



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

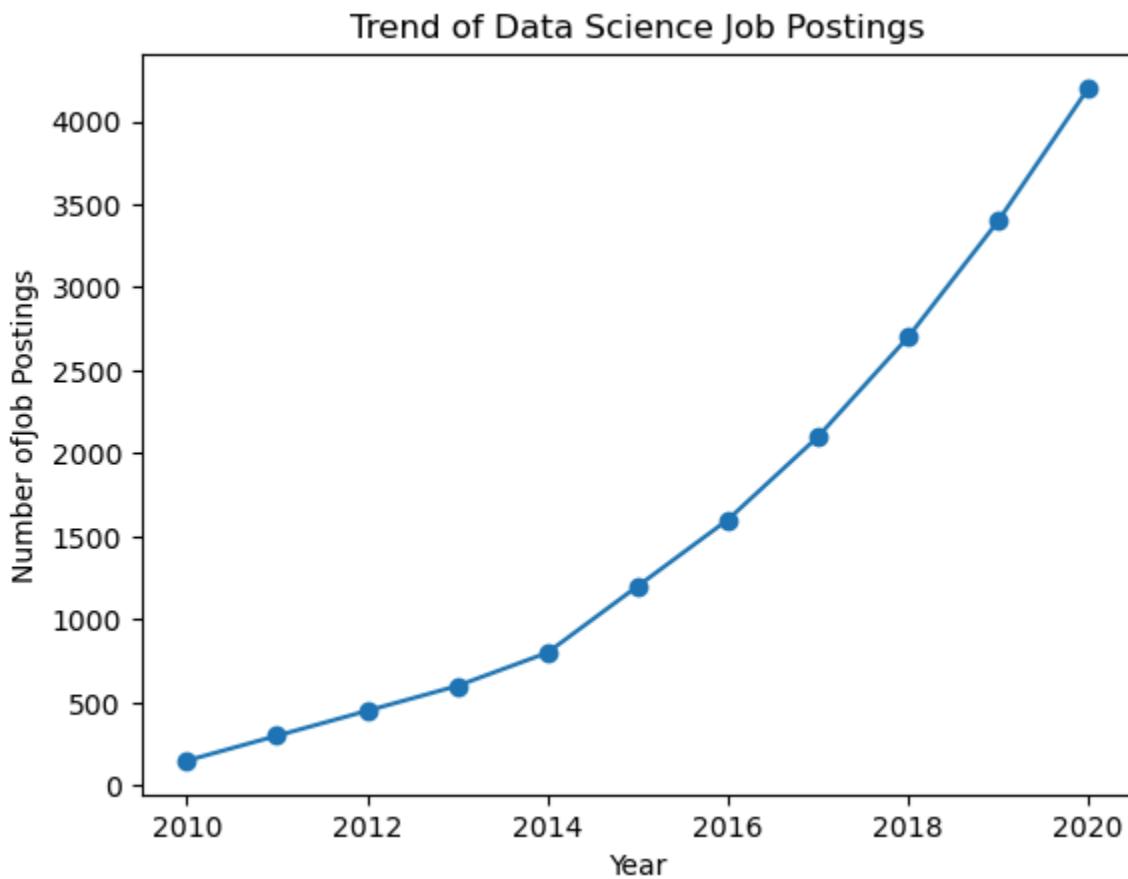
III semester II Year (2023R)

Name of the Student : Phaveen S

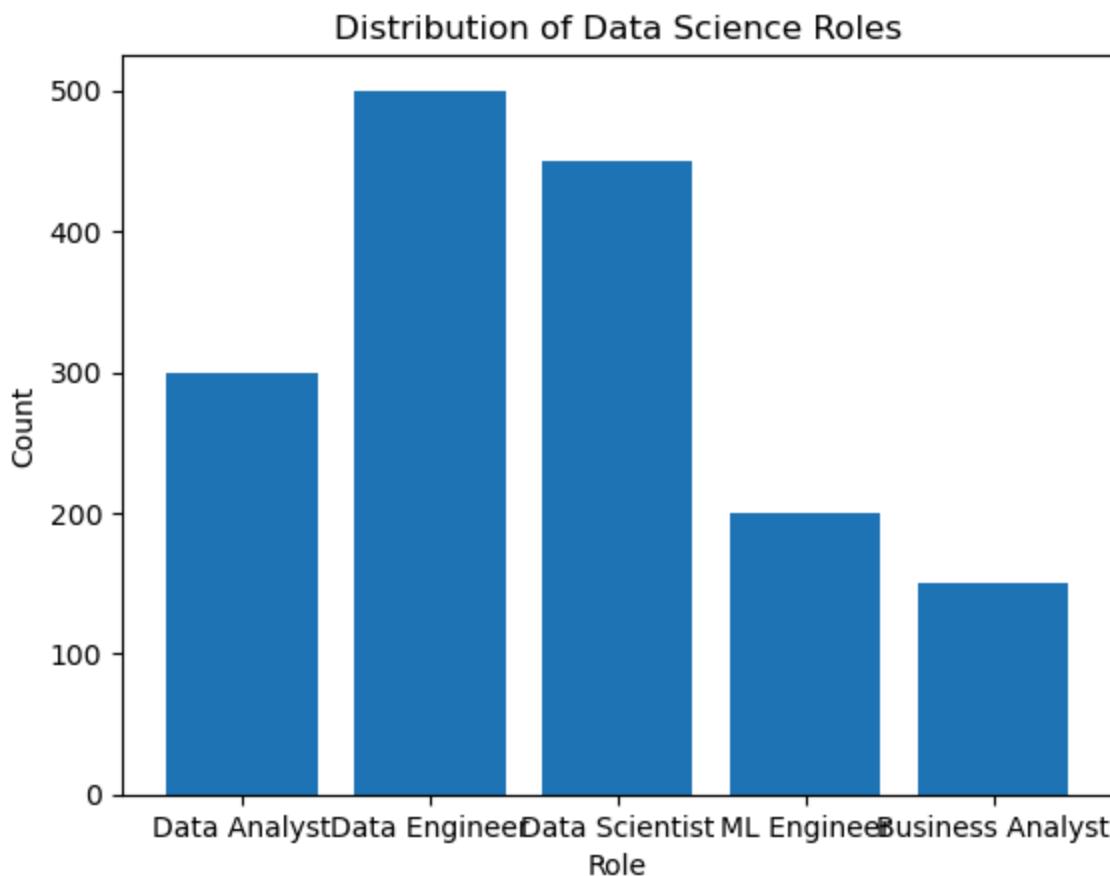
Register Number : 240701383

```
In [2]: import pandas as pd
import matplotlib.pyplot as plt
data = {'Year': list(range(2010, 2021)),
'Job Postings': [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]}

df = pd.DataFrame(data)
plt.plot(df['Year'], df['Job Postings'], marker='o')
plt.title('Trend of Data Science Job Postings')
plt.xlabel('Year')
plt.ylabel('Number of Job Postings')
plt.show()
```



```
In [4]: roles = ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML Engineer',
'Business Analyst']
counts = [300, 500, 450, 200, 150]
plt.bar(roles, counts)
plt.title('Distribution of Data Science Roles')
plt.xlabel('Role')
plt.ylabel('Count')
plt.show()
```



```
In [8]: structured_data = pd.DataFrame({
    'ID': [1, 2, 3],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35]
})
print("Structured Data:\n", structured_data)
unstructured_data = "This is an example of unstructured data. It can be a piece of text, an image, or a video file."
print("\nUnstructured Data:\n", unstructured_data)
semi_structured_data = {'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight': 68}}
print("\nSemi-structured Data:\n", semi_structured_data)
```

Structured Data:

	ID	Name	Age
0	1	Alice	25
1	2	Bob	30
2	3	Charlie	35

Unstructured Data:

This is an example of unstructured data. It can be a piece of text, an image, or a video file.

Semi-structured Data:

```
{'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight': 68}}
```

```
In [9]: from cryptography.fernet import Fernet
key = Fernet.generate_key()
```

```
f = Fernet(key)
token = f.encrypt(b"Rajalakshmi Engineering College")
token
b'...'
f.decrypt(token)
b'Rajalakshmi Engineering College'
key = Fernet.generate_key()
cipher_suite = Fernet(key)
plain_text = b"Rajalakshmi Engineering College."
cipher_text = cipher_suite.encrypt(plain_text)
decrypted_text = cipher_suite.decrypt(cipher_text)
print("Original Data:", plain_text)
print("Encrypted Data:", cipher_text)
print("Decrypted Data:", decrypted_text)
```

```
Original Data: b'Rajalakshmi Engineering College.'
Encrypted Data: b'gAAAAABoy36qv8K3SP9FFkL2oRpeWkWLVEvk63Dqp-w35qlyqsA9NQQVIm1k
0j7v4oGW8PCYvo5jxuF7ZKg5sguSI5uilUWsdXEeEMAWg1hJIja0Ze0WzGSkTkI7p40qIUIthzL8e6
x'
Decrypted Data: b'Rajalakshmi Engineering College.'
```

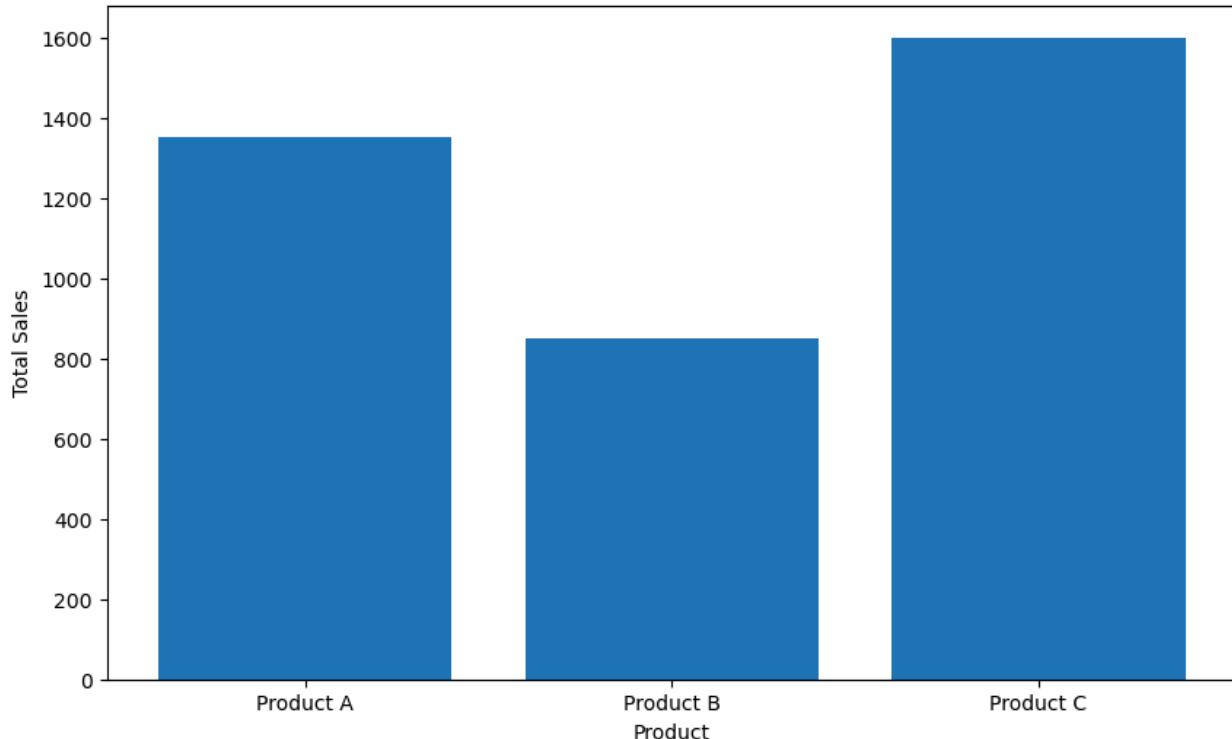


```
In [3]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Load the data into a pandas DataFrame
file_path='D:\sales_data.csv'
df = pd.read_csv(file_path)
# Display the first few rows of the DataFrame
print(df.head())
# Check for missing values
print(df.isnull().sum())
# Fill or drop missing values if necessary
df['Sales'].fillna(df['Sales'].mean(), inplace=True)
df.dropna(subset=['Product', 'Quantity', 'Region'], inplace=True)
# Summary statistics
print(df.describe())
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()
df['Date'] = pd.to_datetime(df['Date'])
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('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)
# Correlation matrix
correlation_matrix = df.corr()
print(correlation_matrix)
# Heatmap of the correlation matrix
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
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
Date          0
Product        0
Sales          0
Quantity        0
Region         0
dtype: int64
```

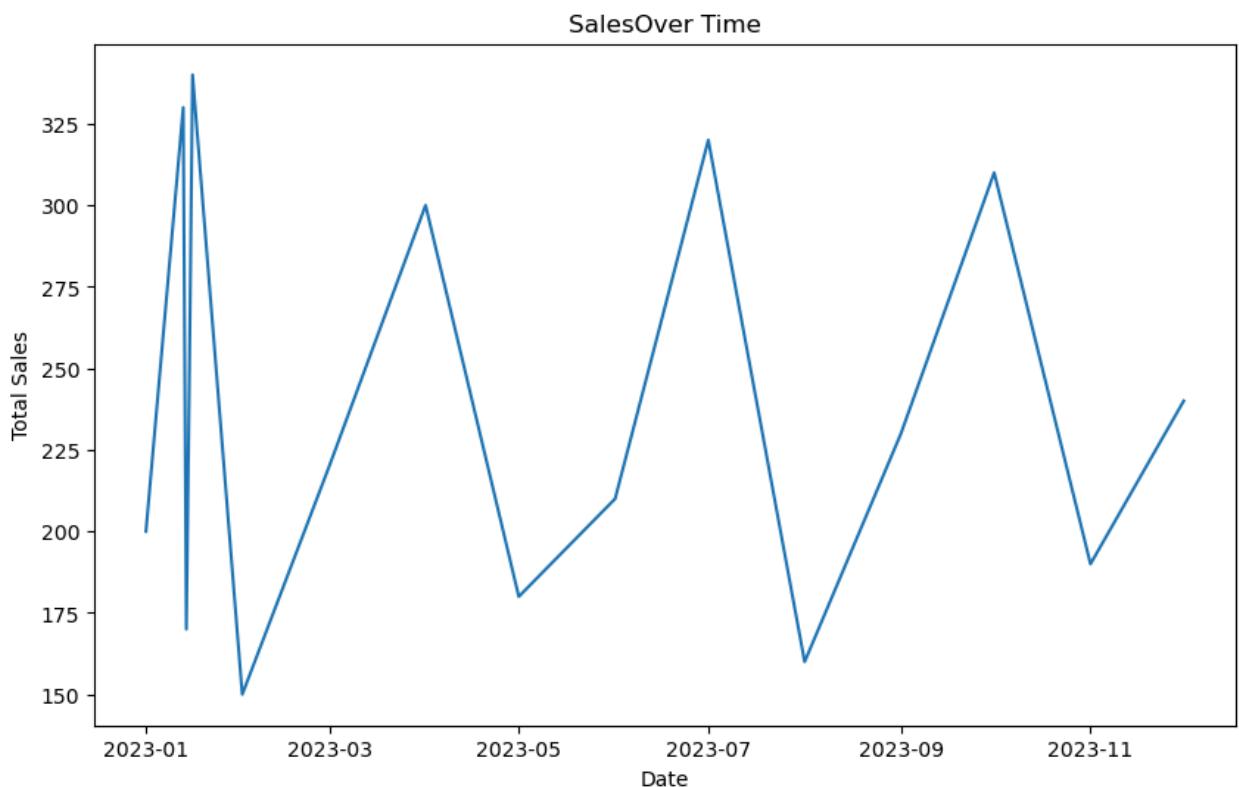
```
Sales  Quantity
count   16.000000  16.000000
mean    237.500000  5.375000
std     64.031242  1.746425
min    150.000000  3.000000
25%   187.500000  4.000000
50%   225.000000  5.500000
75%   302.500000  7.000000
max    340.000000  8.000000
Product  Sales  Quantity
0  Product A    1350       33
1  Product B     850       17
2  Product C    1600       36
```

Total Sales by Product



```
C:\Users\REC\AppData\Local\Temp\ipykernel_15972\889183971.py:27: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to ensure consistent parsing.
```

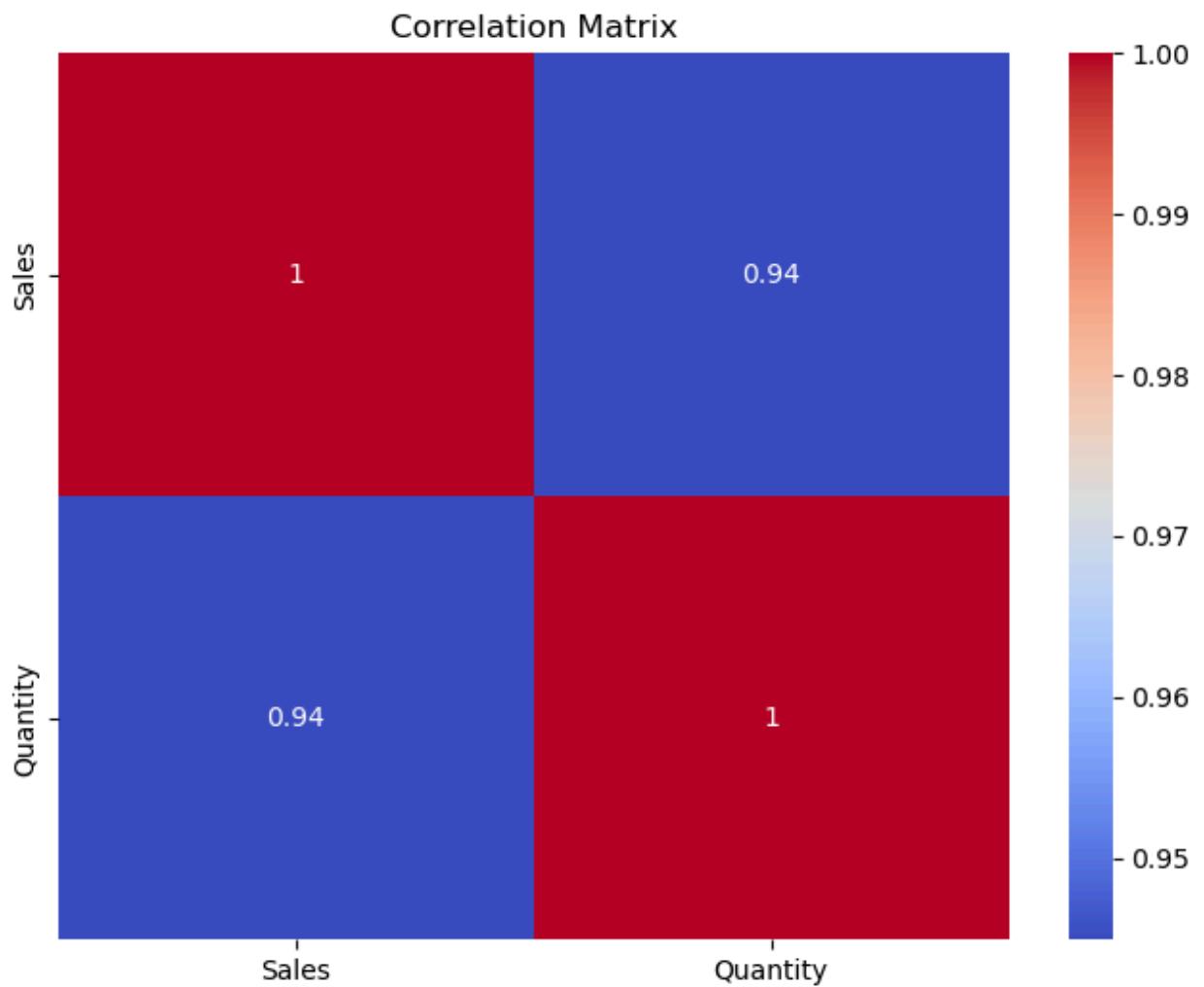
```
df['Date'] = pd.to_datetime(df['Date'])
```



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

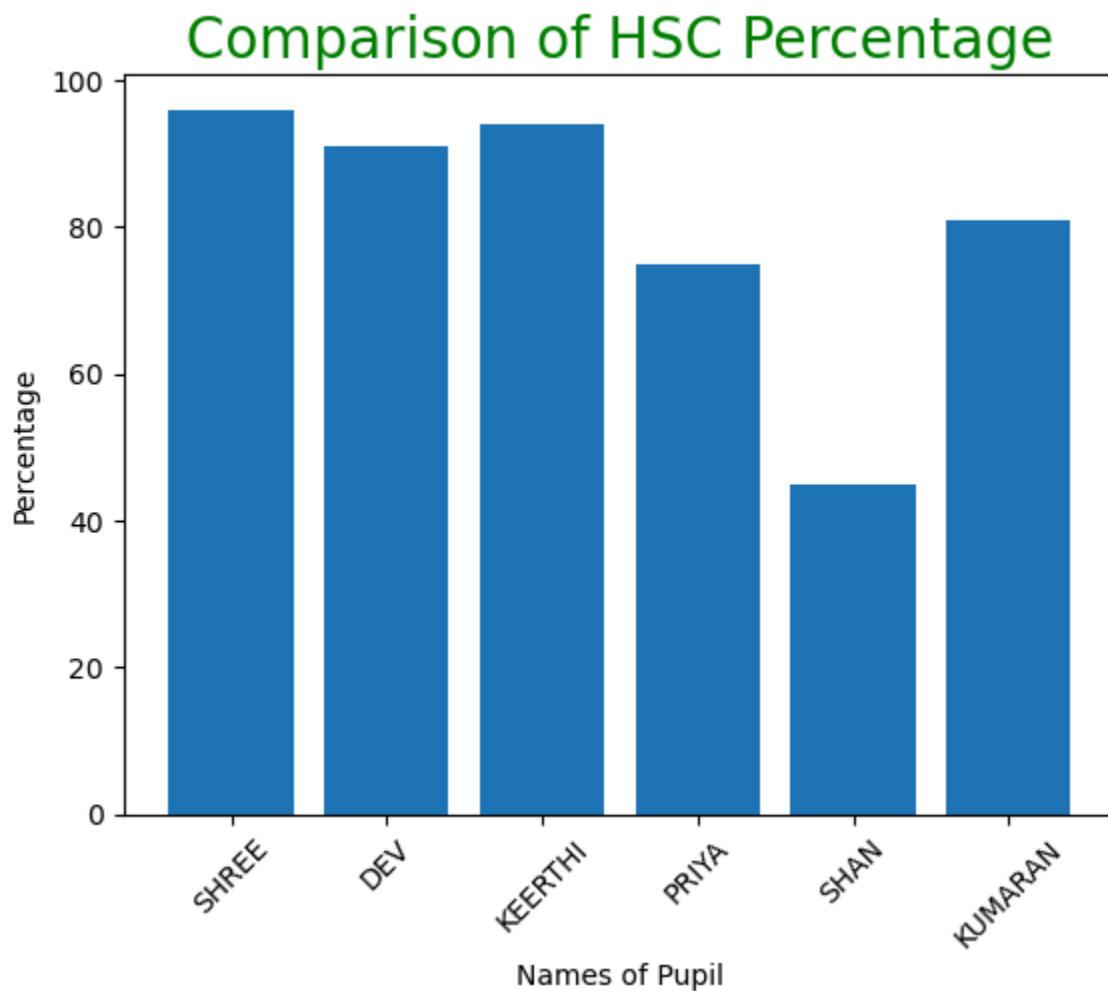
	Sales	Quantity
Sales	1.000000	0.944922
Quantity	0.944922	1.000000

```
C:\Users\REC\AppData\Local\Temp\ipykernel_15972\889183971.py:39: FutureWarning:  
The default value of numeric_only in DataFrame.corr is deprecated. In a future  
version, it will default to False. Select only valid columns or specify the val  
ue of numeric_only to silence this warning.  
correlation_matrix = df.corr()
```



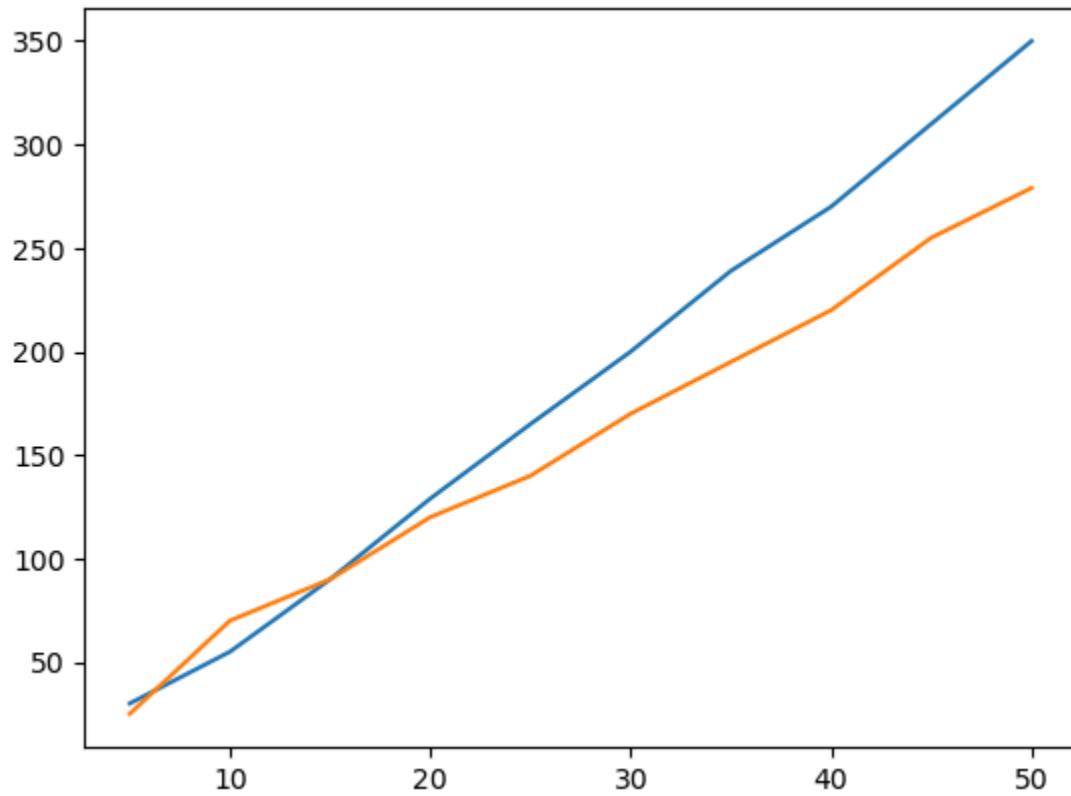
In []:

```
In [3]: import matplotlib.pyplot as hscmark
import numpy as np
Names = ['SHREE', 'DEV', 'KEERTHI', 'PRIYA', 'SHAN', 'KUMARAN']
xaxis = np.arange(len(Names))
Percentage_hsc = [96, 91, 94, 75, 45, 81]
hscmark.bar(Names, Percentage_hsc)
hscmark.xticks(xaxis, Names, rotation=45)
hscmark.xlabel("Names of Pupil")
hscmark.ylabel("Percentage")
hscmark.title("Comparison of HSC Percentage", fontsize=20, color="green")
hscmark.show()
```



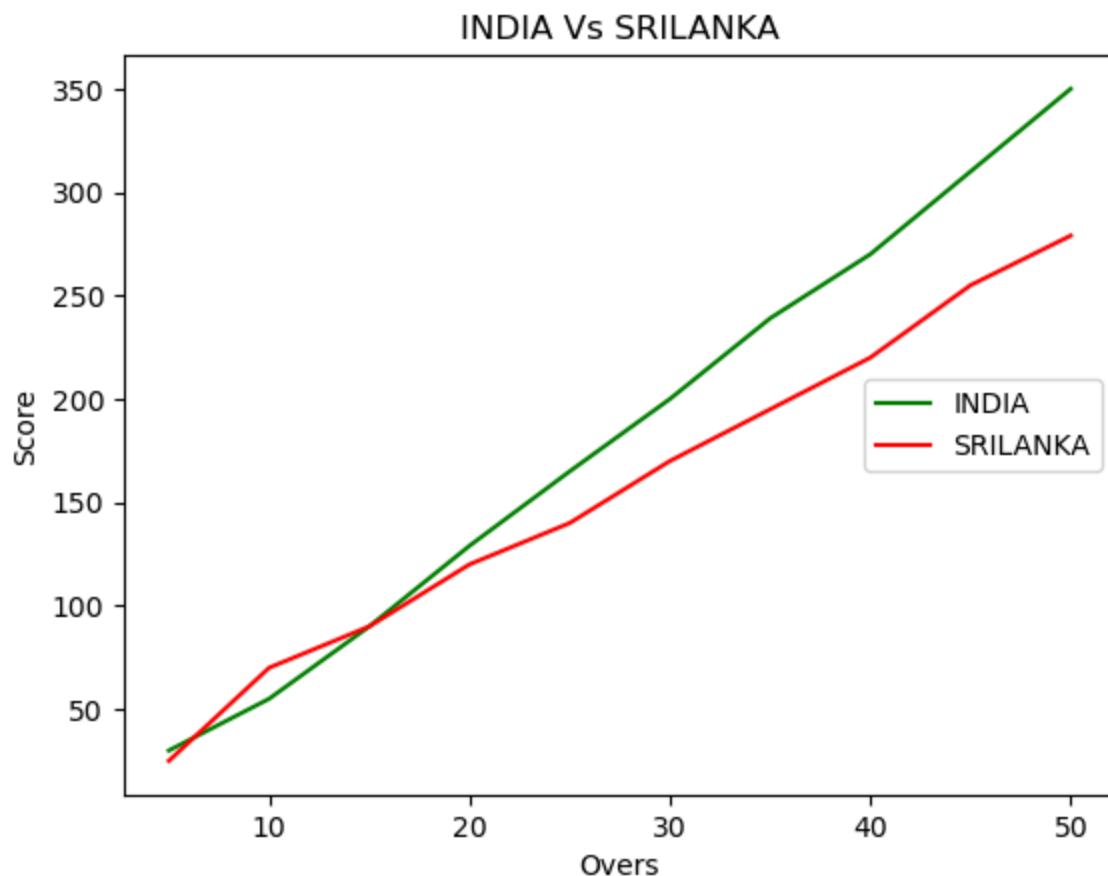
```
In [5]: import matplotlib.pyplot as cricket
Overs=list(range(5,51,5))
Indian_Score=[30,55,90,129,165,200,239,270,310,350]
Srilankan_Score=[25,70,90,120,140,170,195,220,255,279]
cricket.plot(Overs,Indian_Score)
cricket.plot(Overs,Srilankan_Score)
cricket.show()
cricket.title("INDIA Vs SRILANKA")
cricket.xlabel("Overs")
cricket.ylabel("Score")
cricket.legend()
```

```
cricket.plot(Overs,Indian_Score,color="green",label="INDIA")
cricket.plot(Overs,Srilankan_Score,color="red",label="SRILANKA")
cricket.legend(loc="center right")
```



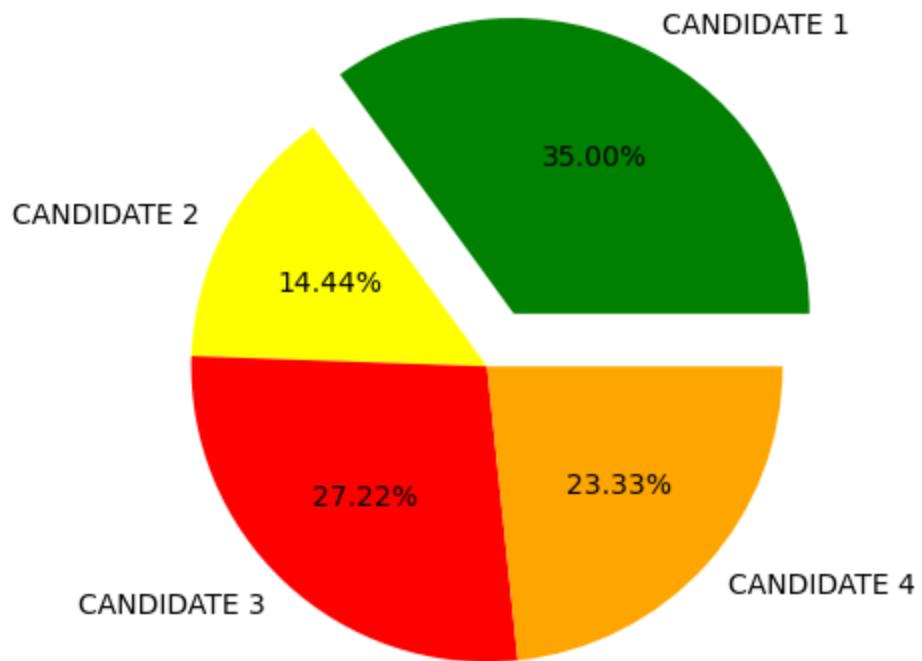
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when `legend()` is called with no argument.

Out[5]: <matplotlib.legend.Legend at 0x1f79da80e80>



```
In [6]: import matplotlib.pyplot as election
labels = ['CANDIDATE 1', 'CANDIDATE 2', 'CANDIDATE 3', 'CANDIDATE 4']
Votes = [315, 130, 245, 210]
colors = ['green', 'yellow', 'red', 'orange']
explode = (0.2, 0, 0, 0)
election.pie(Votes, labels=labels, colors=colors, explode=explode, autopct='%0.1f%%')
election.title('Election Results')
election.show()
```

Election Results



```
In [6]: import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import gutenberg
nltk.download('gutenberg')
nltk.download('punkt')
sample = gutenberg.raw("austen-emma.txt")
token = word_tokenize(sample)
wlist = []
for i in range(50):
    wlist.append(token[i])
wordfreq = [wlist.count(w) for w in wlist]
print("Pairs\n" + str(list(zip(wlist, wordfreq))))
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]      C:\Users\REC\AppData\Roaming\nltk_data...
[nltk_data]      Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]      C:\Users\REC\AppData\Roaming\nltk_data...
[nltk_data]      Package punkt is already up-to-date!
Pairs
[('[', 1), ('Emma', 2), ('by', 1), ('Jane', 1), ('Austen', 1), ('1816', 1),
(']', 1), ('VOLUME', 1), ('I', 2), ('CHAPTER', 1), ('I', 2), ('Emma', 2), ('Woo
dhouse', 1), ('', 5), ('handsome', 1), ('', 5), ('clever', 1), ('', 5), ('an
d', 3), ('rich', 1), ('', 5), ('with', 2), ('a', 1), ('comfortable', 1), ('hom
e', 1), ('and', 3), ('happy', 1), ('disposition', 1), ('', 5), ('seemed', 1),
('to', 1), ('unite', 1), ('some', 1), ('of', 2), ('the', 2), ('best', 1), ('ble
ssings', 1), ('of', 2), ('existence', 1), (';', 1), ('and', 3), ('had', 1), ('l
ived', 1), ('nearly', 1), ('twenty-one', 1), ('years', 1), ('in', 1), ('the',
2), ('world', 1), ('with', 2)]
```

```
In [ ]:
```

```
In [ ]:
```



```
In [3]: import pandas as pd
db = pd.read_csv("D:\diabetes.csv")
print(db.head())
print(db.info())
print(db.describe())
import matplotlib.pyplot as plt
import seaborn as sns
db.hist(bins=50, figsize=(20,15))
plt.show()
sns.pairplot(db)
plt.show()
```

Empty DataFrame
Columns: [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome]
Index: []
<class 'pandas.core.frame.DataFrame'>
Index: 0 entries
Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Pregnancies	0 non-null	object
1	Glucose	0 non-null	object
2	BloodPressure	0 non-null	object
3	SkinThickness	0 non-null	object
4	Insulin	0 non-null	object
5	BMI	0 non-null	object
6	DiabetesPedigreeFunction	0 non-null	object
7	Age	0 non-null	object
8	Outcome	0 non-null	object

dtypes: object(9)
memory usage: 0.0+ bytes
None

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
count	0	0	0	0	0	0	0
unique	0	0	0	0	0	0	0
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN

DiabetesPedigreeFunction Age Outcome

	DiabetesPedigreeFunction	Age	Outcome
count	0	0	0
unique	0	0	0
top	NaN	NaN	NaN
freq	NaN	NaN	NaN

```
-----  
ValueError Traceback (most recent call last)  
Cell In[3], line 8  
      6 import matplotlib.pyplot as plt  
      7 import seaborn as sns  
----> 8 db.hist(bins=50, figsize=(20,15))  
      9 plt.show()  
     10 sns.pairplot(db)  
  
File C:\ProgramData\anaconda3\lib\site-packages\pandas\plotting\_core.py:231, in hist_frame(data, column, by, grid, xlabelsize, xrot, ylabelsize, yrot, ax, sharex, sharey, figsize, layout, bins, backend, legend, **kwargs)  
140 """  
141 Make a histogram of the DataFrame's columns.  
142  
(...)  
228     >>> hist = df.hist(bins=3)  
229 """  
230 plot_backend = _get_plot_backend(backend)  
--> 231 return plot_backend.hist_frame(  
232     data,  
233     column=column,  
234     by=by,  
235     grid=grid,  
236     xlabelsize=xlabelsize,  
237     xrot=xrot,  
238     ylabelsize=ylabelsize,  
239     yrot=yrot,  
240     ax=ax,  
241     sharex=sharex,  
242     sharey=sharey,  
243     figsize=figsize,  
244     layout=layout,  
245     legend=legend,  
246     bins=bins,  
247     **kwargs,  
248 )  
  
File C:\ProgramData\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\hist.py:499, in hist_frame(data, column, by, grid, xlabelsize, xrot, ylabelsize, yrot, ax, sharex, sharey, figsize, layout, bins, legend, **kwds)  
496 naxes = len(data.columns)  
497 if naxes == 0:  
--> 499     raise ValueError(  
500         "hist method requires numerical or datetime columns, nothing to  
plot."  
501     )  
502 fig, axes = create_subplots(  
503     naxes=naxes,  
504     ax=ax,  
(...)  
510     layout=layout,  
511 )  
512 _axes = flatten_axes(axes)
```

ValueError: hist method requires numerical or datetime columns, nothing to plot.

```
In [1]: import numpy as np
import pandas as pd

df = pd.read_csv("D:\Hotel_Dataset.csv")

print(df.duplicated())

df.drop_duplicates(inplace=True)

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

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

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.NoOfPax < 1) | (df.NoOfPax > 20), 'NoOfPax'] = np.nan

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

print(df)
```

```

0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9    True
10   False
dtype: bool
   CustomerID  Age_Group  Rating(1-5)      Hotel FoodPreference    Bill \
0        1.0     20-25            4       Ibis          Veg  1300.0
1        2.0     30-35            5  LemonTree      Non-Veg  2000.0
2        3.0     25-30            6     RedFox          Veg  1322.0
3        4.0     20-25           -1  LemonTree          Veg  1234.0
4        5.0      35+             3       Ibis          Veg   989.0
5        6.0      35+             3       Ibis      Non-Veg  1909.0
6        7.0      35+             4     RedFox          Veg  1000.0
7        8.0     20-25            7  LemonTree          Veg  2999.0
8        9.0     25-30            2       Ibis      Non-Veg  3456.0
9       10.0     30-35            5     RedFox      Non-Veg  6755.0

  NoOfPax  EstimatedSalary
0        2.0      40000.0
1        3.0      59000.0
2        2.0      30000.0
3        2.0     120000.0
4        2.0      45000.0
5        2.0     122220.0
6        2.0      21122.0
7        2.0     345673.0
8        3.0      96755.0
9        4.0      87777.0

```

```
In [3]: import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
array = np.random.randint(1, 100, 16)
print("Original Array:", array)
print("Mean:", array.mean())
print("Q1 (25%):", np.percentile(array, 25))
print("Median (50%):", np.percentile(array, 50))
print("Q3 (75%):", np.percentile(array, 75))
print("Max:", np.percentile(array, 100))
def outDetection(array):
    Q1, Q3 = np.percentile(array, [25, 75])
    IQR = Q3 - Q1
    lr = Q1 - (1.5 * IQR)
    ur = Q3 + (1.5 * IQR)
    return lr, ur

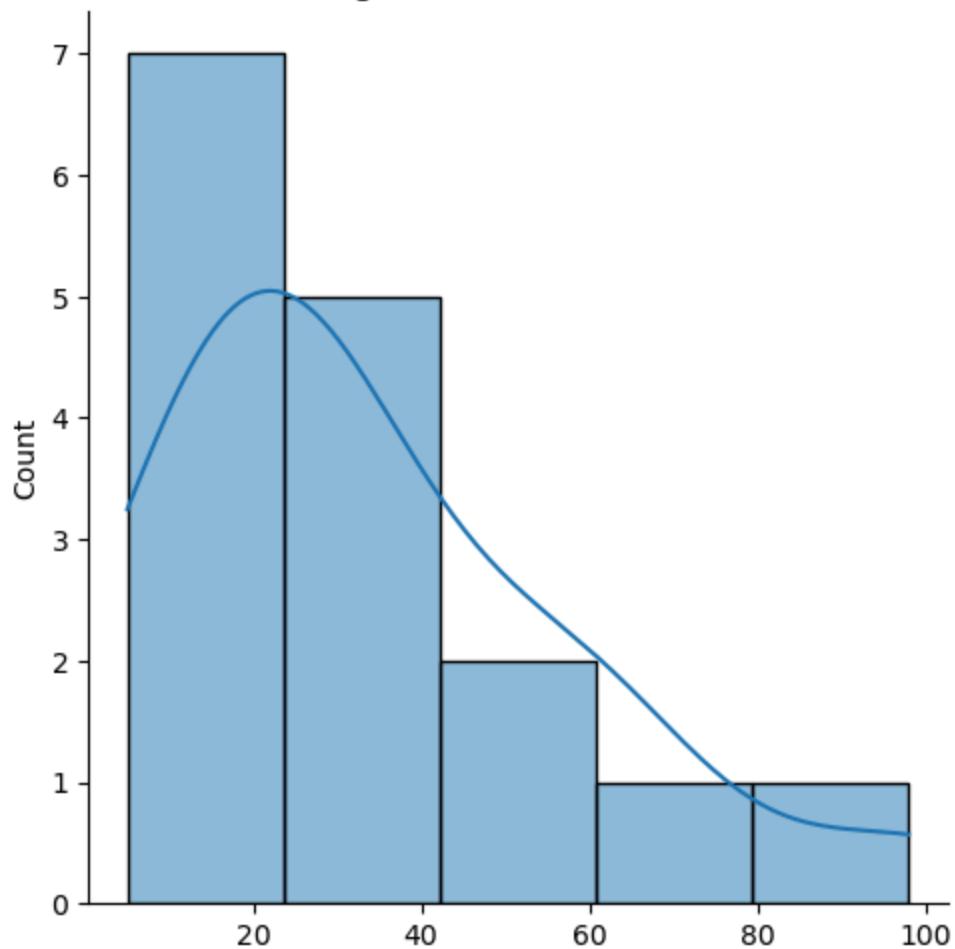
lr, ur = outDetection(array)
print("Lower Range:", lr, "Upper Range:", ur)
sns.displot(array, kde=True)
plt.title("Original Data Distribution")
plt.show()
new_array = array[(array > lr) & (array < ur)]
print("Array without outliers:", new_array)

sns.displot(new_array, kde=True)
plt.title("Data After Removing Outliers")
plt.show()
lrl, url = outDetection(new_array)
final_array = new_array[(new_array > lrl) & (new_array < url)]
print("Final Cleaned Array:", final_array)

sns.displot(final_array, kde=True)
plt.title("Final Cleaned Data Distribution")
plt.show()
```

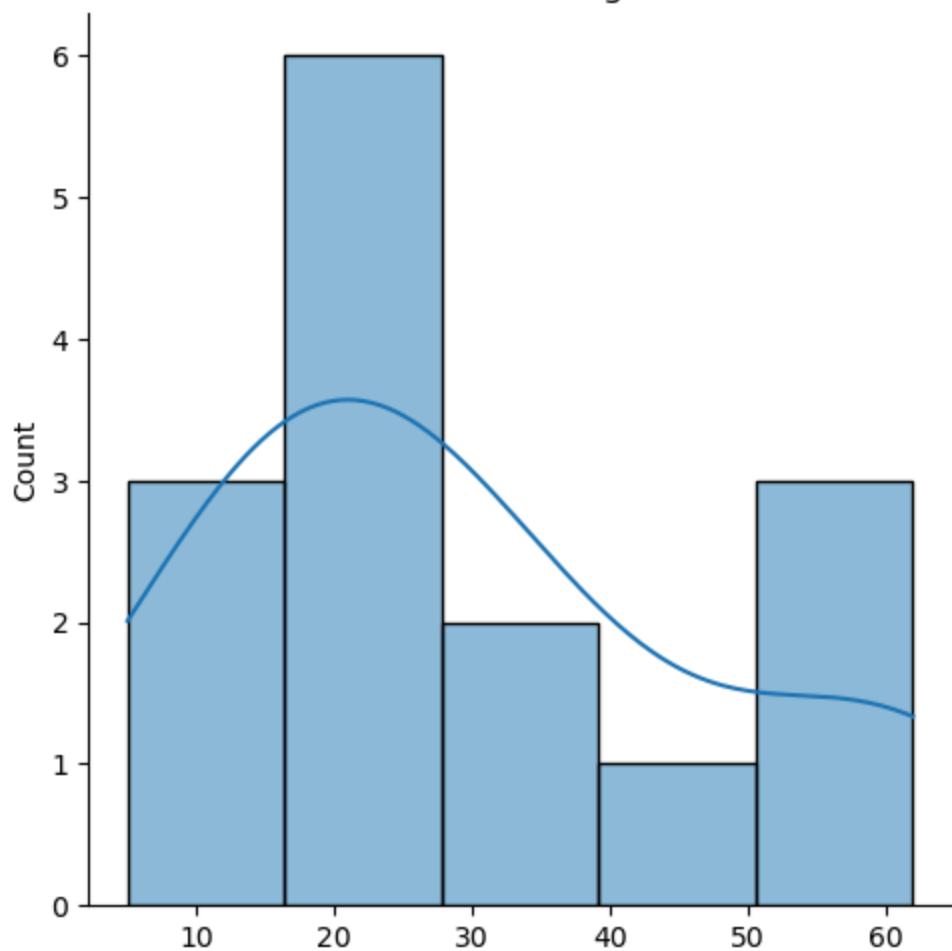
Original Array: [18 27 5 9 60 23 40 25 31 62 19 98 17 38 58 13]
Mean: 33.9375
Q1 (25%): 17.75
Median (50%): 26.0
Q3 (75%): 44.5
Max: 98.0
Lower Range: -22.375 Upper Range: 84.625

Original Data Distribution



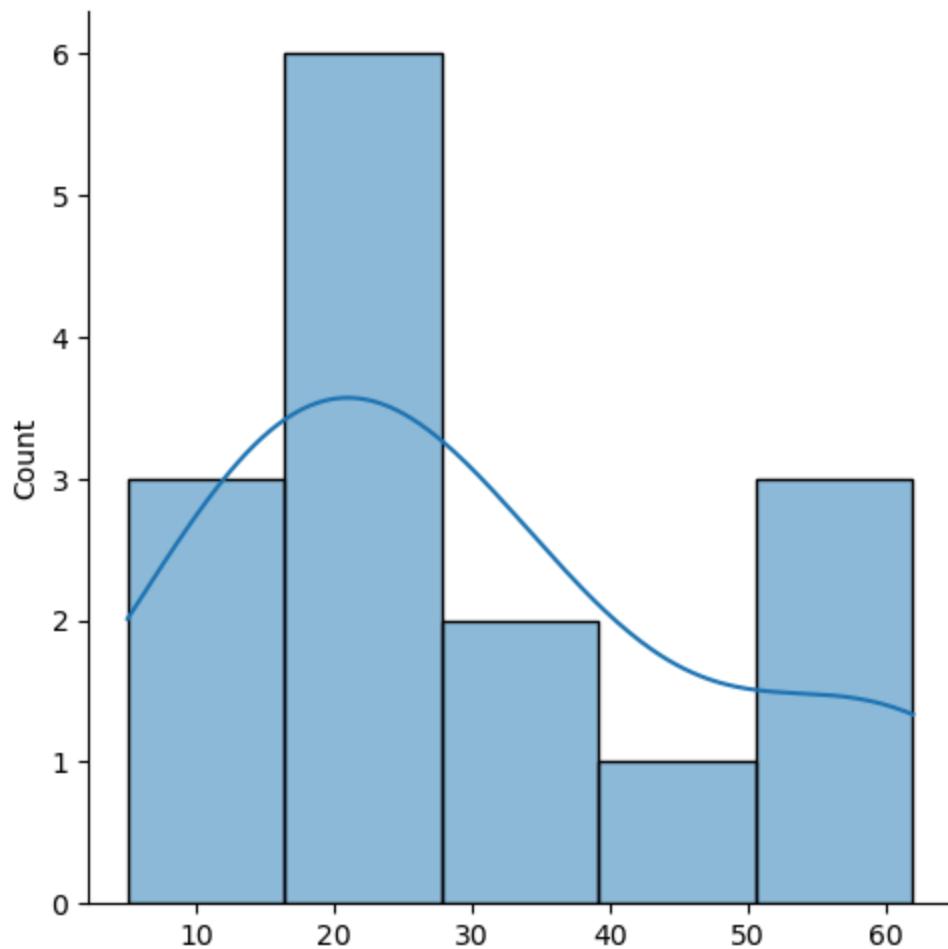
Array without outliers: [18 27 5 9 60 23 40 25 31 62 19 17 38 58 13]

Data After Removing Outliers



Final Cleaned Array: [18 27 5 9 60 23 40 25 31 62 19 17 38 58 13]

Final Cleaned Data Distribution



In []:

In []:



```
In [2]: import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, MinMaxScaler

df = pd.read_csv('E:\pre_process_datasample.csv')
print("Initial DataFrame:\n", df, "\n")

df.Country.fillna(df.Country.mode()[0], inplace=True)

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

age = SimpleImputer(strategy="mean", missing_values=np.nan)
Salary = SimpleImputer(strategy="mean", missing_values=np.nan)

age.fit(features[:, [1]])
Salary.fit(features[:, [2]])

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

print("After Missing Value Imputation:\n", features, "\n")

oh = OneHotEncoder(sparse_output=False)
Country = oh.fit_transform(features[:, [0]])

final_set = np.concatenate((Country, features[:, [1, 2]]), axis=1)
print("After One-Hot Encoding:\n", final_set, "\n")

sc = StandardScaler()
feat_standard_scaler = sc.fit_transform(final_set)
print("After Standard Scaling:\n", feat_standard_scaler, "\n")

mms = MinMaxScaler(feature_range=(0, 1))
feat_minmax_scaler = mms.fit_transform(final_set)
print("After Min-Max Scaling:\n", feat_minmax_scaler)
```

Initial DataFrame:

```
   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
```

After Missing Value Imputation:

```
[['France' 44.0 72000.0]
['Spain' 27.0 48000.0]
['Germany' 30.0 54000.0]
['Spain' 38.0 61000.0]
['Germany' 40.0 63777.7777777778]
['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]]
```

After One-Hot Encoding:

```
[[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 63777.7777777778]
[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]]
```

After Standard Scaling:

```
[[ 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]]

```

After Min-Max Scaling:

```

[[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]]

```

```

<>:6: SyntaxWarning: invalid escape sequence '\p'
<>:6: SyntaxWarning: invalid escape sequence '\p'
C:\Users\sownd\AppData\Local\Temp\ipykernel_14424\2206898782.py:6: SyntaxWarning: invalid escape sequence '\p'
  df = pd.read_csv('E:\pre_process_datasample.csv')
C:\Users\sownd\AppData\Local\Temp\ipykernel_14424\2206898782.py:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chain
ed assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.

```
df.Country.fillna(df.Country.mode()[0], inplace=True)
```

```
In [4]: #EXP 8 (B)
import numpy as np
import pandas as pd

df = pd.read_csv("E:\pre_process_datasample.csv")
print("Initial DataFrame:\n", df, "\n")

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)
print("After Filling Missing Values:\n", df, "\n")

encoded_country = pd.get_dummies(df.Country)
print("Encoded Country Columns:\n", encoded_country, "\n")

updated_dataset = pd.concat([encoded_country, df.iloc[:, [1, 2, 3]]], axis=1)
print("Combined Dataset (Before Encoding 'Purchased'):\n", updated_dataset, "\n")
```

```
updated_dataset.Purchased.replace(['No', 'Yes'], [0, 1], inplace=True)
print("Final Preprocessed Dataset:\n", updated_dataset)
```

Initial DataFrame:

	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

After Filling Missing Values:

	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

Encoded Country Columns:

	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

Combined Dataset (Before Encoding 'Purchased'):

	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	38.0	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

Final Preprocessed Dataset:

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	38.0	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

```
<>:5: SyntaxWarning: invalid escape sequence '\p'
<>:5: SyntaxWarning: invalid escape sequence '\p'
C:\Users\sownd\AppData\Local\Temp\ipykernel_21780\3069696153.py:5: SyntaxWarning: invalid escape sequence '\p'
    df = pd.read_csv("E:\pre_process_datasample.csv")
C:\Users\sownd\AppData\Local\Temp\ipykernel_21780\3069696153.py:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df.Country.fillna(df.Country.mode()[0], inplace=True)
C:\Users\sownd\AppData\Local\Temp\ipykernel_21780\3069696153.py:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df.Age.fillna(df.Age.median(), inplace=True)
C:\Users\sownd\AppData\Local\Temp\ipykernel_21780\3069696153.py:10: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df.Salary.fillna(round(df.Salary.mean()), inplace=True)
C:\Users\sownd\AppData\Local\Temp\ipykernel_21780\3069696153.py:19: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
    updated_dataset.Purchased.replace(['No', 'Yes'], [0, 1], inplace=True)
C:\Users\sownd\AppData\Local\Temp\ipykernel_21780\3069696153.py:19: FutureWarning:
Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
    updated_dataset.Purchased.replace(['No', 'Yes'], [0, 1], inplace=True)
```

In []:



```
In [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')

print(tips.head())

sns.displot(tips['total_bill'], kde=True)
sns.displot(tips['total_bill'], kde=False)

sns.jointplot(x='tip', y='total_bill', data=tips)
sns.jointplot(x='tip', y='total_bill', data=tips, kind='reg')
sns.jointplot(x='tip', y='total_bill', data=tips, kind='hex')

sns.pairplot(tips)
print(tips['time'].value_counts())

sns.pairplot(tips, hue='time')
sns.pairplot(tips, hue='day')

sns.heatmap(tips.corr(numeric_only=True), annot=True, cmap='coolwarm')

sns.boxplot(x='total_bill', data=tips)
sns.boxplot(x='tip', data=tips)

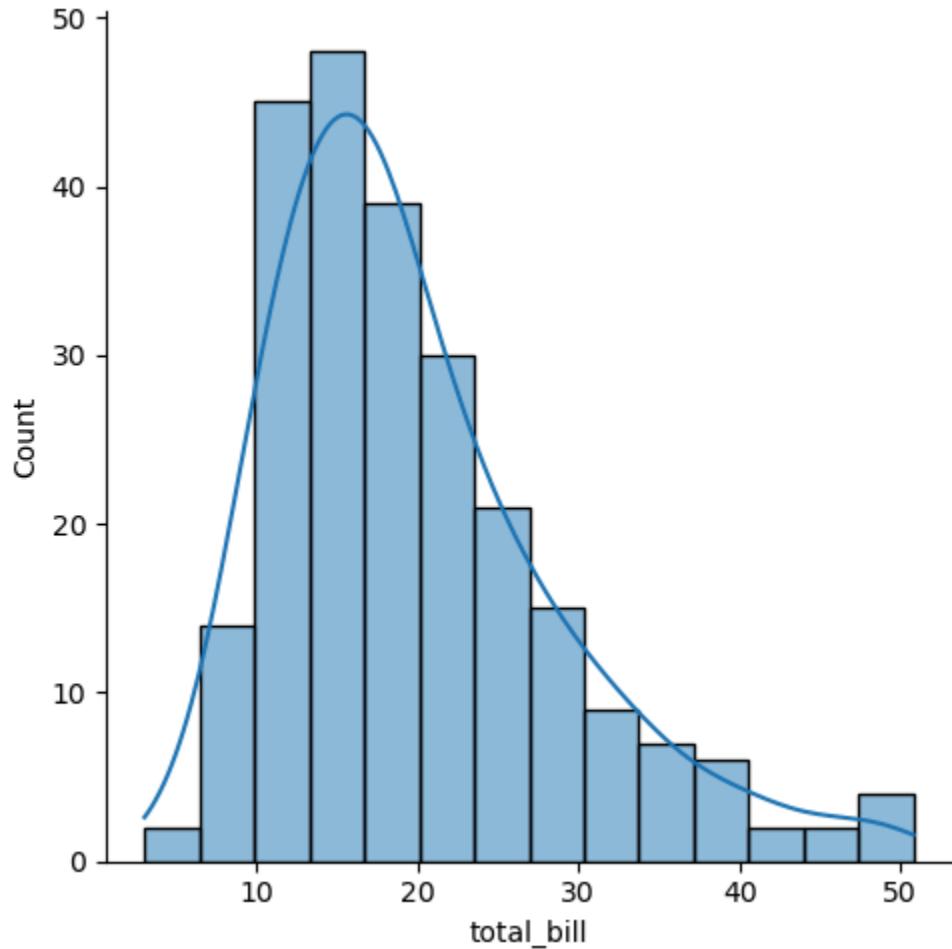
sns.countplot(x='day', data=tips)
sns.countplot(x='sex', data=tips)

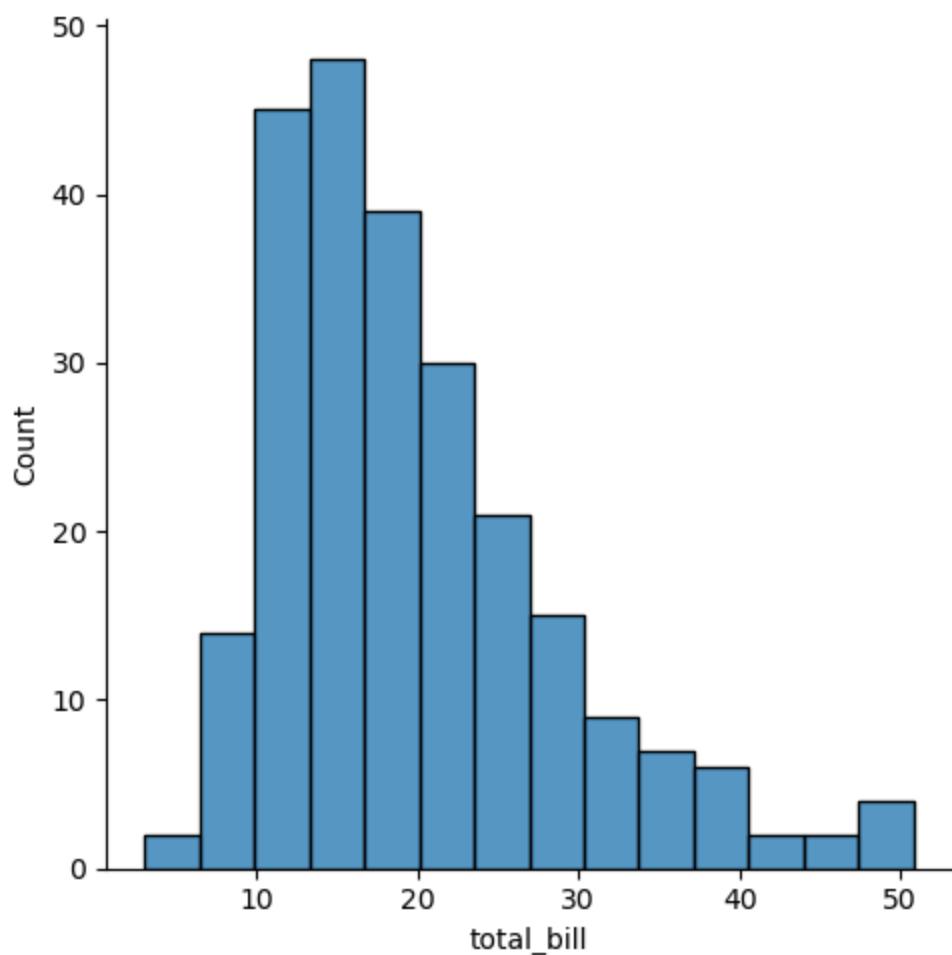
tips['sex'].value_counts().plot(kind='pie', autopct='%1.1f%%', ylabel='')
plt.title('Gender Distribution')
plt.show()

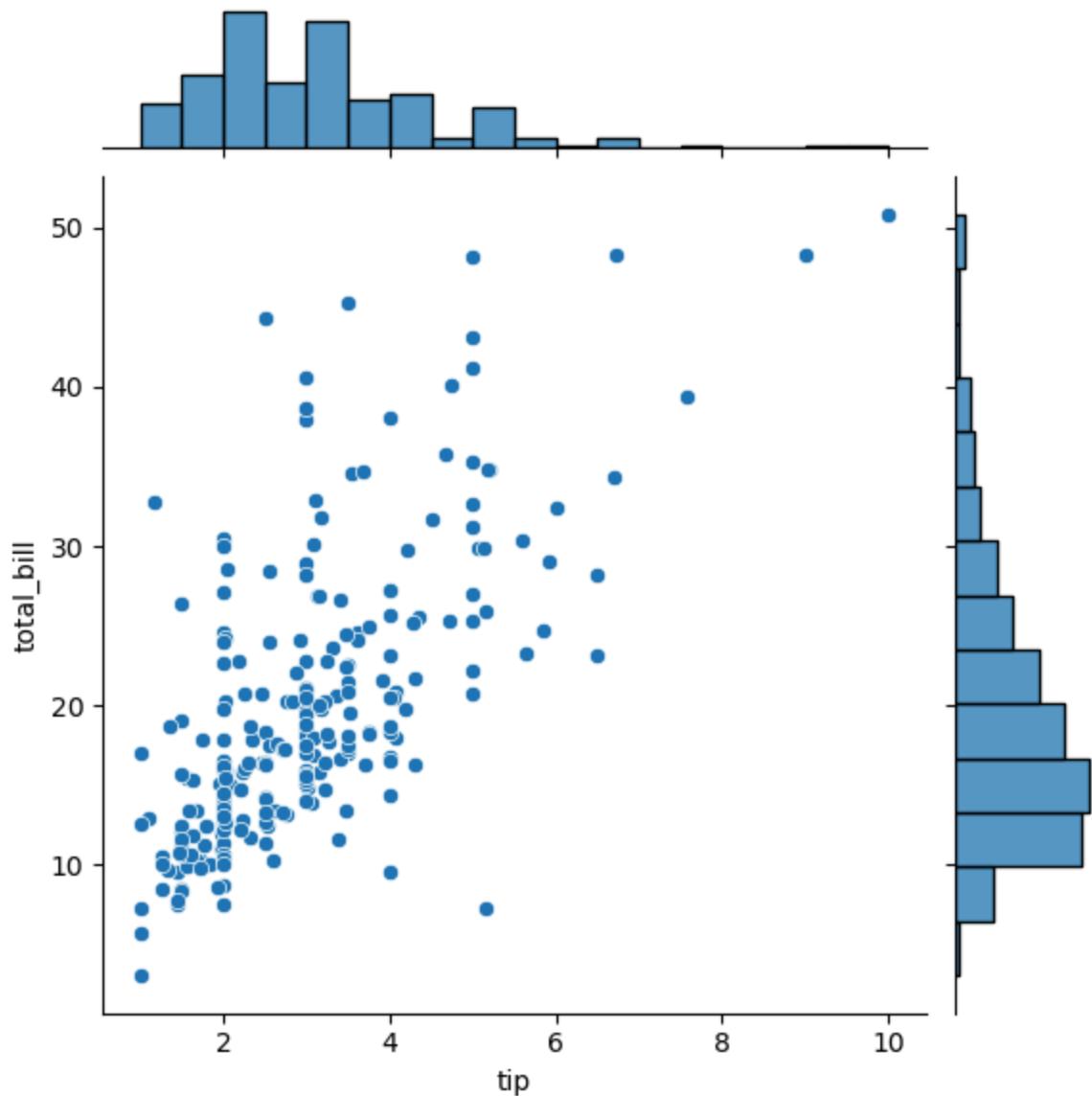
tips['sex'].value_counts().plot(kind='bar', color=['skyblue', 'salmon'])
plt.title('Gender Count')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()

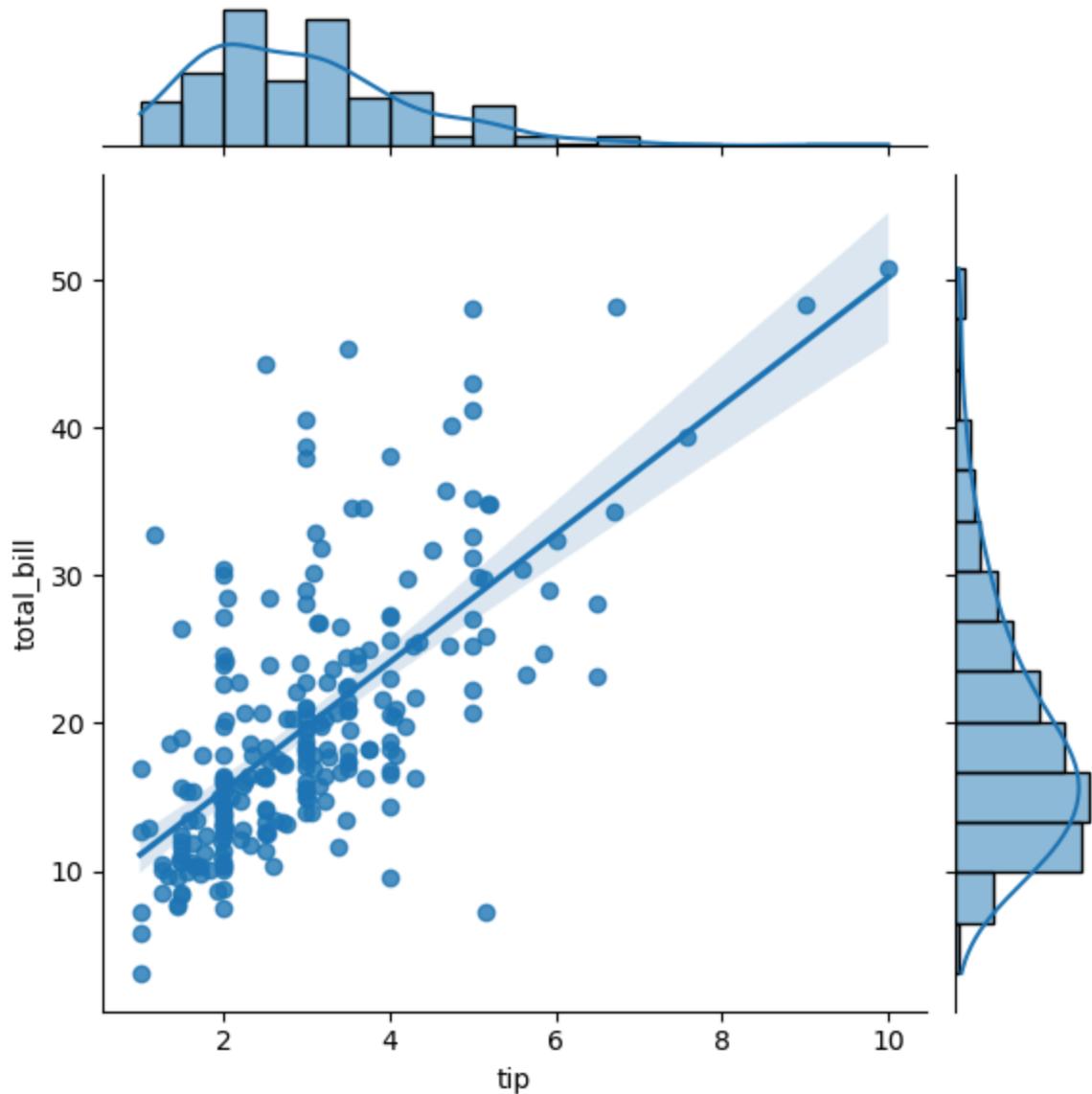
sns.countplot(x='day', data=tips[tips['time'] == 'Dinner'])
plt.title('Dinner Count by Day')
plt.show()
```

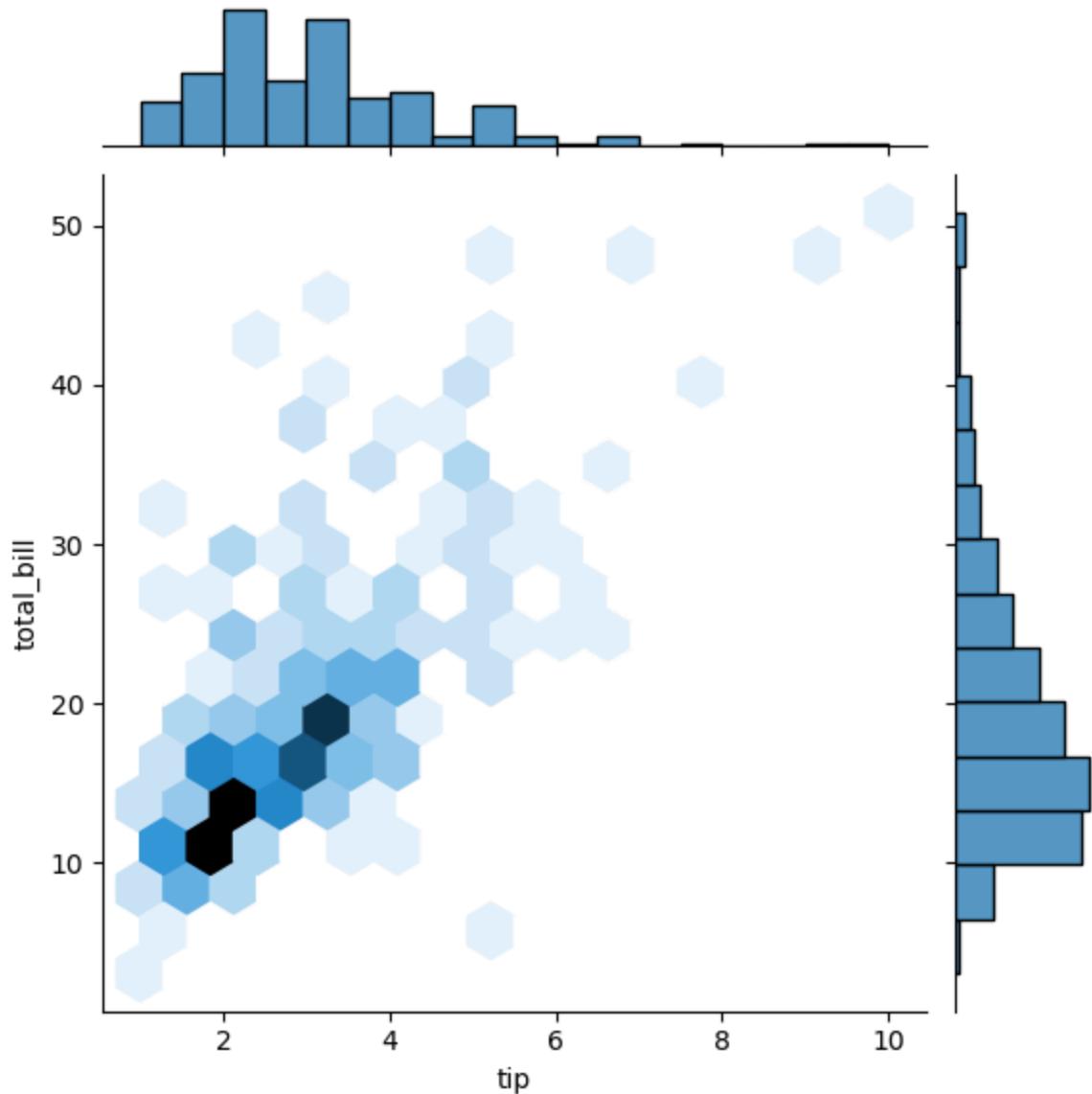
```
total_bill      tip      sex smoker  day    time    size
0        16.99  1.01  Female     No  Sun  Dinner      2
1        10.34  1.66   Male     No  Sun  Dinner      3
2        21.01  3.50   Male     No  Sun  Dinner      3
3        23.68  3.31   Male     No  Sun  Dinner      2
4        24.59  3.61 Female     No  Sun  Dinner      4
time
Dinner    176
Lunch     68
Name: count, dtype: int64
```

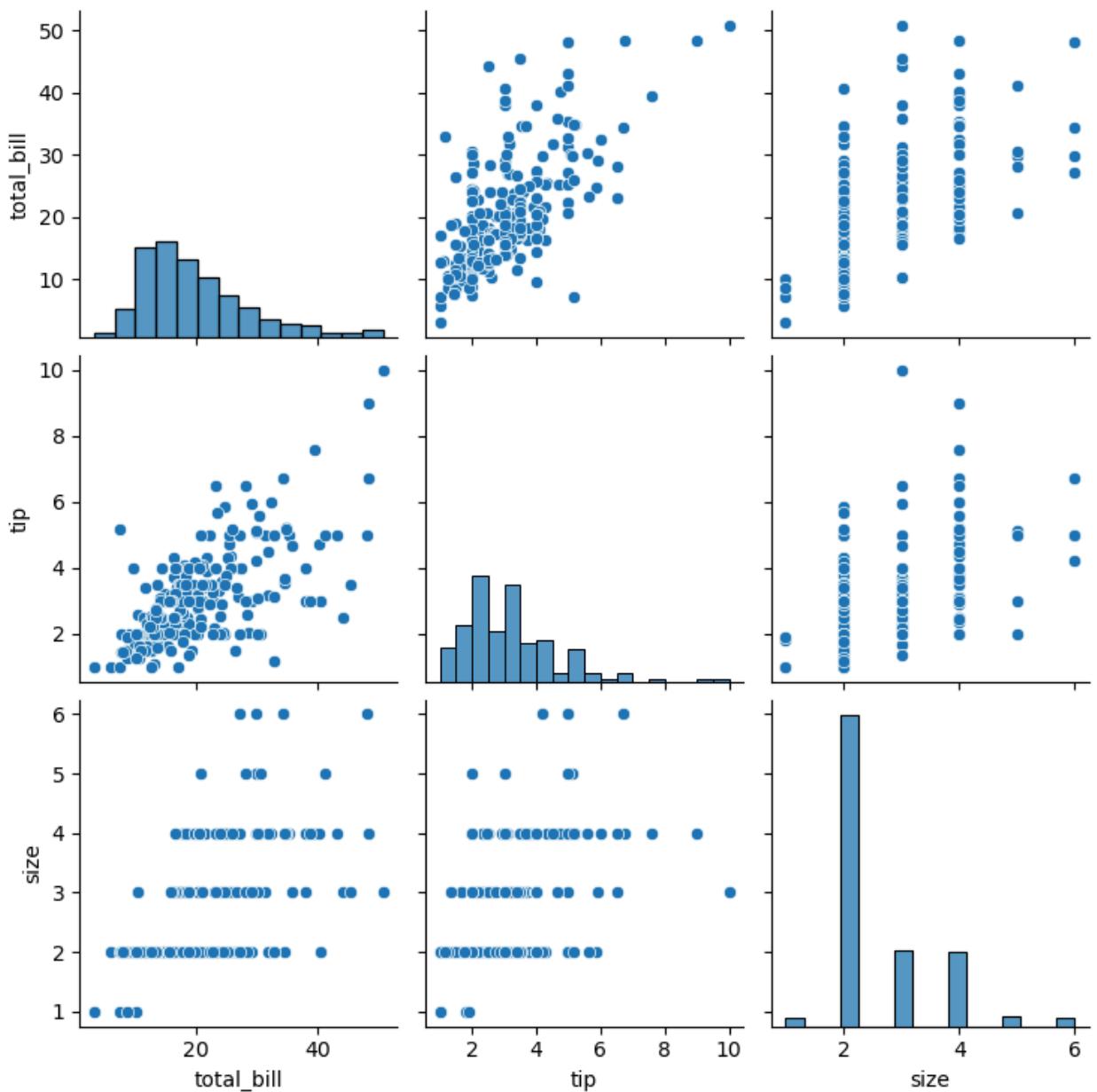


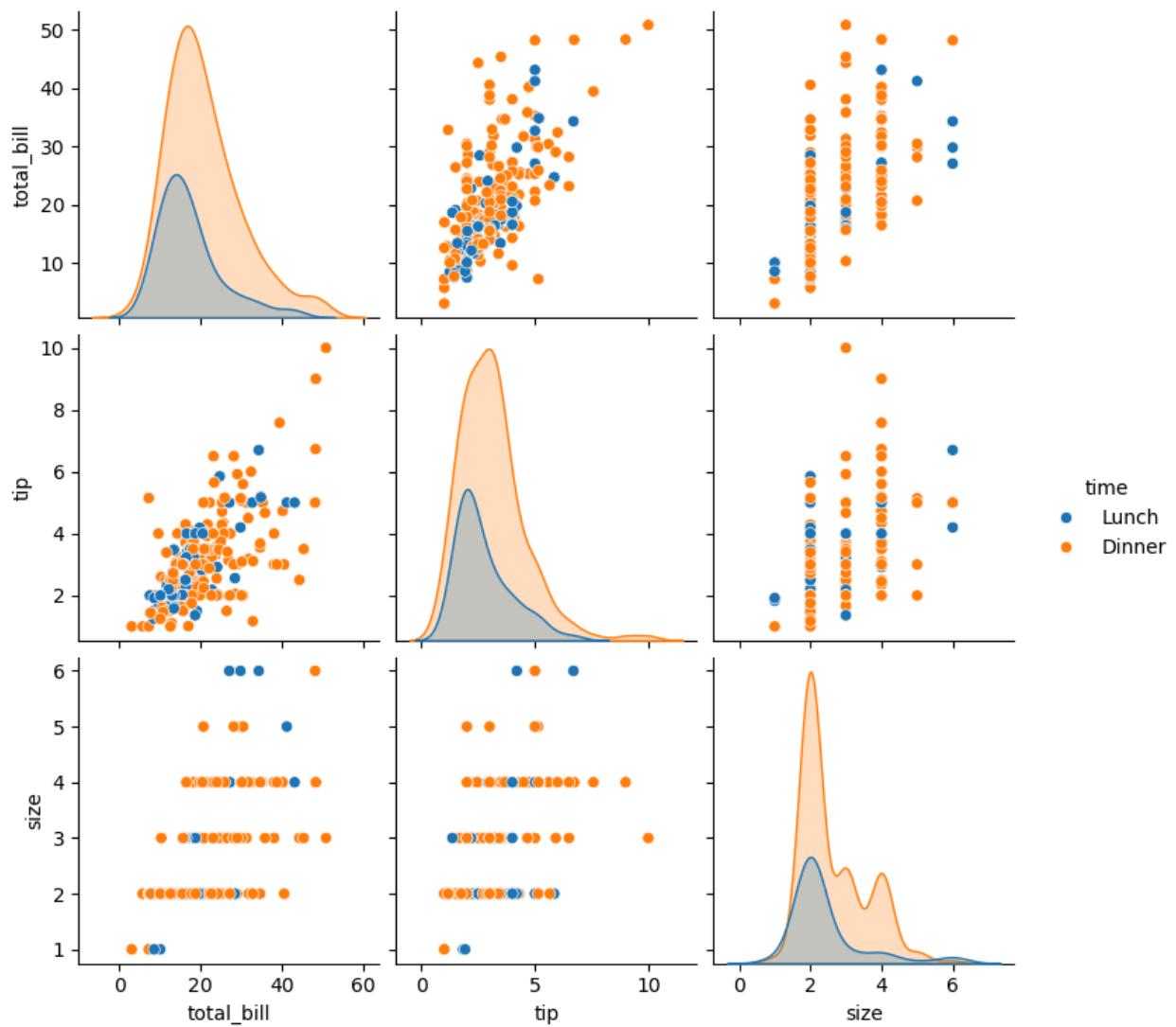


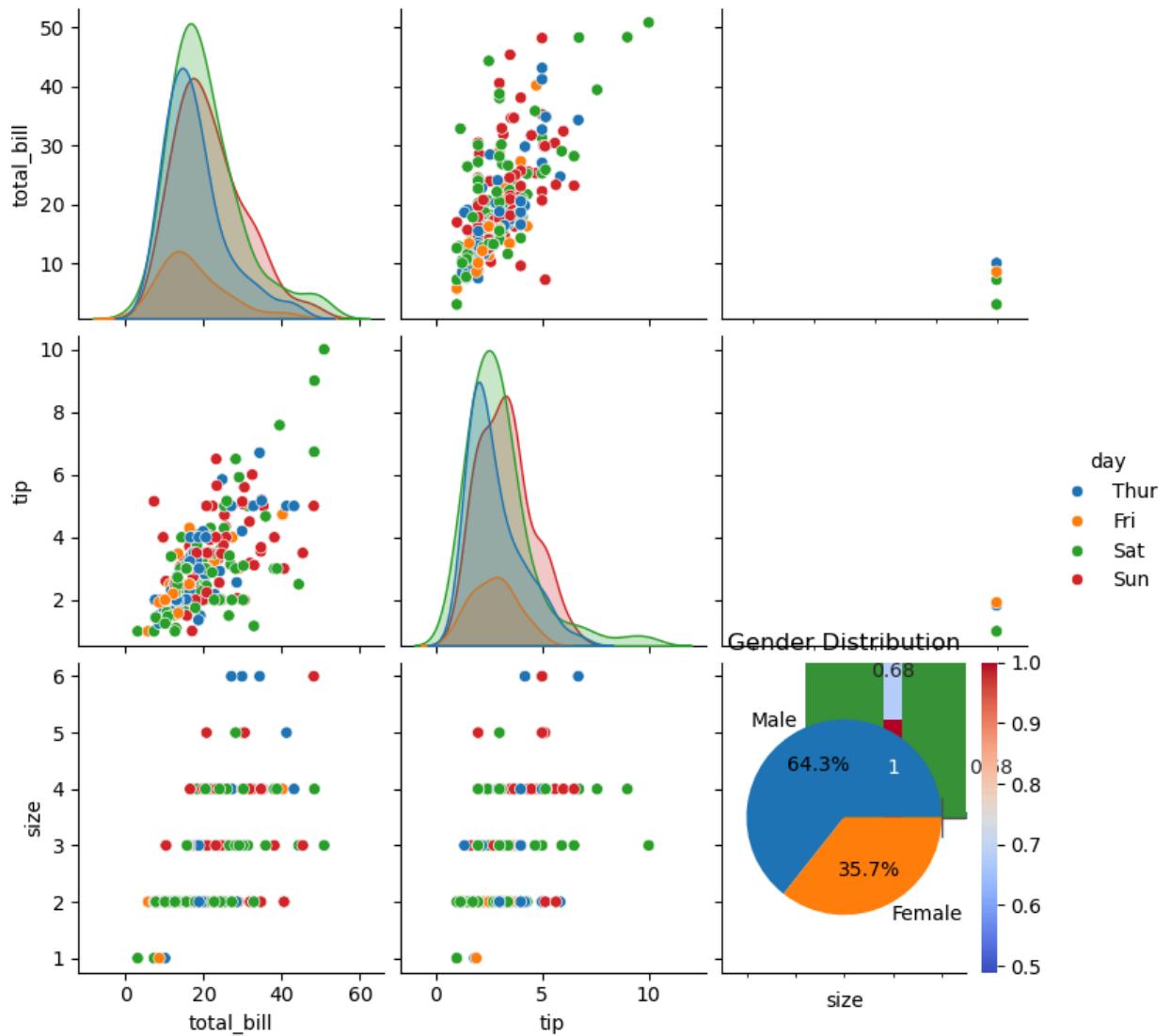


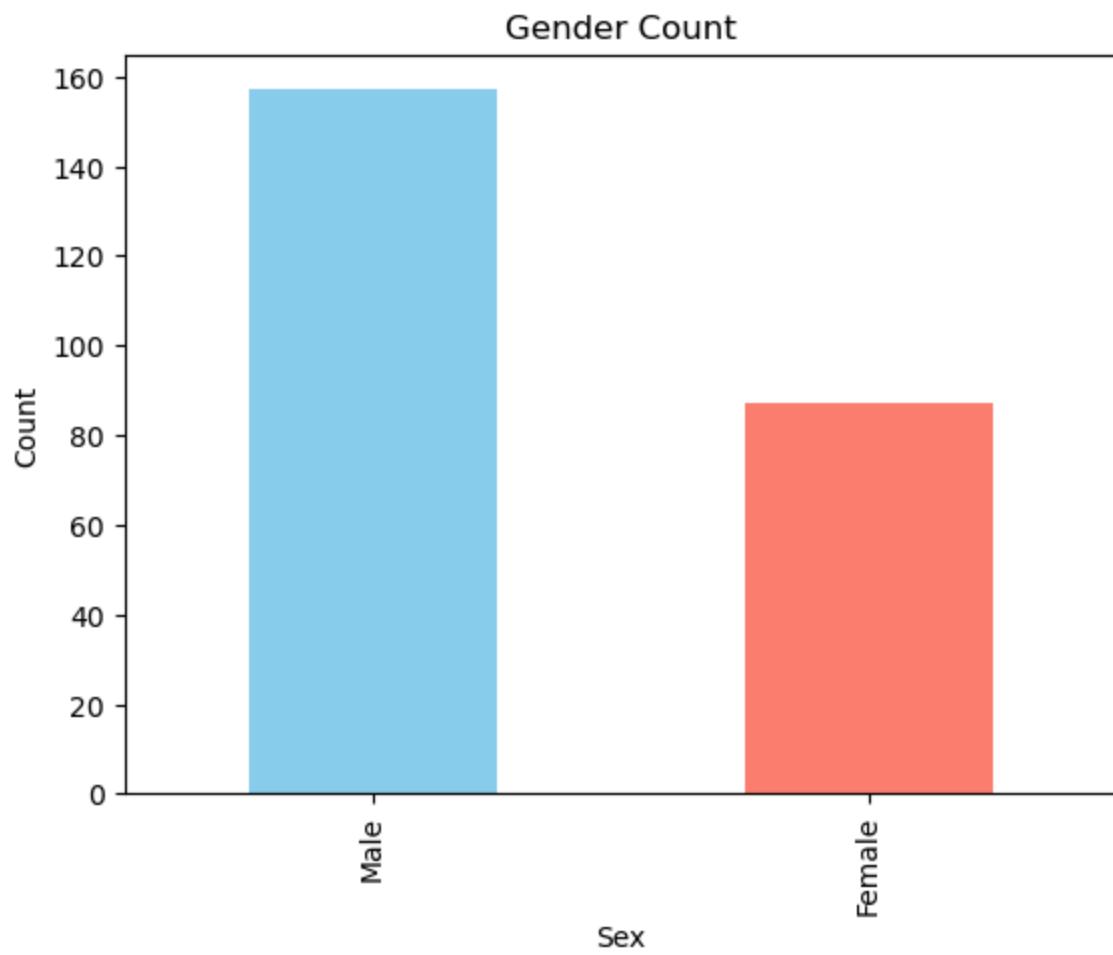


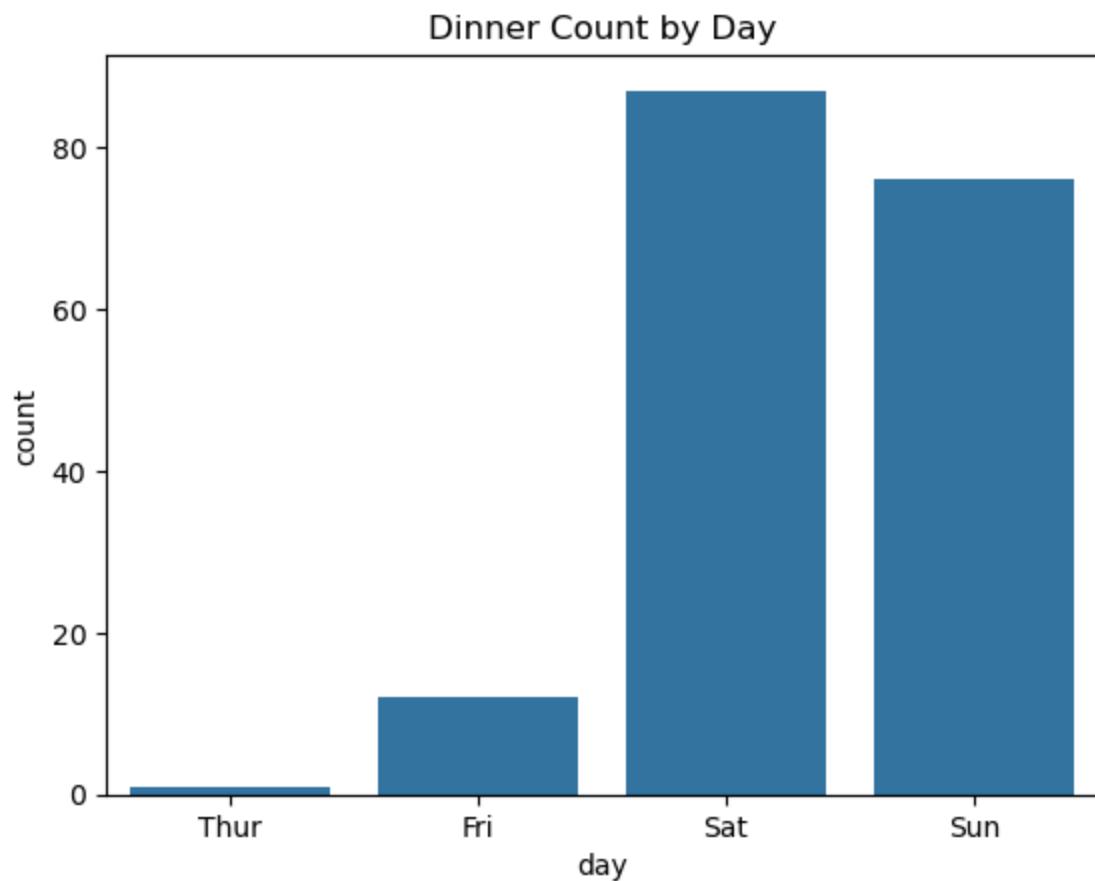












In []:



In [1]: #EXP10

```
import numpy as np
import pandas as pd

df = pd.read_csv('E:\Salary_data.csv')
print(df)

df.info()

df.dropna(inplace=True)
df.info()

df.describe()

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

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_state=0
)

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

print("Training Accuracy:", model.score(x_train, y_train))
print("Testing Accuracy:", model.score(x_test, y_test))

print("Coefficient:", model.coef_)
print("Intercept:", model.intercept_)

import pickle
pickle.dump(model, open('SalaryPred.model', 'wb'))

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)

print(f"Estimated Salary for {yr_of_exp} years of experience is {Salary}")
```

```
<ipython>:5: SyntaxWarning: invalid escape sequence '\S'
<ipython>:5: SyntaxWarning: invalid escape sequence '\S'
C:\Users\sownd\AppData\Local\Temp\ipykernel_13788\4287389640.py:5: SyntaxWarning: invalid escape sequence '\S'
  df = pd.read_csv('E:\Salary_data.csv')
```

```
    YearsExperience  Salary
0           1.1  39343
1           1.3  46205
2           1.5  37731
3           2.0  43525
4           2.2  39891
5           2.9  56642
6           3.0  60150
7           3.2  54445
8           3.2  64445
9           3.7  57189
10          3.9  63218
11          4.0  55794
12          4.0  56957
13          4.1  57081
14          4.5  61111
15          4.9  67938
16          5.1  66029
17          5.3  83088
18          5.9  81363
19          6.0  93940
20          6.8  91738
21          7.1  98273
22          7.9  101302
23          8.2  113812
24          8.7  109431
25          9.0  105582
26          9.5  116969
27          9.6  112635
28         10.3  122391
29         10.5  121872
<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
<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
Training Accuracy: 0.9411949620562126
Testing Accuracy: 0.988169515729126
Coefficient: [[9312.57512673]]
Intercept: [26780.09915063]
Estimated Salary for 8.0 years of experience is [[101280.70016446]]
```

In []:



```
In [1]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

df = pd.read_csv('E:\Social_Network_Ads.csv')
print(df.head())
print("\n")

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

for i in range(1, 401):
    x_train, x_test, y_train, y_test = train_test_split(
        features, label, test_size=0.2, random_state=i
    )
    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,
                                                       #print("\n")
print("\n")

x_train, x_test, y_train, y_test = train_test_split(
    features, label, test_size=0.2, random_state=26
)
finalModel = LogisticRegression()
finalModel.fit(x_train, y_train)

print(finalModel.score(x_train, y_train))
print("\n")
print(finalModel.score(x_test, y_test))
print("\n")
print(classification_report(label, finalModel.predict(features)))
```

```
<>:7: SyntaxWarning: invalid escape sequence '\S'
<>:7: SyntaxWarning: invalid escape sequence '\S'
C:\Users\sownd\AppData\Local\Temp\ipykernel_14224\2726418315.py:7: SyntaxWarning: invalid escape sequence '\S'
  df = pd.read_csv('E:\Social_Network_Ads.csv')
```

	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

Test 0.9 Train0.840625 Random State 4
 Test 0.8625 Train0.85 Random State 5
 Test 0.8625 Train0.859375 Random State 6
 Test 0.8875 Train0.8375 Random State 7
 Test 0.8625 Train0.8375 Random State 9
 Test 0.9 Train0.840625 Random State 10
 Test 0.8625 Train0.85625 Random State 14
 Test 0.85 Train0.84375 Random State 15
 Test 0.8625 Train0.85625 Random State 16
 Test 0.875 Train0.834375 Random State 18
 Test 0.85 Train0.84375 Random State 19
 Test 0.875 Train0.84375 Random State 20
 Test 0.8625 Train0.834375 Random State 21
 Test 0.875 Train0.840625 Random State 22
 Test 0.875 Train0.840625 Random State 24
 Test 0.85 Train0.834375 Random State 26
 Test 0.85 Train0.840625 Random State 27
 Test 0.8625 Train0.834375 Random State 30
 Test 0.8625 Train0.85625 Random State 31
 Test 0.875 Train0.853125 Random State 32
 Test 0.8625 Train0.84375 Random State 33
 Test 0.875 Train0.83125 Random State 35
 Test 0.8625 Train0.853125 Random State 36
 Test 0.8875 Train0.840625 Random State 38
 Test 0.875 Train0.8375 Random State 39
 Test 0.8875 Train0.8375 Random State 42
 Test 0.875 Train0.846875 Random State 46
 Test 0.9125 Train0.83125 Random State 47
 Test 0.875 Train0.83125 Random State 51
 Test 0.9 Train0.84375 Random State 54
 Test 0.85 Train0.84375 Random State 57
 Test 0.875 Train0.84375 Random State 58
 Test 0.925 Train0.8375 Random State 61
 Test 0.8875 Train0.834375 Random State 65
 Test 0.8875 Train0.840625 Random State 68
 Test 0.9 Train0.83125 Random State 72
 Test 0.8875 Train0.8375 Random State 75
 Test 0.925 Train0.825 Random State 76
 Test 0.8625 Train0.840625 Random State 77
 Test 0.8625 Train0.859375 Random State 81
 Test 0.875 Train0.8375 Random State 82
 Test 0.8875 Train0.8375 Random State 83
 Test 0.8625 Train0.853125 Random State 84
 Test 0.8625 Train0.840625 Random State 85
 Test 0.8625 Train0.840625 Random State 87
 Test 0.875 Train0.846875 Random State 88

Test 0.9125 Train0.8375 Random State 90
Test 0.8625 Train0.85 Random State 95
Test 0.875 Train0.85 Random State 99
Test 0.85 Train0.840625 Random State 101
Test 0.85 Train0.840625 Random State 102
Test 0.9 Train0.825 Random State 106
Test 0.8625 Train0.840625 Random State 107
Test 0.85 Train0.834375 Random State 109
Test 0.85 Train0.840625 Random State 111
Test 0.9125 Train0.840625 Random State 112
Test 0.8625 Train0.85 Random State 115
Test 0.8625 Train0.840625 Random State 116
Test 0.875 Train0.834375 Random State 119
Test 0.9125 Train0.828125 Random State 120
Test 0.8625 Train0.859375 Random State 125
Test 0.85 Train0.846875 Random State 128
Test 0.875 Train0.85 Random State 130
Test 0.9 Train0.84375 Random State 133
Test 0.925 Train0.834375 Random State 134
Test 0.8625 Train0.85 Random State 135
Test 0.875 Train0.83125 Random State 138
Test 0.8625 Train0.85 Random State 141
Test 0.85 Train0.846875 Random State 143
Test 0.85 Train0.846875 Random State 146
Test 0.85 Train0.84375 Random State 147
Test 0.8625 Train0.85 Random State 148
Test 0.875 Train0.8375 Random State 150
Test 0.8875 Train0.83125 Random State 151
Test 0.925 Train0.84375 Random State 152
Test 0.85 Train0.840625 Random State 153
Test 0.9 Train0.84375 Random State 154
Test 0.9 Train0.840625 Random State 155
Test 0.8875 Train0.846875 Random State 156
Test 0.8875 Train0.834375 Random State 158
Test 0.875 Train0.828125 Random State 159
Test 0.9 Train0.83125 Random State 161
Test 0.85 Train0.8375 Random State 163
Test 0.875 Train0.83125 Random State 164
Test 0.8625 Train0.85 Random State 169
Test 0.875 Train0.840625 Random State 171
Test 0.85 Train0.840625 Random State 172
Test 0.9 Train0.825 Random State 180
Test 0.85 Train0.834375 Random State 184
Test 0.925 Train0.821875 Random State 186
Test 0.9 Train0.83125 Random State 193
Test 0.8625 Train0.85 Random State 195
Test 0.8625 Train0.840625 Random State 196
Test 0.8625 Train0.8375 Random State 197
Test 0.875 Train0.840625 Random State 198
Test 0.8875 Train0.8375 Random State 199
Test 0.8875 Train0.84375 Random State 200
Test 0.8625 Train0.8375 Random State 202
Test 0.8625 Train0.840625 Random State 203
Test 0.8875 Train0.83125 Random State 206

Test 0.8625 Train0.834375 Random State 211
Test 0.85 Train0.84375 Random State 212
Test 0.8625 Train0.834375 Random State 214
Test 0.875 Train0.83125 Random State 217
Test 0.9625 Train0.81875 Random State 220
Test 0.875 Train0.84375 Random State 221
Test 0.85 Train0.840625 Random State 222
Test 0.9 Train0.84375 Random State 223
Test 0.8625 Train0.853125 Random State 227
Test 0.8625 Train0.834375 Random State 228
Test 0.9 Train0.840625 Random State 229
Test 0.85 Train0.84375 Random State 232
Test 0.875 Train0.846875 Random State 233
Test 0.9125 Train0.840625 Random State 234
Test 0.8625 Train0.840625 Random State 235
Test 0.85 Train0.846875 Random State 236
Test 0.875 Train0.846875 Random State 239
Test 0.85 Train0.84375 Random State 241
Test 0.8875 Train0.85 Random State 242
Test 0.8875 Train0.825 Random State 243
Test 0.875 Train0.846875 Random State 244
Test 0.875 Train0.840625 Random State 245
Test 0.875 Train0.846875 Random State 246
Test 0.8625 Train0.859375 Random State 247
Test 0.8875 Train0.84375 Random State 248
Test 0.8625 Train0.85 Random State 250
Test 0.875 Train0.83125 Random State 251
Test 0.8875 Train0.84375 Random State 252
Test 0.8625 Train0.846875 Random State 255
Test 0.9 Train0.840625 Random State 257
Test 0.8625 Train0.85625 Random State 260
Test 0.8625 Train0.840625 Random State 266
Test 0.8625 Train0.8375 Random State 268
Test 0.875 Train0.840625 Random State 275
Test 0.8625 Train0.85 Random State 276
Test 0.925 Train0.8375 Random State 277
Test 0.875 Train0.846875 Random State 282
Test 0.85 Train0.846875 Random State 283
Test 0.85 Train0.84375 Random State 285
Test 0.9125 Train0.834375 Random State 286
Test 0.85 Train0.840625 Random State 290
Test 0.85 Train0.840625 Random State 291
Test 0.85 Train0.846875 Random State 292
Test 0.8625 Train0.8375 Random State 294
Test 0.8875 Train0.828125 Random State 297
Test 0.8625 Train0.834375 Random State 300
Test 0.8625 Train0.85 Random State 301
Test 0.8875 Train0.85 Random State 302
Test 0.875 Train0.846875 Random State 303
Test 0.8625 Train0.834375 Random State 305
Test 0.9125 Train0.8375 Random State 306
Test 0.875 Train0.846875 Random State 308
Test 0.9 Train0.84375 Random State 311
Test 0.8625 Train0.834375 Random State 313

Test 0.9125 Train0.834375 Random State 314
Test 0.875 Train0.8375 Random State 315
Test 0.9 Train0.846875 Random State 317
Test 0.9125 Train0.821875 Random State 319
Test 0.8625 Train0.85 Random State 321
Test 0.9125 Train0.828125 Random State 322
Test 0.85 Train0.846875 Random State 328
Test 0.85 Train0.8375 Random State 332
Test 0.8875 Train0.853125 Random State 336
Test 0.85 Train0.8375 Random State 337
Test 0.875 Train0.840625 Random State 343
Test 0.8625 Train0.84375 Random State 346
Test 0.8875 Train0.83125 Random State 351
Test 0.8625 Train0.85 Random State 352
Test 0.95 Train0.81875 Random State 354
Test 0.8625 Train0.85 Random State 356
Test 0.9125 Train0.840625 Random State 357
Test 0.8625 Train0.8375 Random State 358
Test 0.85 Train0.840625 Random State 362
Test 0.9 Train0.84375 Random State 363
Test 0.8625 Train0.853125 Random State 364
Test 0.9375 Train0.821875 Random State 366
Test 0.9125 Train0.840625 Random State 369
Test 0.8625 Train0.853125 Random State 371
Test 0.925 Train0.834375 Random State 376
Test 0.9125 Train0.828125 Random State 377
Test 0.8875 Train0.85 Random State 378
Test 0.8875 Train0.85 Random State 379
Test 0.8625 Train0.840625 Random State 382
Test 0.8625 Train0.859375 Random State 386
Test 0.85 Train0.8375 Random State 387
Test 0.875 Train0.828125 Random State 388
Test 0.85 Train0.84375 Random State 394
Test 0.8625 Train0.8375 Random State 395
Test 0.9 Train0.84375 Random State 397
Test 0.8625 Train0.84375 Random State 400

0.834375

0.85

	precision	recall	f1-score	support
0	0.85	0.91	0.88	257
1	0.81	0.71	0.76	143
accuracy			0.84	400
macro avg	0.83	0.81	0.82	400
weighted avg	0.84	0.84	0.83	400

In []:

```
In [1]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report

df = pd.read_csv('E:\Iris.csv')
df.info()
df.variety.value_counts()
df.head()

features = df.iloc[:, :-1].values
label = df.iloc[:, 4].values

xtrain, xtest, ytrain, ytest = train_test_split(
    features, label, test_size=0.2, random_state=42
)

model_KNN = KNeighborsClassifier(n_neighbors=5)
model_KNN.fit(xtrain, ytrain)

print(model_KNN.score(xtrain, ytrain))
print("\n")
print(model_KNN.score(xtest, ytest))
print("\n")
print(confusion_matrix(label, model_KNN.predict(features)))
print("\n")
print(classification_report(label, model_KNN.predict(features)))
```

```
<ipython>:7: SyntaxWarning: invalid escape sequence '\I'
<ipython>:7: SyntaxWarning: invalid escape sequence '\I'
C:\Users\sownd\AppData\Local\Temp\ipykernel_2540\140597085.py:7: SyntaxWarning:
invalid escape sequence '\I'
  df = pd.read_csv('E:\Iris.csv')
```

```
<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
0.9666666666666667
```

1.0

```
[[50  0  0]
 [ 0 47  3]
 [ 0  1 49]]
```

	precision	recall	f1-score	support
Setosa	1.00	1.00	1.00	50
Versicolor	0.98	0.94	0.96	50
Virginica	0.94	0.98	0.96	50
accuracy			0.97	150
macro avg	0.97	0.97	0.97	150
weighted avg	0.97	0.97	0.97	150

In []:



```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

df = pd.read_csv('E:\Mall_Customers.csv')
df.info()
df.head()

sns.pairplot(df)

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

model = KMeans(n_clusters=5)
model.fit(features)

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

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()

features_el = df.iloc[:, [2, 3, 4]].values
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)
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.title("Elbow Method")
plt.show()
```

```
<ipython>:7: SyntaxWarning: invalid escape sequence '\M'
<ipython>:7: SyntaxWarning: invalid escape sequence '\M'
C:\Users\sownd\AppData\Local\Temp\ipykernel_17980\1677738136.py:7: SyntaxWarning: invalid escape sequence '\M'
  df = pd.read_csv('E:\Mall_Customers.csv')
```

```
<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
```

```
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: 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(
```

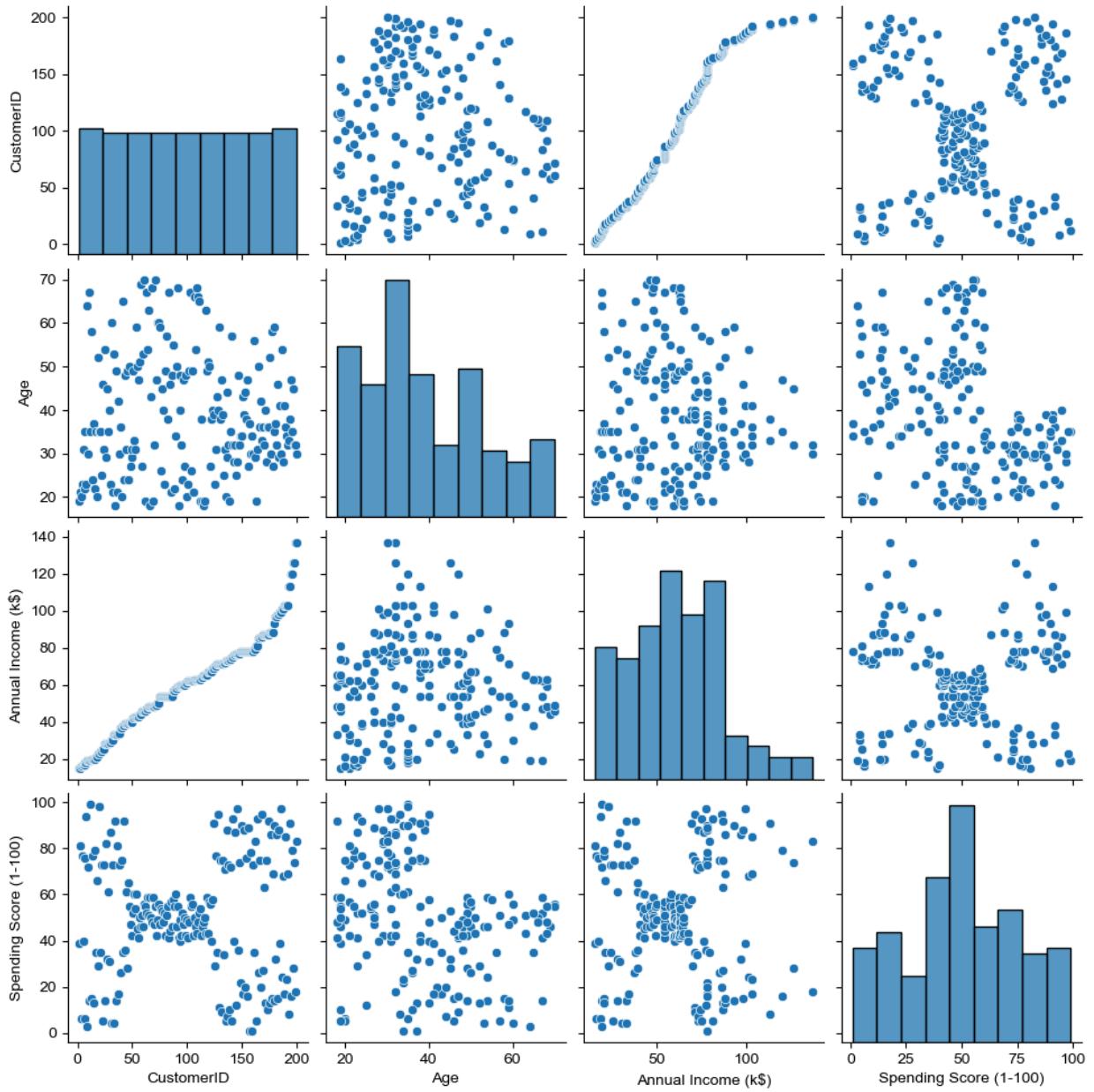
```
C:\Users\sownd\AppData\Local\Temp\ipykernel_17980\1677738136.py:19: SettingWithCopyWarning:
```

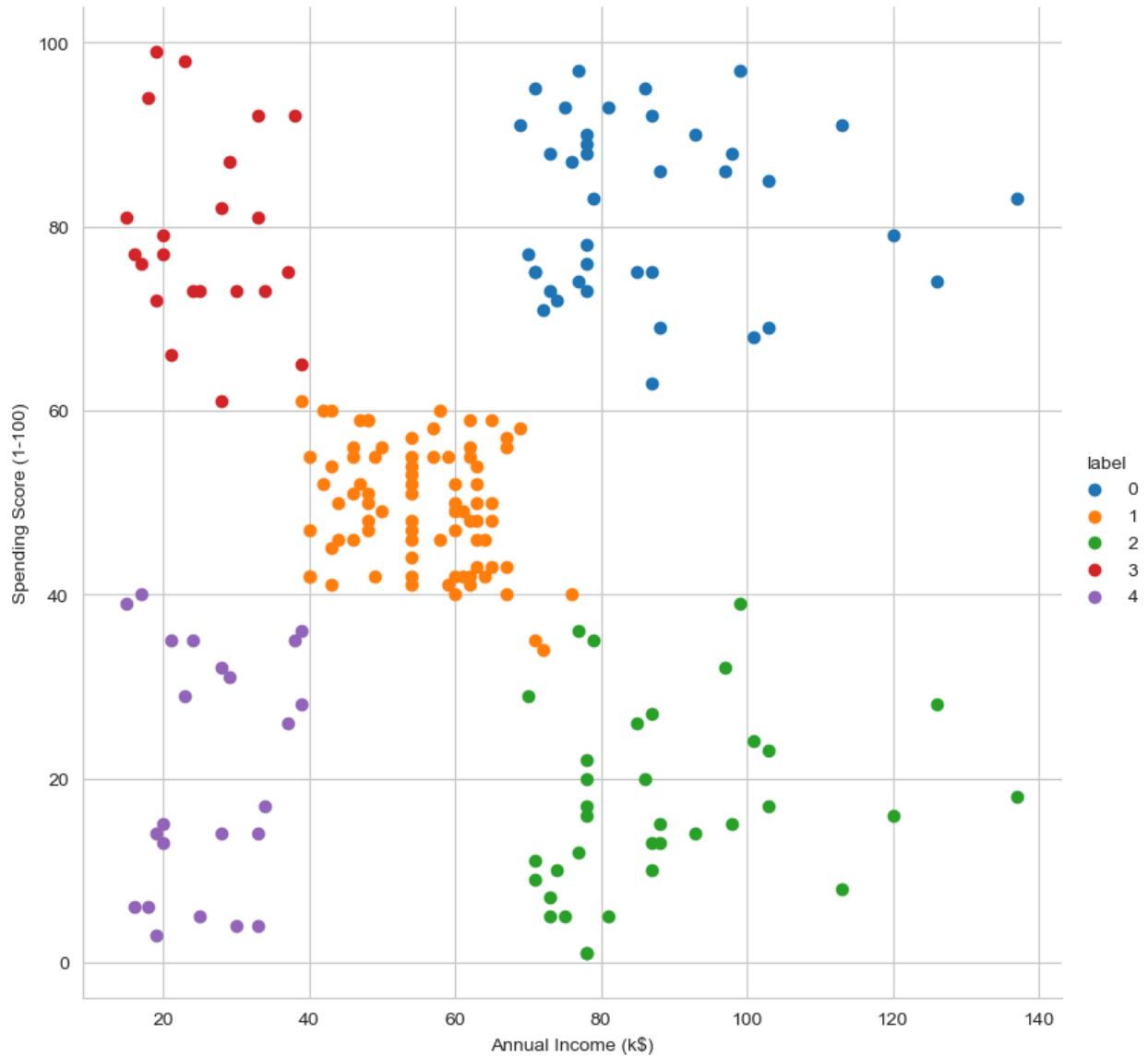
```
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
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
```

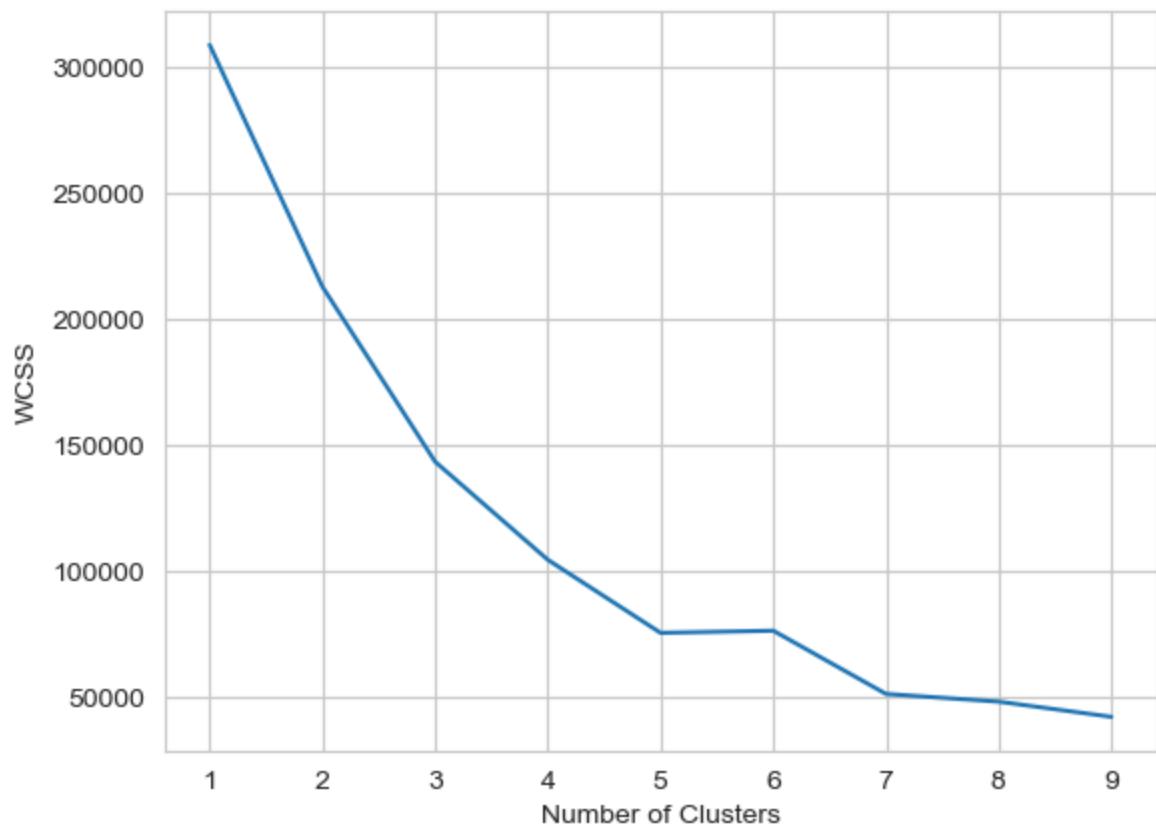
```
    Final['label'] = model.predict(features)
```





```
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
    warnings.warn(
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
    warnings.warn(
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
    warnings.warn(
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
    warnings.warn(
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
    warnings.warn(
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
    warnings.warn(
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
    warnings.warn(
C:\Users\sownd\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
```

Elbow Method



In []:

```
In [1]: import numpy as np
from scipy import stats

marks = np.array([72, 68, 75, 70, 74, 69, 71, 73, 70, 72])
mu_0 = 70

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.
```

```
In [2]: 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(f"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: 0.0164
Reject Null Hypothesis → Mean is significantly different from 50 g.
```

```
In [3]: 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.
```

In []: