```

In [7]:

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[7]:

```
LinearRegression()
```

In [8]:

```
model.score(x_train,y_train)
```

Out[8]:
```
0.9411949620562126
```

In [9]:
```
model.score(x_test,y_test)
```

Out[9]:
```
0.988169515729126
```

In [10]:
```
model.coef_
```

Out[10]:
```
array([[9312.57512673]])
```

In [11]:
```
model.intercept_
```

Out[11]:
```
array([26780.09915063])
```

In [12]:
```
import pickle
pickle.dump(model,open('SalaryPred.model','wb'))
```

In [13]:
```
model=pickle.load(open('SalaryPred.model','rb'))
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: 6
```

In [15]:
```
print("Estimated Salary for {} years of experience is {}: " .format(yr_of_exp,Salar
```

```
Estimated Salary for 6.0 years of experience is [[82655.549911]]:
```

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 [3]:

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

```
Out[3]: 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],
               [    48,  29000],
               [    45,  22000],
               [    47,  49000],
               [    48,  41000],
               [    45,  22000],
               [    46,  23000],
               [    47,  20000],
               [    49,  28000],
               [    47,  30000],
               [    29,  43000],
               [    31,  18000],
               [    31,  74000],
               [    27, 137000],
               [    21,  16000],
               [    28,  44000],
               [    27,  90000],
               [    35,  27000],
               [    33,  28000],
               [    30,  49000],
               [    26,  72000],
               [    27,  31000],
               [    27,  17000],
               [    33,  51000],
               [    35, 108000],
               [    30,  15000],
               [    28,  84000],
               [    23,  20000],
               [    25,  79000],
               [    27,  54000],
               [    30, 135000],
               [    31,  89000],
               [    24,  32000],
               [    18,  44000],
               [    29,  83000],
               [    35,  23000],
               [    27,  58000],
               [    24,  55000],
               [    23,  48000],
               [    28,  79000],
               [    22,  18000],
               [    32, 117000],
               [    27,  20000],
               [    25,  87000],
               [    23,  66000],
               [    32, 120000],
               [    59,  83000],
               [    24,  58000],
               [    24,  19000],
               [    23,  82000],
               [    22,  63000],
               [    31,  68000],
               [    25,  80000],
               [    24,  27000],
               [    20,  23000],
               [    33, 113000],
               [    32,  18000],
               [    34, 112000],
               [    18,  52000],
               [    22,  27000],
               [    28,  87000],
               [    26,  17000],
               [    30,  80000],
               [    39,  42000],
               [    20,  49000],
               [    35,  88000],
               [    30,  62000],
               [    31, 118000],
               [    24,  55000],
               [    28,  85000],
               [    26,  81000],
               [    35,  50000],
               [    22,  81000],
               [    30, 116000],
               [    26,  15000],
               [    29,  28000],
               [    29,  83000],
               [    35,  44000],
               [    35,  25000],
               [    28, 123000],
               [    35,  73000],
               [    28,  37000],
               [    27,  88000],
               [    28,  59000],
               [    32,  86000],
               [    33, 149000],
               [    19,  21000],
               [    21,  72000],
               [    26,  35000],
               [    27,  89000],
               [    26,  86000],
               [    38,  80000],
               [    39,  71000],
               [    37,  71000],
               [    38,  61000],
               [    37,  55000],
               [    42,  80000],
               [    40,  57000],
               [    35,  75000],
               [    36,  52000],
               [    40,  59000],
               [    41,  59000],
               [    36,  75000],
               [    37,  72000],
               [    40,  75000],
               [    35,  53000],
               [    41,  51000],
               [    39,  61000],
               [    42,  65000],
               [    26,  32000],
               [    30,  17000],
               [    26,  84000],
               [    31,  58000],
               [    33,  31000],
               [    30,  87000],
               [    21,  68000],
               [    28,  55000],
               [    23,  63000],
               [    20,  82000],
               [    30, 107000],
               [    28,  59000],
               [    19,  25000],
               [    19,  85000],
               [    18,  68000],
               [    35,  59000],
               [    30,  89000],
               [    34,  25000],
               [    24,  89000],
               [    27,  96000],
               [    41,  30000],
               [    29,  61000],
               [    20,  74000],
               [    26,  15000],
               [    41,  45000],
               [    31,  76000],
               [    36,  50000],
               [    40,  47000],
               [    31,  15000],
               [    46,  59000],
               [    29,  75000],
               [    26,  30000],
               [    32, 135000],
               [    32, 100000],
               [    25,  90000],
               [    37,  33000],
               [    35,  38000],
               [    33,  69000],
               [    18,  86000],
               [    22,  55000],
               [    35,  71000],
               [    29, 148000],
               [    29,  47000],
               [    21,  88000],
               [    34, 115000],
               [    26, 118000],
               [    34,  43000],
               [    34,  72000],
               [    23,  28000],
               [    35,  47000],
               [    25,  22000],
               [    24,  23000],
               [    31,  34000],
               [    26,  16000],
               [    31,  71000],
               [    32, 117000],
               [    33,  43000],
               [    33,  60000],
               [    31,  66000],
               [    20,  82000],
               [    33,  41000],
               [    35,  72000],
               [    28,  32000],
               [    24,  84000],
               [    19,  26000],
               [    29,  43000],
               [    19,  70000],
               [    28,  89000],
               [    34,  43000],
               [    30,  79000],
               [    20,  36000],
               [    26,  80000],
               [    35,  22000],
               [    35,  39000],
               [    49,  74000],
               [    39, 134000],
               [    41,  71000],
               [    58, 101000],
               [    47,  47000],
               [    55, 130000],
               [    52, 114000],
               [    40, 142000],
               [    46,  22000],
               [    48,  96000],
               [    52, 150000],
               [    59,  42000],
               [    35,  58000],
               [    47,  43000],
               [    60, 108000],
               [    49,  65000],
               [    40,  78000],
               [    46,  96000],
               [    59, 143000],
               [    41,  80000],
               [    35,  91000],
               [    37, 144000],
               [    60, 102000],
               [    35,  60000],
               [    37,  53000],
               [    36, 126000],
               [    56, 133000],
               [    40,  72000],
               [    42,  80000],
               [    35, 147000],
               [    39,  42000],
               [    40, 107000],
               [    49,  86000],
               [    38, 112000],
               [    46,  79000],
               [    40,  57000],
               [    37,  80000],
               [    46,  82000],
               [    53, 143000],
               [    42, 149000],
               [    38,  59000],
               [    50,  88000],
               [    56, 104000],
               [    41,  72000],
               [    51, 146000],
               [    35,  50000],
               [    57, 122000],
               [    41,  52000],
               [    35,  97000],
               [    44,  39000],
               [    37,  52000],
               [    48, 134000],
               [    37, 146000],
               [    50,  44000],
               [    52,  90000],
               [    41,  72000],
               [    40,  57000],
               [    58,  95000],
               [    45, 131000],
               [    35,  77000],
               [    36, 144000],
               [    55, 125000],
               [    35,  72000],
               [    48,  90000],
               [    42, 108000],
               [    40,  75000],
               [    37,  74000],
               [    47, 144000],
               [    40,  61000],
               [    43, 133000],
               [    59,  76000],
               [    60,  42000],
               [    39, 106000],
               [    57,  26000],
               [    57,  74000],
               [    38,  71000],
               [    49,  88000],
               [    52,  38000],
               [    50,  36000],
               [    59,  88000],
               [    35,  61000],
               [    37,  70000],
               [    52,  21000],
               [    48, 141000],
               [    37,  93000],
               [    37,  62000],
               [    48, 138000],
               [    41,  79000],
               [    37,  78000],
               [    39, 134000],
               [    49,  89000],
               [    55,  39000],
               [    37,  77000],
               [    35,  57000],
               [    36,  63000],
               [    42,  73000],
               [    43, 112000],
               [    45,  79000],
               [    46, 117000],
               [    58,  38000],
               [    48,  74000],
               [    37, 137000],
               [    37,  79000],
               [    40,  60000],
               [    42,  54000],
               [    51, 134000],
               [    47, 113000],
               [    36, 125000],
               [    38,  50000],
               [    42,  70000],
               [    39,  96000],
               [    38,  50000],
               [    49, 141000],
               [    39,  79000],
               [    39,  75000],
               [    54, 104000],
               [    35,  55000],
               [    45,  32000],
               [    36,  60000],
               [    52, 138000],
               [    53,  82000],
               [    41,  52000],
               [    48,  30000],
               [    48, 131000],
               [    41,  60000],
               [    41,  72000],
               [    42,  75000],
               [    36, 118000],
               [    47, 107000],
               [    38,  51000],
               [    48, 119000],
               [    42,  65000],
               [    40,  65000],
               [    57,  60000],
               [    36,  54000],
               [    58, 144000],
               [    35,  79000],
               [    38,  55000],
               [    39, 122000],
               [    53, 104000],
               [    35,  75000],
               [    38,  65000],
               [    47,  51000],
               [    47, 105000],
               [    41,  63000],
               [    53,  72000],
               [    54, 108000],
               [    39,  77000],
               [    38,  61000],
               [    38, 113000],
               [    37,  75000],
               [    42,  90000],
               [    37,  57000],
               [    36,  99000],
               [    60,  34000],
               [    54,  70000],
               [    41,  72000],
               [    40,  71000],
               [    42,  54000],
               [    43, 129000],
               [    53,  34000],
               [    47,  50000],
               [    42,  79000],
               [    42, 104000],
               [    59,  29000],
               [    58,  47000],
               [    46,  88000],
               [    38,  71000],
               [    54,  26000],
               [    60,  46000],
               [    60,  83000],
               [    39,  73000],
               [    59, 130000],
               [    37,  80000],
               [    46,  32000],
               [    46,  74000],
               [    42,  53000],
               [    41,  87000],
               [    58,  23000],
               [    42,  64000],
               [    48,  33000],
               [    44, 139000],
               [    49,  28000],
               [    57,  33000],
               [    56,  60000],
               [    49,  39000],
               [    39,  71000],
               [    47,  34000],
               [    48,  35000],
               [    48,  33000],
               [    47,  23000],
               [    45,  45000],
               [    60,  42000],
               [    39,  59000],
               [    46,  41000],
               [    51,  23000],
               [    50,  20000],
               [    36,  33000],
               [    49,  36000]], dtype=int64)
```

In [4]:

```
label
```

Out[4]:

```
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, 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, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 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, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 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, 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, 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, 1, 0, 1,
       1, 1, 0, 1], dtype=int64)
```

In [9]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
for i in range(1,401):
 x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,random
 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
Test 0.6625 Train0.6375 Random State 37
Test 0.7 Train0.628125 Random State 39
Test 0.7 Train0.628125 Random State 40
Test 0.65 Train0.640625 Random State 42
Test 0.725 Train0.621875 Random State 46
Test 0.65 Train0.640625 Random State 48
Test 0.675 Train0.634375 Random State 50
Test 0.65 Train0.640625 Random State 51
Test 0.65 Train0.640625 Random State 54
Test 0.7 Train0.634375 Random State 55
Test 0.65 Train0.640625 Random State 56
Test 0.6625 Train0.6375 Random State 58
Test 0.6875 Train0.63125 Random State 59
Test 0.7 Train0.628125 Random State 60
Test 0.6625 Train0.6375 Random State 62
Test 0.6875 Train0.63125 Random State 63
Test 0.65 Train0.640625 Random State 66
Test 0.7 Train0.628125 Random State 70
Test 0.65 Train0.640625 Random State 74
Test 0.65 Train0.640625 Random State 75
Test 0.6875 Train0.63125 Random State 76
Test 0.6875 Train0.63125 Random State 80
Test 0.675 Train0.634375 Random State 81
Test 0.875 Train0.8375 Random State 82
Test 0.7 Train0.628125 Random State 83
Test 0.675 Train0.634375 Random State 84
Test 0.675 Train0.634375 Random State 86
Test 0.65 Train0.640625 Random State 87
Test 0.675 Train0.634375 Random State 90
Test 0.65 Train0.640625 Random State 91
Test 0.7 Train0.628125 Random State 93
Test 0.7375 Train0.61875 Random State 94
Test 0.65 Train0.640625 Random State 97
Test 0.7 Train0.628125 Random State 99
Test 0.675 Train0.634375 Random State 101
Test 0.6625 Train0.6375 Random State 102
Test 0.725 Train0.621875 Random State 103
Test 0.65 Train0.640625 Random State 106
Test 0.65 Train0.640625 Random State 109
Test 0.75 Train0.615625 Random State 114
Test 0.675 Train0.634375 Random State 116
Test 0.65 Train0.640625 Random State 117
Test 0.675 Train0.634375 Random State 119
Test 0.65 Train0.640625 Random State 120
Test 0.6625 Train0.6375 Random State 121
Test 0.725 Train0.621875 Random State 125
Test 0.65 Train0.640625 Random State 127
Test 0.65 Train0.640625 Random State 128
Test 0.6875 Train0.63125 Random State 129
Test 0.6875 Train0.63125 Random State 130
Test 0.6625 Train0.6375 Random State 132
Test 0.6875 Train0.63125 Random State 133
Test 0.675 Train0.634375 Random State 134
Test 0.675 Train0.634375 Random State 138
Test 0.7 Train0.628125 Random State 139
Test 0.7125 Train0.63125 Random State 141
Test 0.725 Train0.621875 Random State 142
Test 0.6625 Train0.6375 Random State 143
Test 0.6625 Train0.6375 Random State 145
Test 0.7125 Train0.625 Random State 150
Test 0.65 Train0.640625 Random State 152
Test 0.6625 Train0.6375 Random State 154
Test 0.675 Train0.634375 Random State 155
Test 0.8875 Train0.834375 Random State 158
Test 0.6625 Train0.6375 Random State 159
Test 0.7125 Train0.625 Random State 161
Test 0.675 Train0.634375 Random State 162
Test 0.6625 Train0.6375 Random State 163
Test 0.65 Train0.640625 Random State 165
Test 0.6625 Train0.6375 Random State 169
Test 0.675 Train0.634375 Random State 170
Test 0.7125 Train0.625 Random State 173
Test 0.65 Train0.640625 Random State 176
Test 0.6625 Train0.6375 Random State 178
Test 0.6625 Train0.6375 Random State 179
Test 0.6625 Train0.6375 Random State 180
Test 0.6625 Train0.6375 Random State 181
Test 0.65 Train0.640625 Random State 184
Test 0.6625 Train0.6375 Random State 185
Test 0.675 Train0.634375 Random State 188
Test 0.7375 Train0.61875 Random State 189
Test 0.7 Train0.628125 Random State 192
Test 0.65 Train0.640625 Random State 193
Test 0.7 Train0.628125 Random State 194
Test 0.65 Train0.640625 Random State 195
Test 0.6625 Train0.6375 Random State 196
Test 0.675 Train0.634375 Random State 198
Test 0.8875 Train0.8375 Random State 199
Test 0.6875 Train0.63125 Random State 204
Test 0.6625 Train0.6375 Random State 209
Test 0.7 Train0.628125 Random State 211
Test 0.65 Train0.640625 Random State 212
Test 0.6625 Train0.6375 Random State 215
Test 0.6625 Train0.6375 Random State 217
Test 0.6875 Train0.63125 Random State 220
Test 0.6625 Train0.6375 Random State 223
Test 0.6625 Train0.6375 Random State 225
Test 0.6625 Train0.6375 Random State 226
Test 0.6875 Train0.63125 Random State 229
Test 0.65 Train0.640625 Random State 232
Test 0.7125 Train0.625 Random State 233
Test 0.6625 Train0.6375 Random State 234
Test 0.6625 Train0.6375 Random State 235
Test 0.6875 Train0.63125 Random State 238
Test 0.725 Train0.621875 Random State 239
Test 0.65 Train0.640625 Random State 241
Test 0.725 Train0.621875 Random State 242
Test 0.6625 Train0.6375 Random State 244
Test 0.675 Train0.634375 Random State 245
Test 0.6875 Train0.63125 Random State 246
Test 0.7 Train0.628125 Random State 247
Test 0.6875 Train0.63125 Random State 248
Test 0.65 Train0.640625 Random State 251
Test 0.7 Train0.628125 Random State 252
Test 0.65 Train0.640625 Random State 253
Test 0.675 Train0.634375 Random State 255
Test 0.75 Train0.615625 Random State 257
Test 0.7 Train0.628125 Random State 260
Test 0.6625 Train0.6375 Random State 261
Test 0.65 Train0.640625 Random State 263
Test 0.6625 Train0.6375 Random State 265
Test 0.8625 Train0.840625 Random State 266
Test 0.6875 Train0.63125 Random State 269
Test 0.6625 Train0.6375 Random State 275
Test 0.7 Train0.628125 Random State 276
Test 0.6625 Train0.6375 Random State 277
Test 0.7 Train0.628125 Random State 278
Test 0.7125 Train0.625 Random State 279
Test 0.6875 Train0.63125 Random State 282
Test 0.6875 Train0.63125 Random State 283
Test 0.7125 Train0.625 Random State 287
Test 0.6625 Train0.6375 Random State 292
Test 0.65 Train0.640625 Random State 293
Test 0.6625 Train0.6375 Random State 294
Test 0.675 Train0.634375 Random State 296
Test 0.675 Train0.634375 Random State 300
Test 0.675 Train0.634375 Random State 302
Test 0.6625 Train0.6375 Random State 303
Test 0.8625 Train0.834375 Random State 305
Test 0.6875 Train0.63125 Random State 306
Test 0.7 Train0.628125 Random State 310
Test 0.7125 Train0.625 Random State 311
Test 0.8625 Train0.834375 Random State 313
Test 0.9125 Train0.834375 Random State 314
Test 0.7 Train0.628125 Random State 315
Test 0.6625 Train0.6375 Random State 317
Test 0.7625 Train0.6125 Random State 318
Test 0.6625 Train0.6375 Random State 319
Test 0.65 Train0.640625 Random State 321
Test 0.7125 Train0.625 Random State 322
Test 0.675 Train0.634375 Random State 323
Test 0.6625 Train0.6375 Random State 325
Test 0.7125 Train0.625 Random State 327
Test 0.6625 Train0.6375 Random State 328
Test 0.7 Train0.628125 Random State 329
Test 0.65 Train0.640625 Random State 330
Test 0.65 Train0.640625 Random State 332
Test 0.675 Train0.634375 Random State 336
Test 0.6875 Train0.63125 Random State 340
Test 0.65 Train0.640625 Random State 344
Test 0.6625 Train0.6375 Random State 345
Test 0.7 Train0.628125 Random State 346
Test 0.65 Train0.640625 Random State 348
Test 0.725 Train0.621875 Random State 349
Test 0.6875 Train0.63125 Random State 350
Test 0.675 Train0.634375 Random State 352
Test 0.725 Train0.621875 Random State 353
Test 0.675 Train0.634375 Random State 354
Test 0.6875 Train0.63125 Random State 355
Test 0.6625 Train0.6375 Random State 356
Test 0.7375 Train0.61875 Random State 357
Test 0.6625 Train0.6375 Random State 358
Test 0.6625 Train0.6375 Random State 359
Test 0.7 Train0.628125 Random State 360
Test 0.65 Train0.640625 Random State 361
Test 0.6625 Train0.6375 Random State 362
Test 0.65 Train0.640625 Random State 363
Test 0.6625 Train0.6375 Random State 364
Test 0.6875 Train0.63125 Random State 365
Test 0.6625 Train0.6375 Random State 366
Test 0.6625 Train0.6375 Random State 368
Test 0.65 Train0.640625 Random State 370
Test 0.725 Train0.621875 Random State 371
Test 0.65 Train0.640625 Random State 373
Test 0.7 Train0.628125 Random State 376
Test 0.6875 Train0.63125 Random State 378
Test 0.675 Train0.634375 Random State 379
Test 0.65 Train0.640625 Random State 387
Test 0.6625 Train0.6375 Random State 393
Test 0.675 Train0.634375 Random State 396
Test 0.7 Train0.628125 Random State 397
Test 0.7125 Train0.625 Random State 400
```

In [10]:

```
x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,train_s
finalModel=LogisticRegression()
finalModel.fit(x_train,y_train)
```

Out[10]:

```
▾ LogisticRegression
LogisticRegression()
```

In [11]:

```
print(finalModel.score(x_train,y_train))
print(finalModel.score(x_test,y_test))
```

```
0.675
0.5125
```

In [12]:

```
from sklearn.metrics import classification_report
print(classification_report(label,finalModel.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.64      1.00      0.78       257
           1       0.00      0.00      0.00       143

    accuracy                           0.64       400
   macro avg       0.32      0.50      0.39       400
weighted avg       0.41      0.64      0.50       400
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
In [ ]:
```

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df=pd.read_csv('Mall_Customers.csv')
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 [2]:

```
df.head()
```

Out[2]:

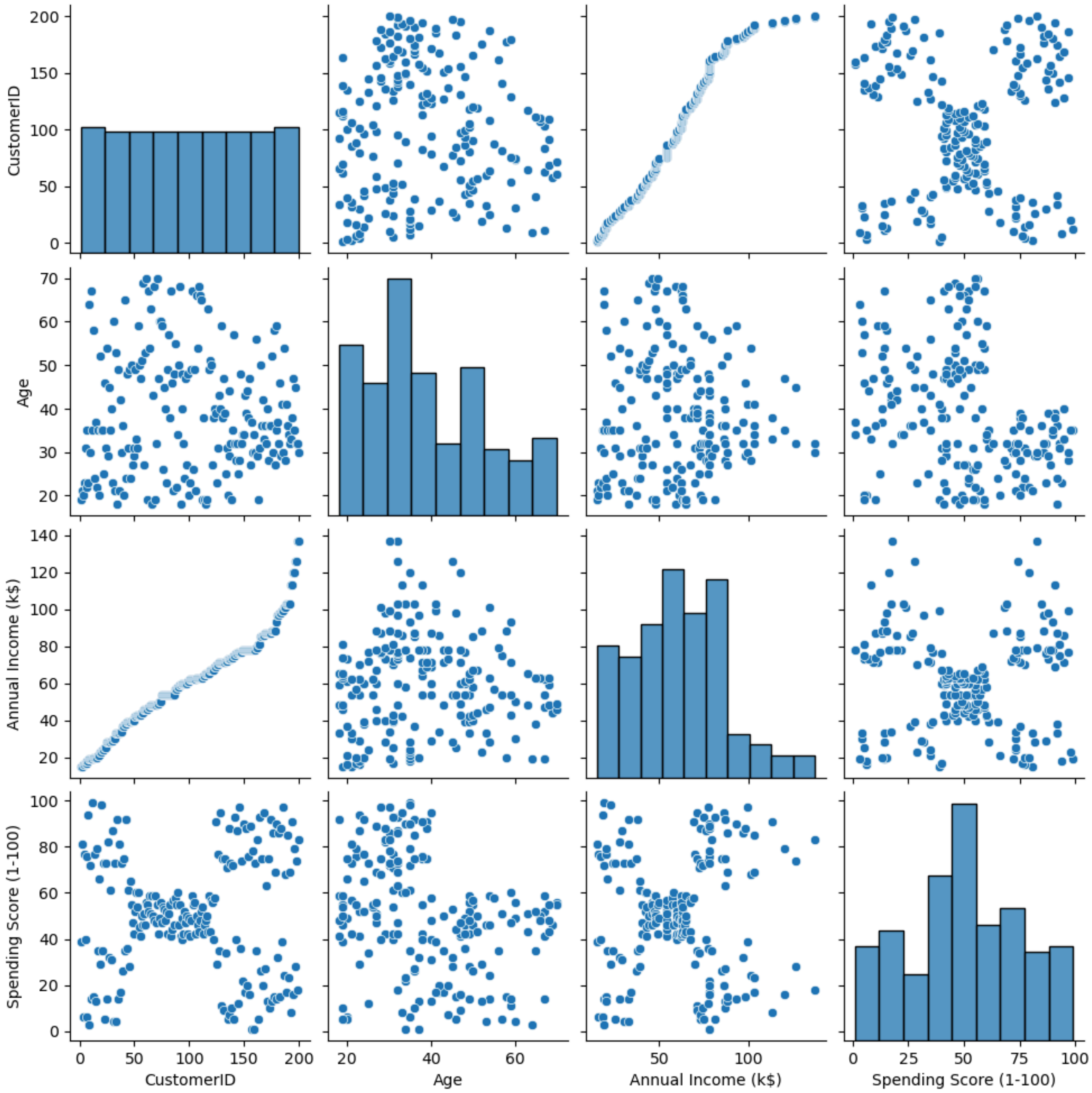
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [3]:

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

Out[3]:

```
<seaborn.axisgrid.PairGrid at 0x2452f751390>
```



In [4]:
```
features=df.iloc[:,[3,4]].values
from sklearn.cluster import KMeans
model=KMeans(n_clusters=5)
model.fit(features)
KMeans(n_clusters=5)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[4]:
```
▾        KMeans
KMeans(n_clusters=5)
```

In [5]:
```
Final=df.iloc[:,[3,4]]
Final['label']=model.predict(features)
Final.head()
```

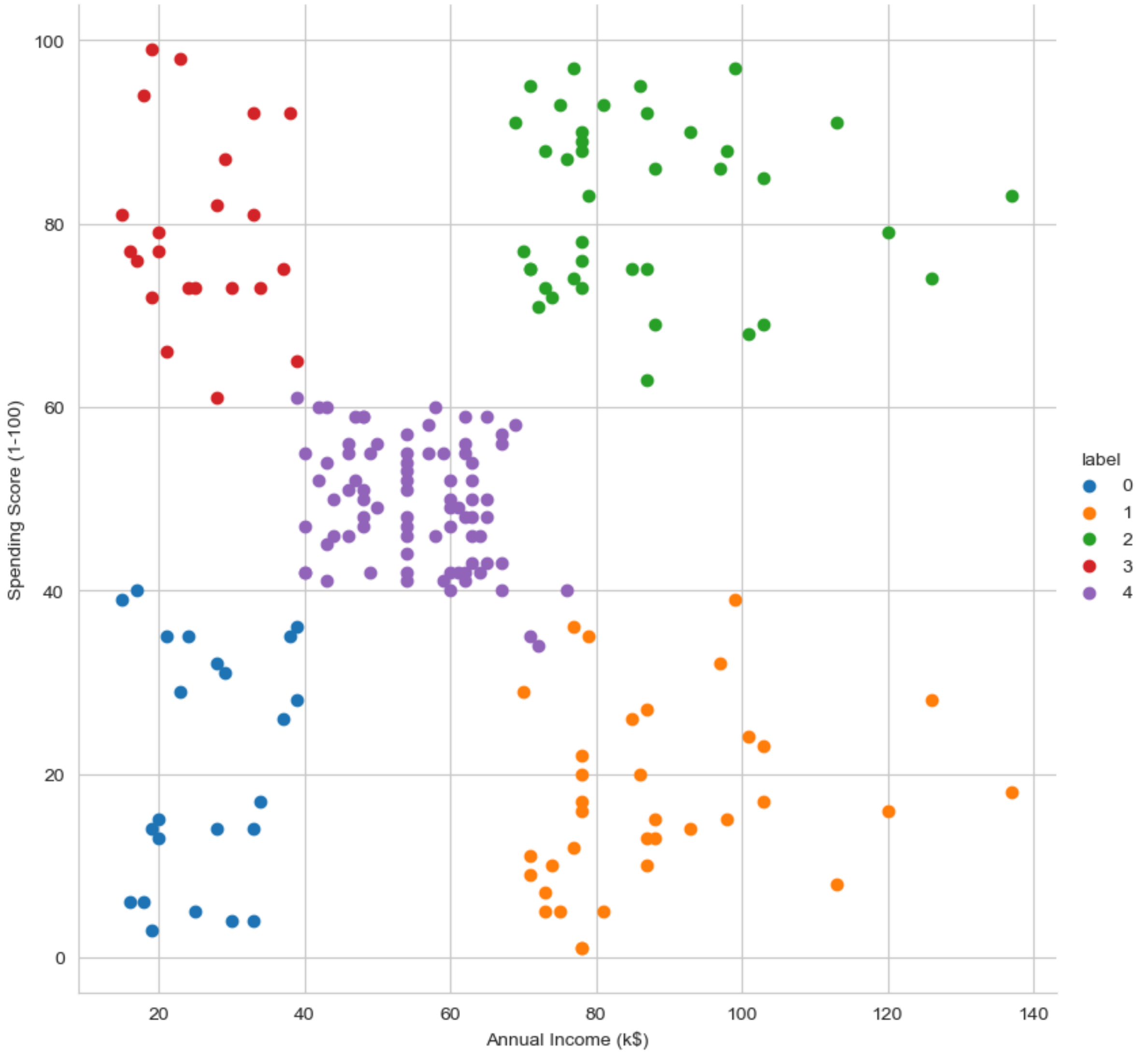
```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_13424\470183701.py:2: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  Final['label']=model.predict(features)
```

Out[5]:

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

In [6]:

```
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 [9]:

```
features_el=df.iloc[:,[2,3,4]].values
from sklearn.cluster import KMeans
wcss=[]
```

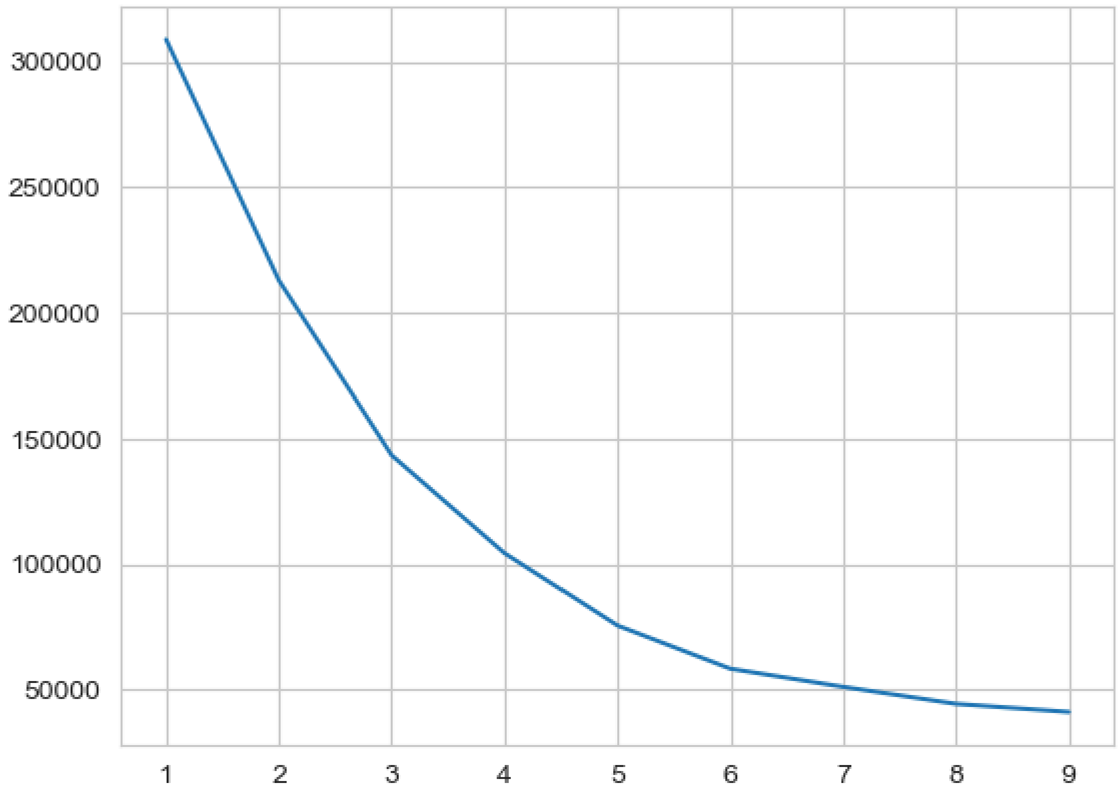
```
for i in range(1,10):
 model=KMeans(n_clusters=i)
 model.fit(features_el)
 wcss.append(model.inertia_)
plt.plot(range(1,10),wcss)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
```

```
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x245339ac3d0>]
```



In [ ]:

In [2]:

```
import numpy as np
import pandas as pd
df=pd.read_csv('Iris.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [3]:

```
df.variety.value_counts()
```

Out[3]:

```
Setosa        50
Versicolor    50
Virginica     50
Name: variety, dtype: int64
```

In [4]:

```
df.head()
```

Out[4]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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

In [9]:

```
features=df.iloc[:,:-1].values
label=df.iloc[:,4].values
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
xtrain,xtest,ytrain,ytest=train_test_split(features,label,test_size=0.2,random_stat
model_KNN=KNeighborsClassifier(n_neighbors=5)
model_KNN.fit(xtrain,ytrain)
```

Out[9]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [10]:

```
print(model_KNN.score(xtrain,ytrain))
print(model_KNN.score(xtest,ytest))
```

```
0.95
0.9666666666666667
```

In [11]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(label,model_KNN.predict(features))
```

```
Out[11]: array([[50,  0,  0],
                [ 0, 45,  5],
                [ 0,  2, 48]], dtype=int64)
```

```
In [12]: from sklearn.metrics import classification_report
         print(classification_report(label,model_KNN.predict(features)))
```

```
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        50
  Versicolor       0.96      0.90      0.93        50
   Virginica       0.91      0.96      0.93        50

    accuracy                           0.95       150
   macro avg       0.95      0.95      0.95       150
weighted avg       0.95      0.95      0.95       150
```

```
In [ ]:
```

```
Shivani R J
240701500
CSE
```

```
import numpy as np
from scipy import stats
# Sample data
marks = np.array([72, 68, 75, 70, 74, 69, 71, 73, 70, 72])
# Hypothesized mean
mu_0 = 70
# One-sample t-test
t_stat, p_value = stats.ttest_1samp(marks, mu_0)
print(f"T-statistic: {t_stat:.3f}")
print(f"P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
  print("Reject Null Hypothesis → Mean is significantly different from 70")
else:
  print("Fail to Reject Null Hypothesis → No significant difference")
```

```
T-statistic: 1.993
P-value: 0.0774
Fail to Reject Null Hypothesis → No significant difference
```

```
Shivani R J
240701500
CSE
```

```
import numpy as np
from math import sqrt
from scipy.stats import norm
x_bar = 51.2
mu_0 = 50
sigma = 3
n = 36
z_stat = (x_bar - mu_0) / (sigma / sqrt(n))
p_value = 2 * (1 - norm.cdf(abs(z_stat)))
print(f"Z-statistic: {z_stat:.3f}")
print("P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
 print("Reject Null Hypothesis → Mean is significantly different from 50 g.")
else:
 print("Fail to Reject Null Hypothesis → No significant difference.")
```

```
Z-statistic: 2.400
P-value: {p_value:.4f}
Reject Null Hypothesis → Mean is significantly different from 50 g.
```

```
Shivani R J
240701500
CSE
```

```
import numpy as np
from scipy import stats

A = [20, 22, 23]
B = [19, 20, 18]
C = [25, 27, 26]

f_stat, p_value = stats.f_oneway(A, B, C)

print(f"F-statistic: {f_stat:.3f}")
print(f"P-value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
 print("Reject Null Hypothesis → Means are significantly different.")
else:
 print("Fail to Reject Null Hypothesis → No significant difference.")
```

```
F-statistic: 25.923
P-value: 0.0011
Reject Null Hypothesis → Means are significantly different.
```