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

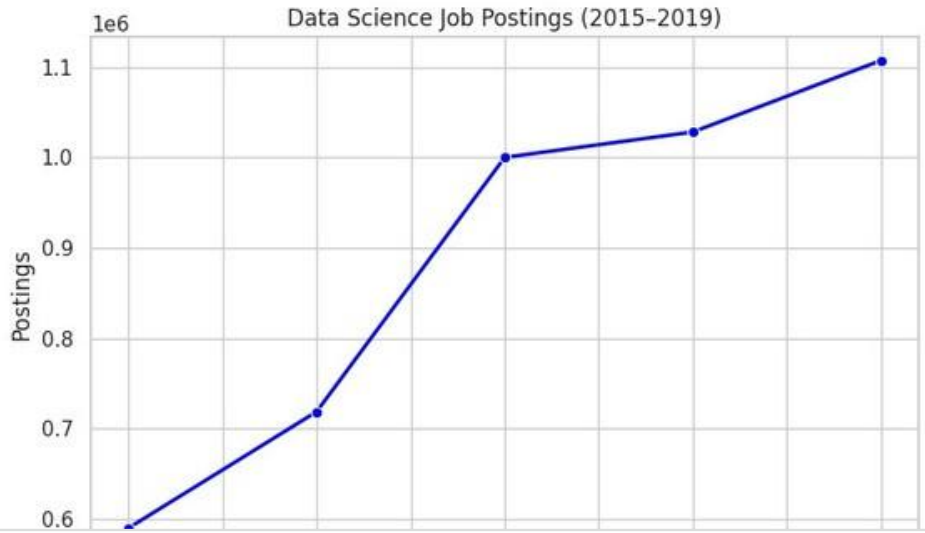
III semester II Year (2023R)

Name of the Student: PRAGADEESH S

Register Number:240701388

```
#1.a
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = {
    'Year': [2015, 2016, 2017, 2018, 2019],
        'Postings': [590184, 718846, 1000000, 1028056, 1107138]
        }

df = pd.DataFrame(data)
df['Growth_%'] = df['Postings'].pct_change() * 100
start_value = df['Postings'].iloc[0]
end_value = df['Postings'].iloc[-1]
years = df['Year'].iloc[-1] - df['Year'].iloc[0]
CAGR = ((end_value / start_value) ** (1 / years) - 1) * 100
print(f"CAGR: {CAGR:.2f}%")

sns.set_theme(style="whitegrid")
plt.figure(figsize=(8,5))
sns.lineplot(data=df, x='Year', y='Postings', marker='o', linewidth=2, color='blue')
plt.title("Data Science Job Postings (2015–2019)")
plt.show()
```

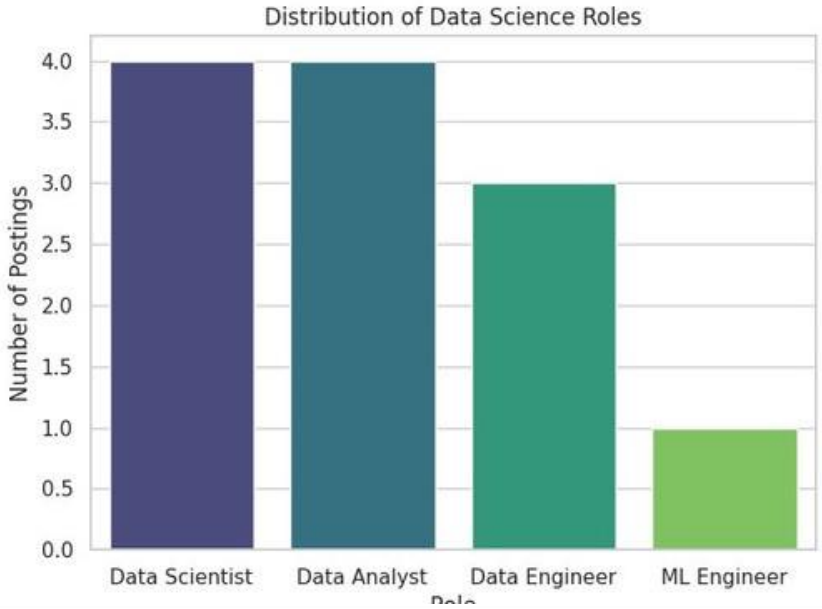
```
CAGR: 17.03%
```




```
#1.b
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = {
    'Job Title': [
            'Data Scientist', 'Data Analyst', 'Machine Learning Engineer', 'Data Engineer',
            'Data Analyst', 'Data Scientist', 'Business Intelligence Analyst',
            'Data Engineer', 'Data Scientist', 'Data Analyst', 'ML Engineer', 'Data Scientist'
      ]
}       d f       =
p d . D a t a F r
a m e ( d a t a )
def    categorize_role(title):
title   =   title.lower()   if
'scientist' in title:
        return 'Data Scientist'
      elif 'engineer' in title and 'ml' not in title:
        return 'Data Engineer'
      elif 'ml' in title or 'machine learning' in title:
        return 'ML Engineer'
       elif 'analyst' in title:
        return 'Data Analyst'
      elif 'business intelligence' in title:
        return 'BI Analyst'
      else:
        return 'Other'

df['Role'] = df['Job Title'].apply(categorize_role)

role_counts = df['Role'].value_counts()
```

```
print(role_counts)

sns.set_theme(style="whitegrid")
plt.figure(figsize=(7,5))
sns.barplot(x=role_counts.index, y=role_counts.values, palette='viridis')
plt.title('DistributionofDataScienceRoles')
plt.xlabel('Role')
plt.ylabel('NumberofPostings')
plt.show()
```

```
Role
Data Scientist    4
Data Analyst      4
Data Engineer     3
ML Engineer       1
Name: count, dtype: int64
/tmp/ipython-input-2318413288.py:38:  FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

 sns.barplot(x=role_counts.index, y=role_counts.values, palette='viridis')
```




```
#1.c
#structured data

import pandas as pd

structured_df = pd.DataFrame({
    'Employee_ID': [101, 102, 103],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [29, 35, 41],
    'Department': ['Data Science', 'Engineering', 'Analytics']
    })

print("Structured Data:\n", structured_df)

#unstructured data

unstructured_data = [
      "Alice joined the Data Science team last year. She's great at Python and ML.",
      "Bob moved from Engineering. Loves working with AWS and building ML pipelines.",
      "Charlie wrote an amazing report on analytics trends and data visualization."
       ]
print("\nUnstructured Data (Text Documents):")
for doc in unstructured_data:
  print("-", doc)

#semi structured

import json

semi_structured_data = [
      {"Name": "Alice", "Skills": ["Python", "SQL"], "Experience": 3},
        {"Name": "Bob", "Skills": ["Java", "AWS"], "Experience": 5, "Certifications": ["AWS Developer"]},
            {"Name": "Charlie", "Skills": ["Tableau"], "Department": "Analytics"}
            ]
```

```
print("\nSemi-Structured Data (JSON):")
print(json.dumps(semi_structured_data, indent=2))
```

```
Structured Data:
    Employee_ID     Name Age     Department
0          101    Alice    29 Data Science
1          102      Bob   35   Engineering
2          103 Charlie    41     Analytics

Unstructured Data (Text Documents):
- Alice joined the Data Science team last year. She's great at Python and ML.
- Bob moved from Engineering. Loves working with AWS and building ML pipelines.
- Charlie wrote an amazing report on analytics trends and data visualization.

Semi-Structured Data (JSON):
[
  {
    "Name": "Alice",
    "Skills": [
    "Python",
    "SQL"
    ],
    "Experience": 3
 },
  {
    "Name": "Bob",
    "Skills": [
    "Java",
    "AWS"
    ],
    "Experience": 5,
    "Certifications": [
    "AWS Developer"
    ]
 },
  {
    "Name": "Charlie",
    "Skills": [
    "Tableau"
    ],
    "Department": "Analytics"
  }
]
```

```
#1.d

!pip install cryptography

from cryptography.fernet import Fernet

key = Fernet.generate_key()
cipher = Fernet(key)

data = "Sensitive_Data_12345".encode()

encrypted_data = cipher.encrypt(data)

decrypted_data = cipher.decrypt(encrypted_data)

print("Encryption Key:", key.decode())
print("Original Data:", data.decode())
print("Encrypted Data:", encrypted_data.decode())
print("Decrypted Data:", decrypted_data.decode())
```

```
Requirement already satisfied: cryptography in /usr/local/lib/python3.12/dist-packages (43.0.3)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.12/dist-packages (from cryptography) (2.0.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from cffi>=1.12->cryptography) (2.23)
Encryption Key: m-aPpoK7WBvwntmkaJaNH8rXXOLCrmQhqrllppImFuw=
Original Data: Sensitive_Data_12345
Encrypted Data: gAAAAABpBeL2iZ5ZOKarggDY0-515odnbWbOhjCHHhgL_bwhbXptP7sA47NVQgSjHClIpmKFxL9BxFCTdM9dCi6ZhacyHRx7HYhAMp0vt_c8
Decrypted Data: Sensitive_Data_12345
```

```
#3.a

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

import numpy as np
import pandas as pd

df = pd.read_csv("pre_process_datasample.csv")
print("Original Data:\n", df, "\n")

df.info()

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)

updated_dataset = pd.concat([pd.get_dummies(df['Country']), df[['Age', 'Sa
updated_dataset['Purchased'].replace(['No', 'Yes'], [0, 1], inplace=True)

print("\n☑ Cleaned and Encoded Dataset:\n", updated_dataset)
updated_dataset.info()
```

```
#3.b

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

iCmphorotse nFuilmepsy parse _npproces…asample.csv

pirmep_oprrto cpeasnsd_adsa taass apmdple.csv(text/csv) - 226 bytes, last modified: 10/29/2025 - 100

```
Sdafv=ipndg. rperaed_pcrsovc(e"sHso_tdealt_aDsaatmapsleet.csv "t)o pre_process_datasa
Odrfiginal Data:
d f . dCuopulnitcraySalary (Purchased
0    iFnrfaon(c)e 44.0 72000.0   N o
d f . d rSoppa_idnu p l2i7c.a0t e s 4(8i0n0p0l.a0c e = T r u e )Yes
1                                         No
2 Germany 30.0 54000.0                    No
df
3    Spain 38.0 61000.0                  Yes
4l e nG(edrfm)any 40.0NaN
5i n d eFxr=annpc.ea r r3a5y.(0l i s5t8(0r0a0n.g0e ( 0 , l e n (Ydefs))))
6d f . s eStp_aiinnd e x (NianNd e x5,2i0n0p0l.a0c e = T r u e ) No
7i n d eFxrance 48.0 79000.0            Yes
8 Germany 50.0 83000.0                  No
9d f France 37.0 67000.0                Yes
df.drop(['Age_Group.1'],axis=1,inplace=True)
<dcflass 'pandas.core.frame.DataFrame'>
Rafn.gCeuIsntdoemxe:r I1D0. leonct[rdife.sC,u s0t otmoe r9ID<0]=np.nan
Daft.aB icloll.ulmoncs[ d(ft.oBtiallI <40 ]c=onlpu.mnnasn):
 #    Column     Non-Null Count Dtype
-d-f-. E s-t-i-m-a-t-e d S a l a-r-y-.-l-o-c-[-d-f-.-E-s-t-i m a-t-e-d-S-a l ary<0]=n
 d0f
      Country    10 non-null     object
d1f [ ' NAogOef P a x ' ] . l o9c [n(odnf-[n'uNlolO f P a x ' ]f<l1o)a t|6 4(df['NoOf
 d2f  Salary     9 non-null      float64
 d3f . A gPeu_rGcrhoauspe.du n i1q0u e n(o)n-null object
dtfy.pHeost:e lf.luonaitq6u4e(2)), object(2)
mdefm.oHroyt euls.argeep:l a4c5e2(.[0'+I bbyyst'e]s,'Ibis',inplace=True)
df.FoodPreference.unique
d f . FColoedaPnreedf earnedn cEen croedpelda
cDea(t[a'sVeetg:etarian', 'veg'], 'Veg',inp d f . FForoadnPcree f eGreernmcaen.yr e
0SIpaacien( [ ' nAogne- V e gS a]].a rNyo n -
      True    False False 44.0 72000.0          0
 df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
1
    False     False    True 27.0 48000.0         1
2d f . NFoaOlfsPea x . f i lTlrnuae( r oFuanlds(ed f .3N0o.O0f P a5x4.0m0e0d.i0a n (
3d f [ 'FRaaltsien g ( 1 -F5a)l's]e. f i lTlrnuae( r o3u8n.d0( d f6[1'0R0a0t.i0n g (
4d f . BFiallls.ef i l l n aT(rruoeu n dF(adlfs.eB i l4l0..m0e a n6(3)7)7,8i.n0p l a
5d f  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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column     Non-Null Count Dtype
--- ------      -------------- -----
F0ance          10 non-null     bool
G1rmany         10 non-null     bool
S2ain           10 non-null     bool
A3e             10 non-null     float64
S4lary          10 non-null     float64
P5rchased 10 non-null           int64
dtypes: bool(3), float64(2), int64(1)
memory usage: 402.0 bytes
/tmp/ipython-input-194434866.py:14: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b

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

```
 d[Couty].          a(d[CoutyU]n.titleodd0.eip(yn)b[ -0 C]p.la c l a b    ue)
/tmp/ipython-input-194434866.py:15: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b

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


   df['Age'].fillna(df['Age'].median(),   inplace=True)   /tmp/ipython-input-
194434866.py:16: FutureWarning: A value is trying to be se The behavior will
change in pandas 3.0. This inplace method will never work b

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


  df['Salary'].fillna(round(df['Salary'].mean()),  inplace=True)  /tmp/ipython-
input-194434866.py:19: FutureWarning: A value is trying to be se The behavior
will change in pandas 3.0. This inplace method will never work b

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


 updated_dataset['Purchased'].replace(['No', 'Yes'], [0, 1], inplace=True)
/tmp/ipython-input-194434866.py:19: FutureWarning: Downcasting behavior in `r
updated_dataset['Purchased'].replace(['No', 'Yes'], [0, 1], inplace=True)
```

In [1]:

```
#13 import numpy
import scipy.stats
np.random.seed(a4s2np
                        as stats



sample_size    = 25
sample_data    = np.random.normal(loc=102, scale=15, size=sample_size)
population_mean
                    = 100

sample_mean = np.mean(sample_data)
sample_std    = np.std(sample_data, ddof=1)


n     = len(sample_data)

t_statistic, p_value        = stats.ttest_1samp(sample_data,
population_mean)
print(f"Sample Mean:
{sample_mean:.2f}")
print(f"T-Statistic: {t_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")

alpha  = 0.05
if p_value      < alpha:
   print("Reject the null hypothesis: The average IQ score is significantly dif
else:
   print("Fail to reject the null hypothesis: There is no significant differenc
```

```
Sample Mean: 99.55
T-Statistic: -0.1577
P-Value: 0.8760
Fail to reject the null hypothesis: There is no significant difference in avera
ge IQ score from 100.
```

In [1]:

```
#12

import numpy   as np
import scipy.stats       as stats

sample_data   = np.array([152, 148, 151, 149, 147, 153, 150, 148, 152,
149,151, 150, 149, 152, 151, 148, 150, 152, 149, 150,148, 153, 151,
150, 149, 152, 148, 151, 150, 153])

population_mean   = 150

sample_mean = np.mean(sample_data)
sample_std    = np.std(sample_data, ddof=1)


n z=stat(sample_data)
np.sqrt(n))
                  = (sample_mean   - population_mean)       / (sample_std       /



p_value   = 2  * (1    - stats.norm.cdf(np.abs(z_statistic)))


print(f"Sample Mean: {sample_mean:.2f}")
print(f"Z-Statistic: {z_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")

alpha   = 0.05
if p_value      < alpha:
   print("Reject the null hypothesis: The average weight is significantly diffe
else:
   print("Fail to reject the null hypothesis: There is no significant differenc
```

```
Sample Mean: 150.20
Z-Statistic: 0.6406
P-Value: 0.5218
Fail to reject the null hypothesis: There is no significant difference in avera
ge weight from 150 grams.
```

In [ ]:

```
#7

from google.colab      import files
uploaded  = files.upload()
import numpy
import pandas  as np
from sklearn.model_selection
                                    import train_test_split
from sklearn.linear_model        import LinearRegression
import pickle

df  = pd.read_csv('Salary_data.csv')
print("=== Dataset Preview ===")
display(df)

print("\n=== Dataset Info ===")
df.info()
df.dropna(inplace=True)

print("\n=== After Dropping Missing Values ===")
df.info()
print("\n=== Descriptive Statistics ===")
display(df.describe())



features    = df.iloc[:, [0]].values
label    = df.iloc[:, [1]].values
x_train, x_test, y_train, y_test
                                        = train_test_split(features, label, test_size

model = LinearRegression()
model.fit(x_train, y_train)
print("\n=== Model Performance ===")

print("Training Score:",          model.score(x_train,        y_train))
print("Testing Score:",         model.score(x_test,       y_test))
print("Coefficient:", model.coef_)
print("Intercept:", model.intercept_)

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"\nEstimated Salary for          {yr_of_exp}    years of experience is        {Salary[0][0
```

UBprloowadsew..i.dgetis onlyavailable whenthecellhasbeenexecuted inthecurrentbrowsersession.Please rerunthiscell to enable.

```
Saving Salary_data.csv to Salary_data (1).csv
=== Dataset Preview ===
```

	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

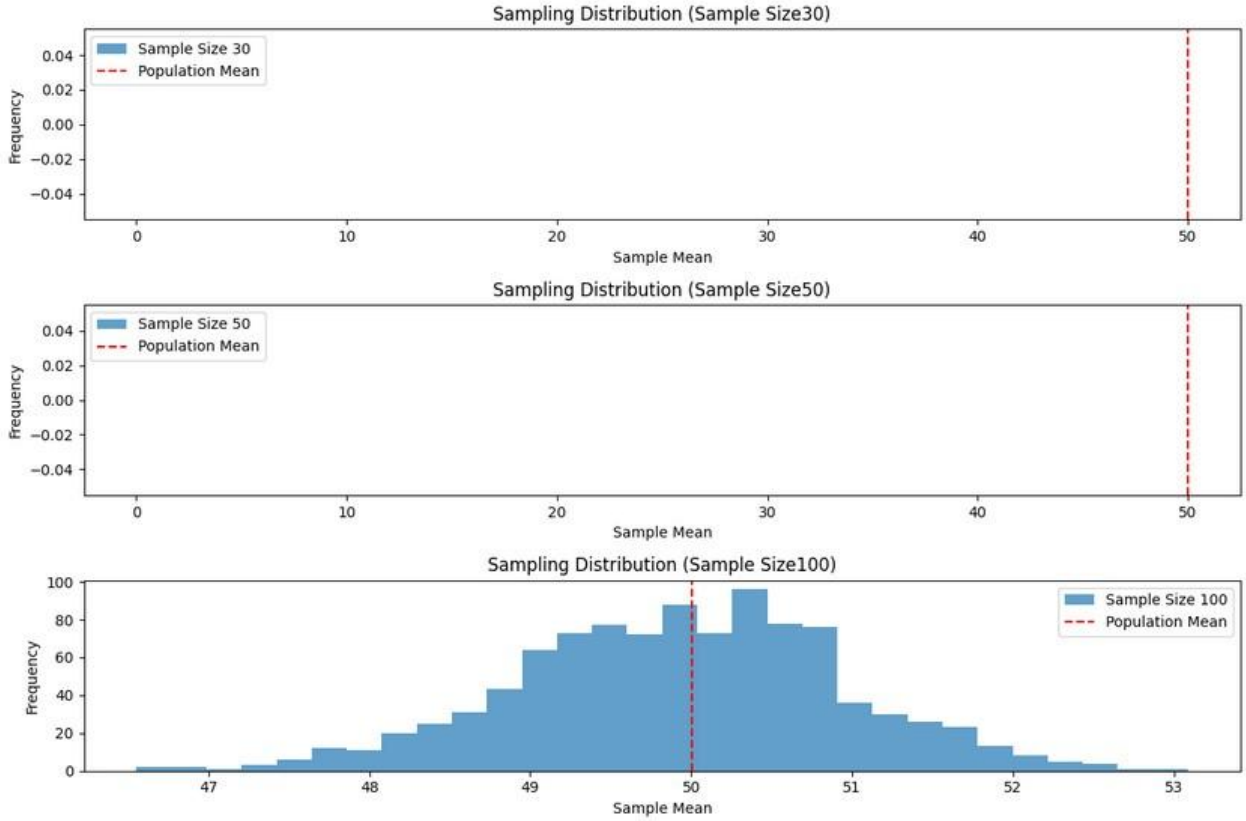
```
=== Dataset Info === <class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29 Data columns (total 2 columns):
 #   Column          ------ Non-Null Count --- Dtype --
 ---  YearsExperience      ----------- 30 non- ---
 0   Salary                null 30 non-null     float64
 1        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

 #   Column          ------ Non-Null Count --- Dtype --
 ---  YearsExperience      ----------- 30 non- ---
 0   Salary                null 30 non-null     float64
 1                                              int64
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
=== Descriptive Statistics ===
```

	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

```
=== Model Performance ===
Training Score: 0.9603182547438908
Testing Score: 0.9184170849214232
Coefficient: [[9281.30847068]]
Intercept: [27166.73682891]
Enter Years of Experience: 4
Estimated Salary for 4.0 years of experience is 64291.97
```

In [1]:

```
#11
import numpy
import matplotlibs.pnypplot
                                as plt
population_mean    = 50
population_std     = 10
population_size     = 100000
population    =  np.random.normal(population_mean,  population_std,  population_size
sample_sizes
num_samples   = [30, 50, 100]
sample_means= 1000
                  = {}
for size          in sample_sizes:
   sample_means[size]    = []

for _    in range(num_samples):
      sample  = np.random.choice(population, size=size, replace=False)
      sample_means[size].append(np.mean(sample))

plt.figure(figsize=(12,  8))
for i, size       in enumerate(sample_sizes):
   plt.subplot(len(sample_sizes), 1, i+1)
   plt.hist(sample_means[size],       bins=30,alpha=0.7,label=f'S ample Size          {size}'
   color='red',   linestyle   plt.title(f'Sampling   Distribution   (Sample      = 'dashed', linewidt
   Size{size})')    plt.xlabel('Sample    Mean')    plt.ylabel('Frequency')
   plt.legend()



plt.tight_layout()
plt.show()
```


Sampling Distribution (Sample Size30)
Sample Size 30
Population Mean
Frequency
0.04
0.02
0.00
−0.02
−0.04
0
10
20
30
40
50
Sample Mean
Sampling Distribution (Sample Size50)
Sample Size 50
Population Mean
Frequency
0.04
0.02
0.00
−0.02
−0.04
0
10
20
30
40
50
Sample Mean
Sampling Distribution (Sample Size100)
Sample Size 100
Population Mean
Frequency
100
80
60
40
20
0
47
48
49
50
51
52
53
Sample Mean

In [5]:

```
#5

from google.colab      import files
uploaded  = files.upload()
import numpy
import pandas  as np
from sklearn.impu[illegible]
from sklearn.preprocessing import SimpleImputer
                                    import LabelEncoder, StandardScaler, MinMaxScaler

filename  = list(uploaded.keys())[0]
df  = pd.read_csv(filename)

print("Dataset Info:")
print(df.info())
print("\nFirst 5 Rows:")
print(df.head())
print("\nChecking missing values before imputation:")
print(df.isnull().sum())


imputer  = SimpleImputer(strategy='mean')
df_imputed                  = pd.DataFrame(imputer.fit_transform(df.select_dtypes(include=[np.n
                                        columns=df.select_dtypes(include=[np.number]).column

print("\nAfter Imputation (numeric columns):")
print(df_imputed.head())

label_enc    = LabelEncoder()
df_encoded   = df.copy()
for col     in  df.select_dtypes(include=['object']).columns:
        df_encoded[col]    = label_enc.fit_transform(df[col].astype(str))

print("\nAfter Label Encoding:")
print(df_encoded.head())

final_set    = df_encoded.values

print("\nFinal Dataset (as NumPy array):")
print(final_set[:5])

mms= MinMaxScaler(feature_range=(0,1))
mms.fit(final_set)
feat_minmax_scaler  = mms.transform(final_set)

print("\nAfter Min-Max Scaling:")
print(pd.DataFrame(feat_minmax_scaler).head())
print("\nScaling Done! Each feature is now between 0 and 1.")

print("Shape of final scaled data:",              feat_minmax_scaler.shape)
```

UBprloowadsew..i.dgetis onlyavailable whenthecellhasbeenexecuted in the current browser session. Please rerun this cell to enable.

Saving pre_process_datasample.csv to pre_process_datasample (4).csv
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
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
None
First 5 Rows:

	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

Checking missing values before imputation:
Country 0
Age 1
Salary 1
Purchased 0
dtype: int64
After Imputation (numeric columns):

	Age	Salary
0	44.0	72000.000000
1	27.0	48000.000000
2	30.0	54000.000000
3	38.0	61000.000000
4	40.0	63777.777778

After Label Encoding:

	Country	Age	Salary	Purchased
0	0	44.0	72000.0	0
1	2	27.0	48000.0	1
2	1	30.0	54000.0	0
3	2	38.0	61000.0	0
4	1	40.0	NaN	1

Final Dataset (as NumPy array):
[[0.0e+00 4.4e+01 7.2e+04 0.0e+00]
[2.0e+00 2.7e+01 4.8e+04 1.0e+00]
[1.0e+00 3.0e+01 5.4e+04 0.0e+00]
[2.0e+00 3.8e+01 6.1e+04 0.0e+00]
[1.0e+00 4.0e+01 nan 1.0e+00]]

After Min-Max Scaling:

	0	1	2	3
0	0.0	0.739130	0.685714	0.0
1	1.0	0.000000	0.000000	1.0
2	0.5	0.130435	0.171429	0.0
3	1.0	0.478261	0.371429	0.0
4	0.5	0.565217	NaN	1.0

Scaling Done! Each feature is now between 0 and 1.
Shape of final scaled data: (10, 4)

In [ ]:

```
#9

from google.colab      import files
uploaded  = files.upload()
import numpy
import pandas  as np
from sklearn.model_selection
                                        import train_test_split
from sklearn.neighbors      import KNeighborsClassifier
from sklearn.metrics      import classification_report

df  = pd.read_csv('Iris.csv')
print(df.info())
print(df['variety'].value_counts())
print(df.head())

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

xtrain, xtest, ytrain, ytest        = train_test_split(features, label, test_size=0.2
model_KNN = KNeighborsClassifier(n_neighbors=5)
model_KNN.fit(xtrain, ytrain)

print(classification_report(label, model_KNN.predict(features)))
```

UBprloowadsew..i.dgetis onlyavailable whenthecellhasbeenexecuted in the current browser session. Please rerun this cell to enable.

```
Saving Iris.csv to Iris.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
  #    Column       ------ Non-Null Count --- Dtype  --
  ---  sepal.length       ----------- 150 non- ---
   0   sepal.width        null 150 non-null float64
   1   petal.length       150 non-null 150 float64
   2   petal.width        non-null 150 non- float64
   3   variety            null              float64
   4                                        object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB None
variety Setosa Versicolor Virginica

                    5
                    0
                    5
Name: count, dty0pe: int64
         sepal.leng5th      sepal.width      petal.length         petal.width variety
0                 50.1              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
       [[5.1 3.5 1.4 0.2]
  [4.9 3.         1.4 0.2]
       [4.7 3.2 1.3 0.2]
       [4.6 3.1 1.5 0.2]
  [5.         3.6 1.4 0.2]
       [5.4 3.9 1.7 0.4]
       [4.6 3.4 1.4 0.3]
  [5.         3.4 1.5 0.2]
       [4.4 2.9 1.4 0.2]
       [4.9 3.1 1.5 0.1]
       [5.4 3.7 1.5 0.2]
       [4.8 3.4 1.6 0.2]
  [4.8      3.     1.4 0.1]
  [4.3      3.     1.1 0.1]
  [5.8 4.         1.2 0.2]
       [5.7 4.4 1.5 0.4]
       [5.4 3.9 1.3 0.4]
       [5.1 3.5 1.4 0.3]
       [5.7 3.8 1.7 0.3]
       [5.1 3.8 1.5 0.3]
       [5.4 3.4 1.7 0.2]
       [5.1 3.7 1.5 0.4]
  [4.6 3.6 1.         0.2]
       [5.1 3.3 1.7 0.5]
       [4.8 3.4 1.9 0.2]
  [5.      3.       1.6 0.2]
  [5.         3.4 1.6 0.4]
       [5.2 3.5 1.5 0.2]
       [5.2 3.4 1.4 0.2]
```

[4.7 3.2 1.6 0.2] [4.8
3.1 1.6 0.2] [5.4 3.4
1.5 0.4] [5.2 4.1 1.5
0.1] [5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2] [5.
3.2 1.2 0.2] [5.5 3.5
1.3 0.2] [4.9 3.6 1.4
0.1] [4.4 3. 1.3 0.2]
[5.1 3.4 1.5 0.2] [5.
3.5 1.3 0.3] [4.5 2.3
1.3 0.3] [4.4 3.2 1.3
0.2] [5. 3.5 1.6 0.6]
[5.1 3.8 1.9 0.4] [4.8
3. 1.4 0.3] [5.1 3.8
1.6 0.2] [4.6 3.2 1.4
0.2] [5.3 3.7 1.5 0.2]
[5. 3.3 1.4 0.2] [7.
3.2 4.7 1.4] [6.4 3.2
4.5 1.5] [6.9 3.1 4.9
1.5] [5.5 2.3 4. 1.3]
[6.5 2.8 4.6 1.5] [5.7
2.8 4.5 1.3] [6.3 3.3
4.7 1.6] [4.9 2.4 3.3
1.] [6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4] [5.
2. 3.5 1.] [5.9 3. 4.2
1.5] [6. 2.2 4. 1.]
[6.1 2.9 4.7 1.4] [5.6
2.9 3.6 1.3] [6.7 3.1
4.4 1.4] [5.6 3. 4.5
1.5] [5.8 2.7 4.1 1.]
[6.2 2.2 4.5 1.5] [5.6
2.5 3.9 1.1] [5.9 3.2
4.8 1.8] [6.1 2.8 4.
1.3] [6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2] [6.4
2.9 4.3 1.3] [6.6 3.
4.4 1.4] [6.8 2.8 4.8
1.4] [6.7 3. 5. 1.7]
[6. 2.9 4.5 1.5] [5.7
2.6 3.5 1.] [5.5 2.4
3.8 1.1] [5.5 2.4 3.7
1.] [5.8 2.7 3.9 1.2]

[6. 2.7 5.1 1.6] [5.4
3. 4.5 1.5] [6. 3.4
4.5 1.6] [6.7 3.1 4.7
1.5] [6.3 2.3 4.4 1.3]
[5.6 3. 4.1 1.3] [5.5
2.5 4. 1.3] [5.5 2.6
4.4 1.2] [6.1 3. 4.6
1.4] [5.8 2.6 4. 1.2]
[5. 2.3 3.3 1.] [5.6
2.7 4.2 1.3] [5.7 3.
4.2 1.2] [5.7 2.9 4.2
1.3] [6.2 2.9 4.3 1.3]
[5.1 2.5 3. 1.1] [5.7
2.8 4.1 1.3] [6.3 3.3
6. 2.5] [5.8 2.7 5.1
1.9] [7.1 3. 5.9 2.1]
[6.3 2.9 5.6 1.8] [6.5
3. 5.8 2.2] [7.6 3.
6.6 2.1] [4.9 2.5 4.5
1.7] [7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8] [7.2
3.6 6.1 2.5] [6.5 3.2
5.1 2.] [6.4 2.7 5.3
1.9] [6.8 3. 5.5 2.1]
[5.7 2.5 5. 2.] [5.8
2.8 5.1 2.4] [6.4 3.2
5.3 2.3] [6.5 3. 5.5
1.8] [7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3] [6.
2.2 5. 1.5] [6.9 3.2
5.7 2.3] [5.6 2.8 4.9
2.] [7.7 2.8 6.7 2.]
[6.3 2.7 4.9 1.8] [6.7
3.3 5.7 2.1] [7.2 3.2
6. 1.8] [6.2 2.8 4.8
1.8] [6.1 3. 4.9 1.8]
[6.4 2.8 5.6 2.1] [7.2
3. 5.8 1.6] [7.4 2.8
6.1 1.9] [7.9 3.8 6.4
2.] [6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5] [6.1
2.6 5.6 1.4] [7.7 3.
6.1 2.3] [6.3 3.4 5.6
2.4]

```
     [6.4 3.1 5.5           1.8]
[6.      3.      4.8           1.8]
     [6.9 3.1 5.4           2.1]
     [6.7 3.1 5.6           2.4]
     [6.9 3.1 5.1           2.3]
     [5.8 2.7 5.1           1.9]
     [6.8 3.2 5.9           2.3]
     [6.7 3.3 5.7           2.5]
[6.7 3.        5.2           2.3]
[6.3 2.5 5.                  1.9]
[6.5     3.    [5.2 2.            ]
3.4 5.4 5.1                  2.3]
[5.9 3.                      1.8]]
```

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

In [ ]:

```
#8

from google.colab      import files
uploaded  = files.upload()
import numpy
import pandas  as np
from sklearn.model_selection
                                        import train_test_split
from sklearn.linear_model       import LogisticRegression
from sklearn.metrics      import classification_report

df  = pd.read_csv('Social_Network_Ads.csv')

print("---- Full Dataset --")
print(df)
print("\n---- Head of Dataset ------")
print(df.head())


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

print("\n---- Features (Age, Estimated Salary) ------")
print(features)
print("\n---- Label (Purchased) ------ ")
print(label)
print("\n---- Finding Best Random State -----")



for i    in range(1, 401):
      x_train, x_test, y_train, y_test          = train_test_split(features, label, test_
      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:      {:.3f}     | Train:     {:.3f}     | Random State:          {}".format(test_scor
                                                  = train_test_split(features, label, test_size
x_train, x_test, y_train, y_test
finalModel    = LogisticRegression()
finalModel.fit(x_train, y_train)

print("\n---- Final Model Accuracy------ ")
print("Train Accuracy:",          finalModel.score(x_train,          y_train))
print("Test Accuracy:",        finalModel.score(x_test,          y_test))

print("\n---- Classification Report ----- ")
print(classification_report(label,  finalModel.predict(features)))
```

UBprloowadsew..i.dgetis onlyavailable whenthecellhasbeenexecuted in the current browser session. Please rerun this cell to enable.

```
Saving Social_Network_Ads.csv to Social_Network_Ads.csv
                                                  ---- Full Dataset ----
          User ID              Age      EstimatedSalary Purchased Gender
0   1  15624510               19                19000         M0ale
2   3  15810944               35                20000         M0ale
4   ..  15668575               26                43000       Fem0ale
395    15603246               27                57000       Fem0ale
396    15804002               19                76000         M0ale
397         ...              ...                  ...       ... ...
398    15691863               46                41000       Fem1ale
399    15706071               51                23000         M1ale
       15654296               50                20000       Fem1ale
       15755018               36                33000         M0ale
      15594041                49                36000       Fem1ale
                                                [400 rows x 5 columns]
                                                  ---- Head of Dataset ----
                                                                    Age
                                                                     19
      User ID  Gender          EstimatedSalary  Purchased          35
  0 15624510    Male                    19000          0          26
  1 15810944    Male                    20000          0          27
  2 15668575  Female                    43000          0          19
  3 15603246  Female                    57000          0
  4 15804002    Male                    76000          0
            ---- Features (Age, Estimated Salary) ----

 [[     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]]
                                                          ---- Label (Purchased) ----
                  [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 0 0 0 1 0 0 0 0 0
                   0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 1 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 1 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]

---- Finding Best Random State Test: 0.900 | Train: 0.841 | Random State: 4 Test: 0.863 | Train: 0.850 | Random State: 5 Test: 0.863 | Train: 0.859 | Random State: 6 Test: 0.887 | Train: 0.838 | Random State: 7 Test: 0.863 | Train: 0.838 | Random State: 9 Test: 0.900 | Train: 0.841 | Random State: 10 Test: 0.863 | Train: 0.856 | Random State: 14 Test: 0.850 | Train: 0.844 | Random State: 15 Test: 0.863 | Train: 0.856 | Random State: 16 Test: 0.875 | Train: 0.834 | Random State: 18 Test: 0.850 | Train: 0.844 | Random State: 19 Test: 0.875 | Train: 0.844 | Random State: 20 Test: 0.863 | Train: 0.834 | Random State: 21 Test: 0.875 | Train: 0.841 | Random State: 22 Test: 0.875 | Train: 0.841 | Random State: 24 Test: 0.850 | Train: 0.834 | Random State: 26 Test: 0.850 | Train: 0.841 | Random State: 27 Test: 0.863 | Train: 0.834 | Random State: 30 Test: 0.863 | Train: 0.856 | Random State: 31 Test: 0.875 | Train: 0.853 | Random State: 32 Test: 0.863 | Train: 0.844 | Random State: 33 Test: 0.875 | Train: 0.831 | Random State: 35 Test: 0.863 | Train: 0.853 | Random State: 36 Test: 0.887 | Train: 0.841 | Random State: 38 Test: 0.875 | Train: 0.838 | Random State: 39 Test: 0.887 | Train: 0.838 | Random State: 42 Test: 0.875 | Train: 0.847 | Random State: 46 Test: 0.912 | Train: 0.831 | Random State: 47 Test: 0.875 | Train: 0.831 | Random State: 51 Test: 0.900 | Train: 0.844 | Random State: 54 Test: 0.850 | Train: 0.844 | Random State: 57 Test: 0.875 | Train: 0.844 | Random State: 58 Test: 0.925 | Train: 0.838 | Random State: 61 Test: 0.887 | Train: 0.834 | Random State: 65 Test: 0.887 | Train: 0.841 | Random State: 68 Test: 0.900 | Train: 0.831 | Random State: 72 Test: 0.887 | Train: 0.838 | Random State: 75 Test: 0.925 | Train: 0.825 | Random State: 76 Test: 0.863 | Train: 0.841 | Random State: 77 Test: 0.863 | Train: 0.859 | Random State: 81 Test: 0.875 | Train: 0.838 | Random State: 82 Test: 0.887 | Train: 0.838 | Random State: 83 Test: 0.863 | Train: 0.853 | Random State: 84 Test: 0.863 | Train: 0.841 | Random State: 85 Test: 0.863 | Train: 0.841 | Random State: 87

Test: 0.875 | Train: 0.847 | Random State: 88 Test: 0.912 | Train: 0.838 | Random State: 90 Test: 0.863 | Train: 0.850 | Random State: 95 Test: 0.875 | Train: 0.850 | Random State: 99 Test: 0.850 | Train: 0.841 | Random State: 101 Test: 0.850 | Train: 0.841 | Random State: 102 Test: 0.900 | Train: 0.825 | Random State: 106 Test: 0.863 | Train: 0.841 | Random State: 107 Test: 0.850 | Train: 0.834 | Random State: 109 Test: 0.850 | Train: 0.841 | Random State: 111 Test: 0.912 | Train: 0.841 | Random State: 112 Test: 0.863 | Train: 0.850 | Random State: 115 Test: 0.863 | Train: 0.841 | Random State: 116 Test: 0.875 | Train: 0.834 | Random State: 119 Test: 0.912 | Train: 0.828 | Random State: 120 Test: 0.863 | Train: 0.859 | Random State: 125 Test: 0.850 | Train: 0.847 | Random State: 128 Test: 0.875 | Train: 0.850 | Random State: 130 Test: 0.900 | Train: 0.844 | Random State: 133 Test: 0.925 | Train: 0.834 | Random State: 134 Test: 0.863 | Train: 0.850 | Random State: 135 Test: 0.875 | Train: 0.831 | Random State: 138 Test: 0.863 | Train: 0.850 | Random State: 141 Test: 0.850 | Train: 0.847 | Random State: 143 Test: 0.850 | Train: 0.847 | Random State: 146 Test: 0.850 | Train: 0.844 | Random State: 147 Test: 0.863 | Train: 0.850 | Random State: 148 Test: 0.875 | Train: 0.838 | Random State: 150 Test: 0.887 | Train: 0.831 | Random State: 151 Test: 0.925 | Train: 0.844 | Random State: 152 Test: 0.850 | Train: 0.841 | Random State: 153 Test: 0.900 | Train: 0.844 | Random State: 154 Test: 0.900 | Train: 0.841 | Random State: 155 Test: 0.887 | Train: 0.847 | Random State: 156 Test: 0.887 | Train: 0.834 | Random State: 158 Test: 0.875 | Train: 0.828 | Random State: 159 Test: 0.900 | Train: 0.831 | Random State: 161 Test: 0.850 | Train: 0.838 | Random State: 163 Test: 0.875 | Train: 0.831 | Random State: 164 Test: 0.863 | Train: 0.850 | Random State: 169 Test: 0.875 | Train: 0.841 | Random State: 171 Test: 0.850 | Train: 0.841 | Random State: 172 Test: 0.900 | Train: 0.825 | Random State: 180 Test: 0.850 | Train: 0.834 | Random State: 184 Test: 0.925 | Train: 0.822 | Random State: 186 Test: 0.900 | Train: 0.831 | Random State: 193 Test: 0.863 | Train: 0.850 | Random State: 195 Test: 0.863 | Train: 0.841 | Random State: 196 Test: 0.863 | Train: 0.838 | Random State: 197 Test: 0.875 | Train: 0.841 | Random State: 198 Test: 0.887 | Train: 0.838 | Random State: 199 Test: 0.887 | Train: 0.844 | Random State: 200 Test: 0.863 | Train: 0.838 | Random State: 202 Test: 0.863 | Train: 0.841 | Random State: 203

Test: 0.887 | Train: 0.831 | Random State: 206 Test: 0.863 | Train: 0.834 | Random State: 211 Test: 0.850 | Train: 0.844 | Random State: 212 Test: 0.863 | Train: 0.834 | Random State: 214 Test: 0.875 | Train: 0.831 | Random State: 217 Test: 0.963 | Train: 0.819 | Random State: 220 Test: 0.875 | Train: 0.844 | Random State: 221 Test: 0.850 | Train: 0.841 | Random State: 222 Test: 0.900 | Train: 0.844 | Random State: 223 Test: 0.863 | Train: 0.853 | Random State: 227 Test: 0.863 | Train: 0.834 | Random State: 228 Test: 0.900 | Train: 0.841 | Random State: 229 Test: 0.850 | Train: 0.844 | Random State: 232 Test: 0.875 | Train: 0.847 | Random State: 233 Test: 0.912 | Train: 0.841 | Random State: 234 Test: 0.863 | Train: 0.841 | Random State: 235 Test: 0.850 | Train: 0.847 | Random State: 236 Test: 0.875 | Train: 0.847 | Random State: 239 Test: 0.850 | Train: 0.844 | Random State: 241 Test: 0.887 | Train: 0.850 | Random State: 242 Test: 0.887 | Train: 0.825 | Random State: 243 Test: 0.875 | Train: 0.847 | Random State: 244 Test: 0.875 | Train: 0.841 | Random State: 245 Test: 0.875 | Train: 0.847 | Random State: 246 Test: 0.863 | Train: 0.859 | Random State: 247 Test: 0.887 | Train: 0.844 | Random State: 248 Test: 0.863 | Train: 0.850 | Random State: 250 Test: 0.875 | Train: 0.831 | Random State: 251 Test: 0.887 | Train: 0.844 | Random State: 252 Test: 0.863 | Train: 0.847 | Random State: 255 Test: 0.900 | Train: 0.841 | Random State: 257 Test: 0.863 | Train: 0.856 | Random State: 260 Test: 0.863 | Train: 0.841 | Random State: 266 Test: 0.863 | Train: 0.838 | Random State: 268 Test: 0.875 | Train: 0.841 | Random State: 275 Test: 0.863 | Train: 0.850 | Random State: 276 Test: 0.925 | Train: 0.838 | Random State: 277 Test: 0.875 | Train: 0.847 | Random State: 282 Test: 0.850 | Train: 0.847 | Random State: 283 Test: 0.850 | Train: 0.844 | Random State: 285 Test: 0.912 | Train: 0.834 | Random State: 286 Test: 0.850 | Train: 0.841 | Random State: 290 Test: 0.850 | Train: 0.841 | Random State: 291 Test: 0.850 | Train: 0.847 | Random State: 292 Test: 0.863 | Train: 0.838 | Random State: 294 Test: 0.887 | Train: 0.828 | Random State: 297 Test: 0.863 | Train: 0.834 | Random State: 300 Test: 0.863 | Train: 0.850 | Random State: 301 Test: 0.887 | Train: 0.850 | Random State: 302 Test: 0.875 | Train: 0.847 | Random State: 303 Test: 0.863 | Train: 0.834 | Random State: 305 Test: 0.912 | Train: 0.838 | Random State: 306 Test: 0.875 | Train: 0.847 | Random State: 308 Test: 0.900 | Train: 0.844 | Random State: 311

Test: 0.863 | Train: 0.834 | Random State: 313 Test: 0.912 | Train: 0.834 | Random State: 314 Test: 0.875 | Train: 0.838 | Random State: 315 Test: 0.900 | Train: 0.847 | Random State: 317 Test: 0.912 | Train: 0.822 | Random State: 319 Test: 0.863 | Train: 0.850 | Random State: 321 Test: 0.912 | Train: 0.828 | Random State: 322 Test: 0.850 | Train: 0.847 | Random State: 328 Test: 0.850 | Train: 0.838 | Random State: 332 Test: 0.887 | Train: 0.853 | Random State: 336 Test: 0.850 | Train: 0.838 | Random State: 337 Test: 0.875 | Train: 0.841 | Random State: 343 Test: 0.863 | Train: 0.844 | Random State: 346 Test: 0.887 | Train: 0.831 | Random State: 351 Test: 0.863 | Train: 0.850 | Random State: 352 Test: 0.950 | Train: 0.819 | Random State: 354 Test: 0.863 | Train: 0.850 | Random State: 356 Test: 0.912 | Train: 0.841 | Random State: 357 Test: 0.863 | Train: 0.838 | Random State: 358 Test: 0.850 | Train: 0.841 | Random State: 362 Test: 0.900 | Train: 0.844 | Random State: 363 Test: 0.863 | Train: 0.853 | Random State: 364 Test: 0.938 | Train: 0.822 | Random State: 366 Test: 0.912 | Train: 0.841 | Random State: 369 Test: 0.863 | Train: 0.853 | Random State: 371 Test: 0.925 | Train: 0.834 | Random State: 376 Test: 0.912 | Train: 0.828 | Random State: 377 Test: 0.887 | Train: 0.850 | Random State: 378 Test: 0.887 | Train: 0.850 | Random State: 379 Test: 0.863 | Train: 0.841 | Random State: 382 Test: 0.863 | Train: 0.859 | Random State: 386 Test: 0.850 | Train: 0.838 | Random State: 387 Test: 0.875 | Train: 0.828 | Random State: 388 Test: 0.850 | Train: 0.844 | Random State: 394 Test: 0.863 | Train: 0.838 | Random State: 395 Test: 0.900 | Train: 0.844 | Random State: 397 Test: 0.863 | Train: 0.844 | Random State: 400 ---- Final Model Accuracy Train Accuracy: 0.8375 Test Accuracy: 0.8875 ---- Classification Report ----

	precision	recall	f1-score	support
	0.85	0.93	0.89	257
0	0.85	0.70	0.77	143
1			0.85	400
accuracy			0.83	400
macro avg			0.84	400
weighted avg	0.85	0.81		
	0.85	0.85		

In [ ]:

```
#10

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


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')
print(df.info())
print(df.head())

sns.pairplot(df)
plt.show()

features    = df.iloc[:, [3, 4]].values
from sklearn.cluster        import KMeans
model = KMeans(n_clusters=5, random_state=42)
model.fit(features)

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

sns.set_style("whitegrid")
plt.figure(figsize=(8,6))
sns.scatterplot(data=Final,          x="Annual Income (k$)",      y="Spending Score (1-100)"
plt.title("K-Means Clustering of Customers")
plt.show()

features_el      = df.iloc[:, [2, 3, 4]].values
wcss   =  []
for i     in range(1, 10):
      model  = KMeans(n_clusters=i, random_state=42)
      model.fit(features_el)
      wcss.append(model.inertia_)

plt.figure(figsize=(8,5))
plt.plot(range(1, 10), wcss, marker='o', color='blue')
plt.title("Elbow Method for Optimal K")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```

UBprloowadsew..i.dgetis onlyavailable whenthecellhasbeenexecuted in the current browser session. Please rerun this cell to enable.

In [ ]:

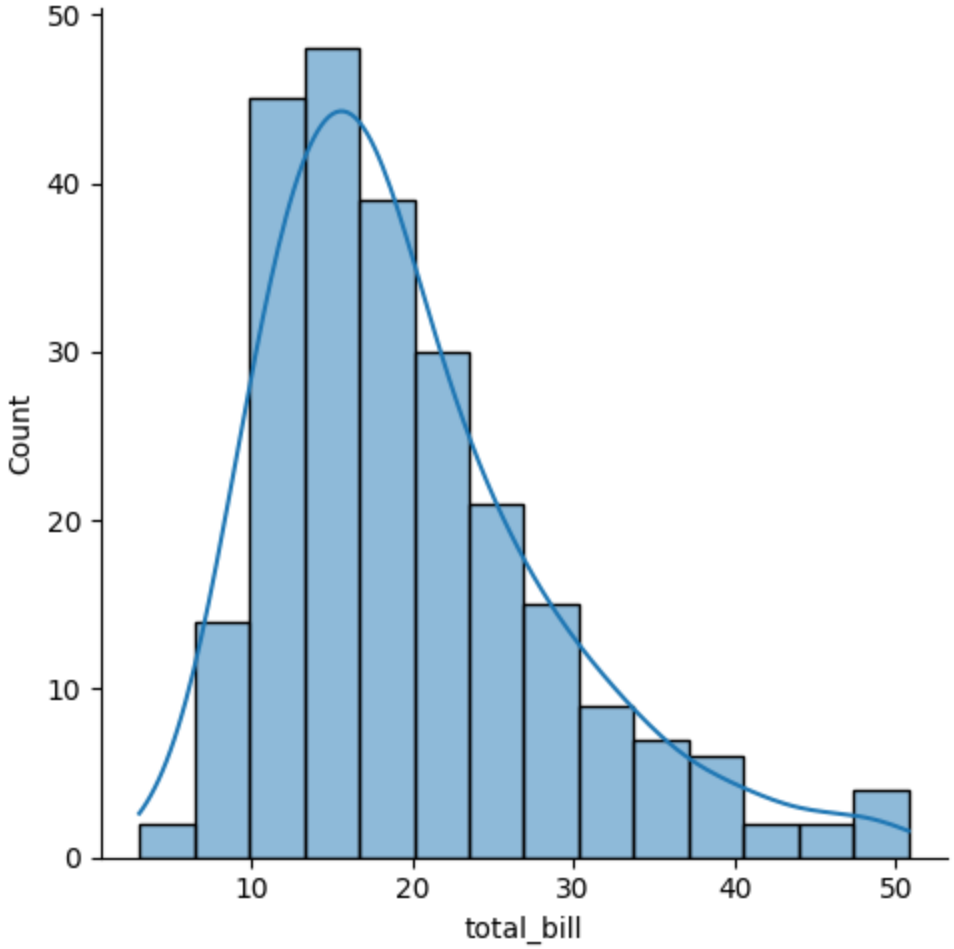
```
#6 import seaborn
import pandas
import numpy       as sns
import matplotlib.pyppdlot
%matplotlib inlina s np
tips=sns.load_dataset('tips')     as plt
tips.head()

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

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

sns.jointplot(x=tips.tip,y=tips.total_bill)

sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")

sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")

sns.pairplot(tips)                                  tips.time.value_counts()

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

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

sns.boxplot(tips.total_bill)                          sns.boxplot(tips.tip)

sns.countplot(tips.day)                             sns.countplot(tips.sex)

tips.sex.value_counts().plot(kind='pie')

tips.sex.value_counts().plot(kind='bar')

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

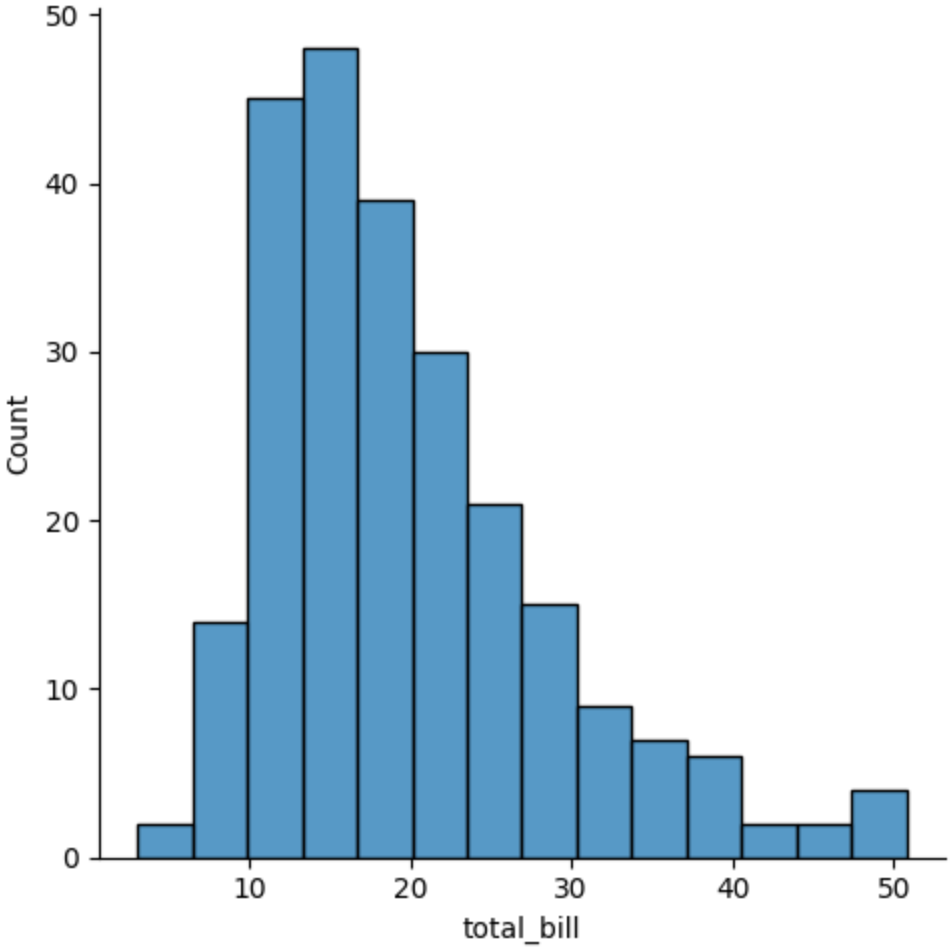
```
WARNING:matplotlib.axes._base:Ignoring fixed y limits to fulfill fixed data asp
ect with adjustable data limits.
/usr/local/lib/python3.12/dist-packages/seaborn/categorical.py:383:  UserWarnin
g: Attempting to set identical low and high ylims makes transformation singula
r; automatically expanding.
          ax.set_ylim(n - .5, -.5, auto=None)
```

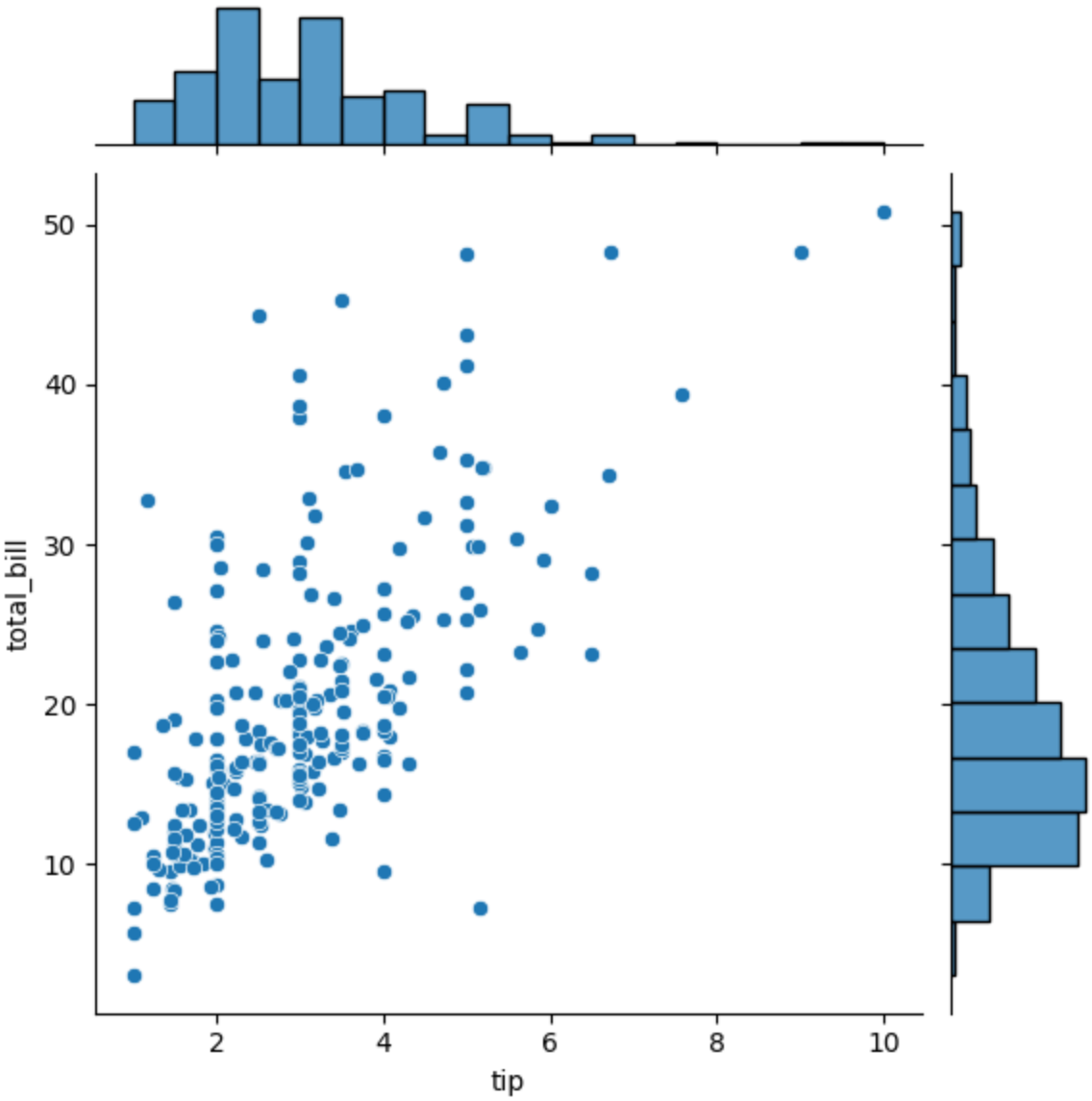
Out[ ]:

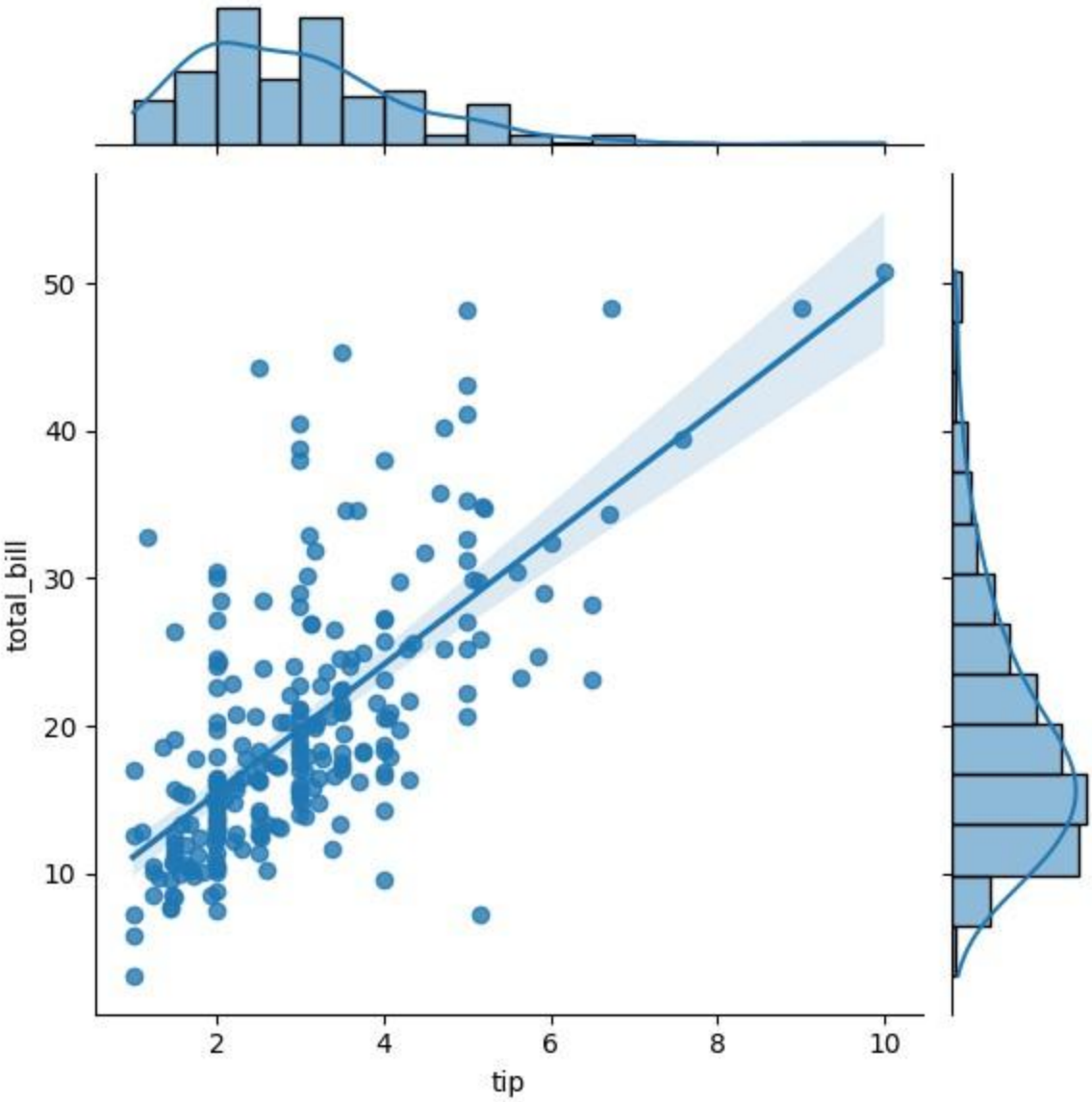
```
<Axes: xlabel='sex', ylabel='count'>
```

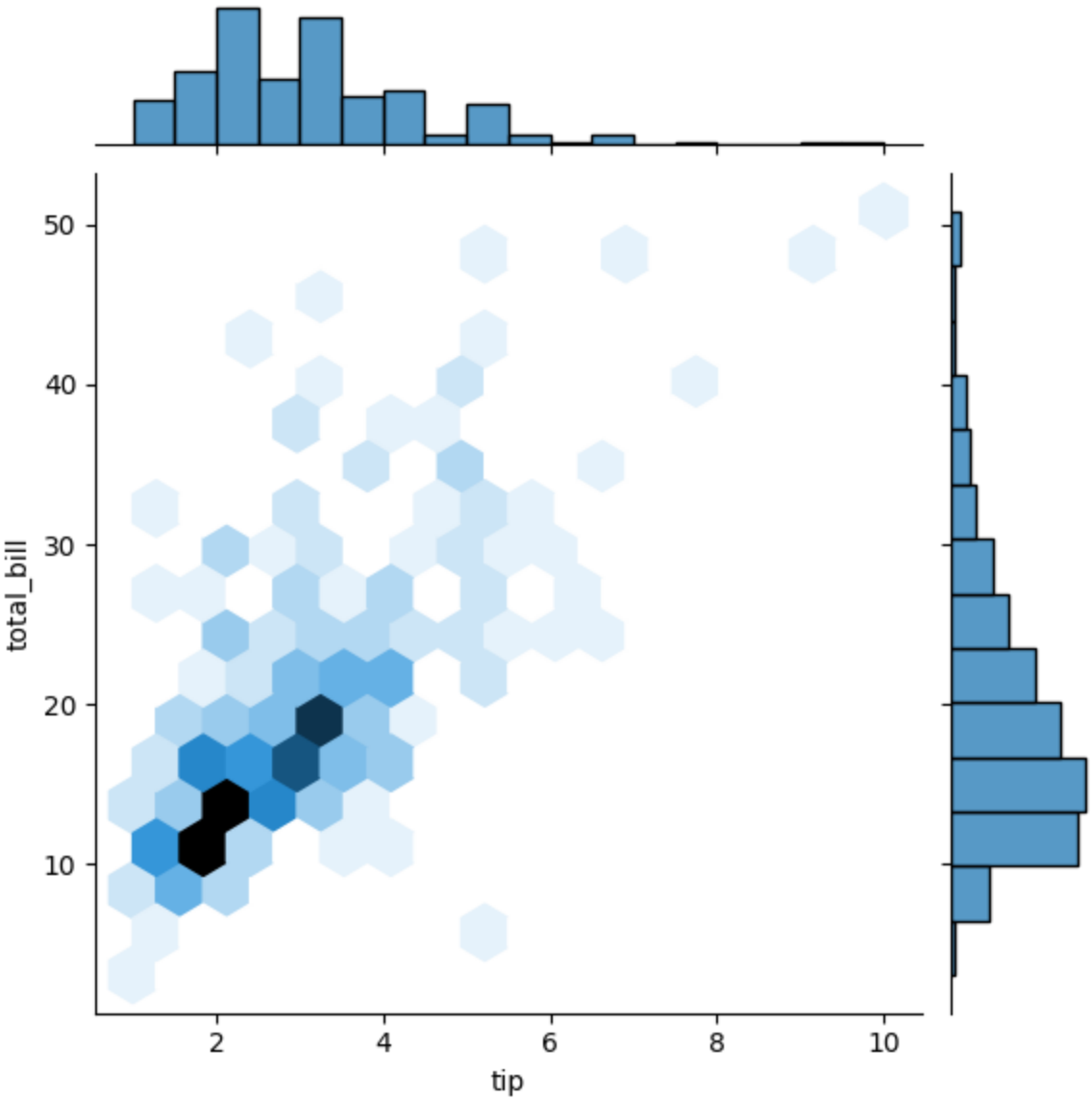
```
WARNING:matplotlib.axes._base:Ignoring fixed y limits to fulfill fixed data asp
ect with adjustable data limits.
```

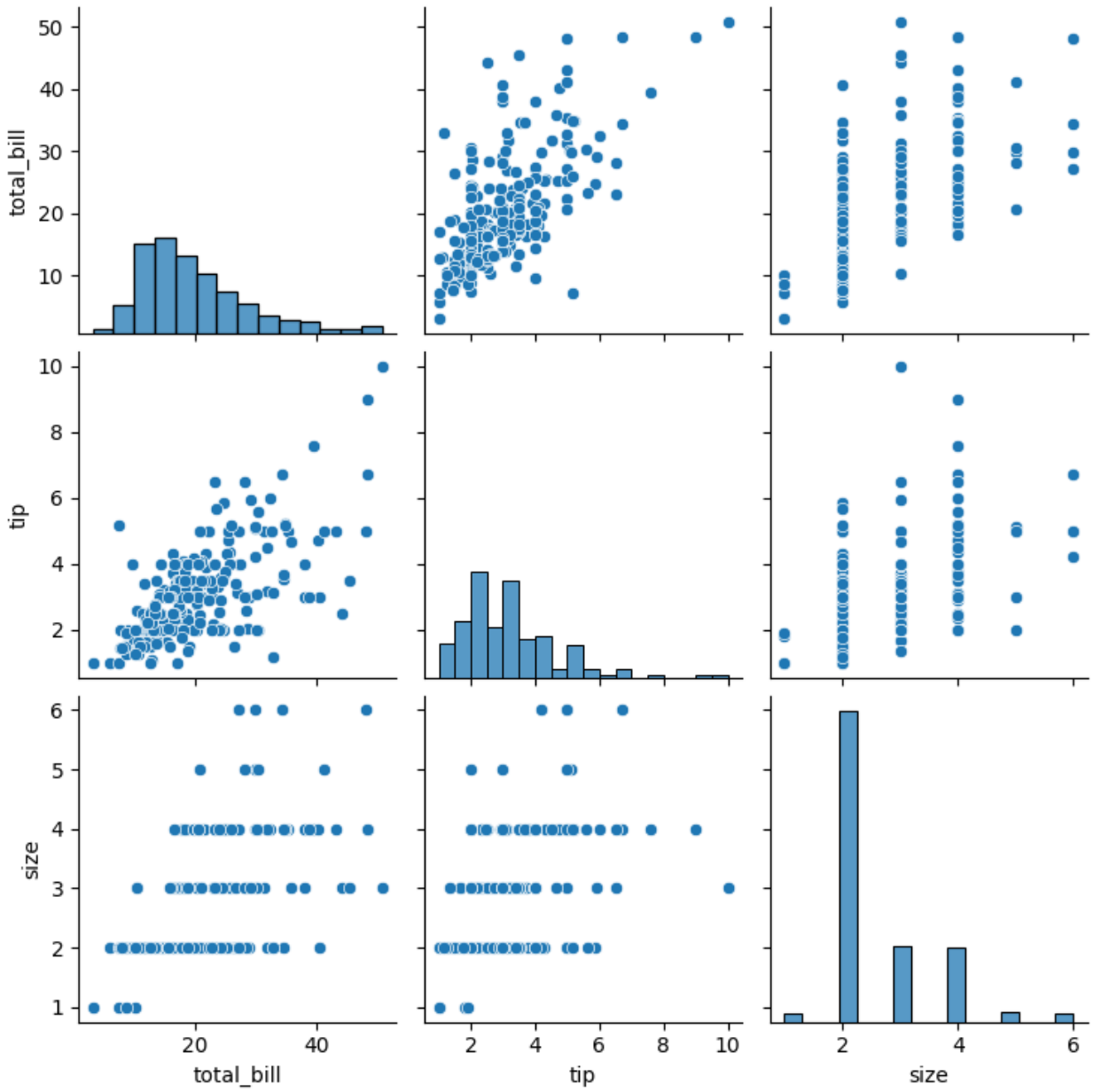

50
40
30
20
10
0
Count
10
20
30
40
50
total_bill

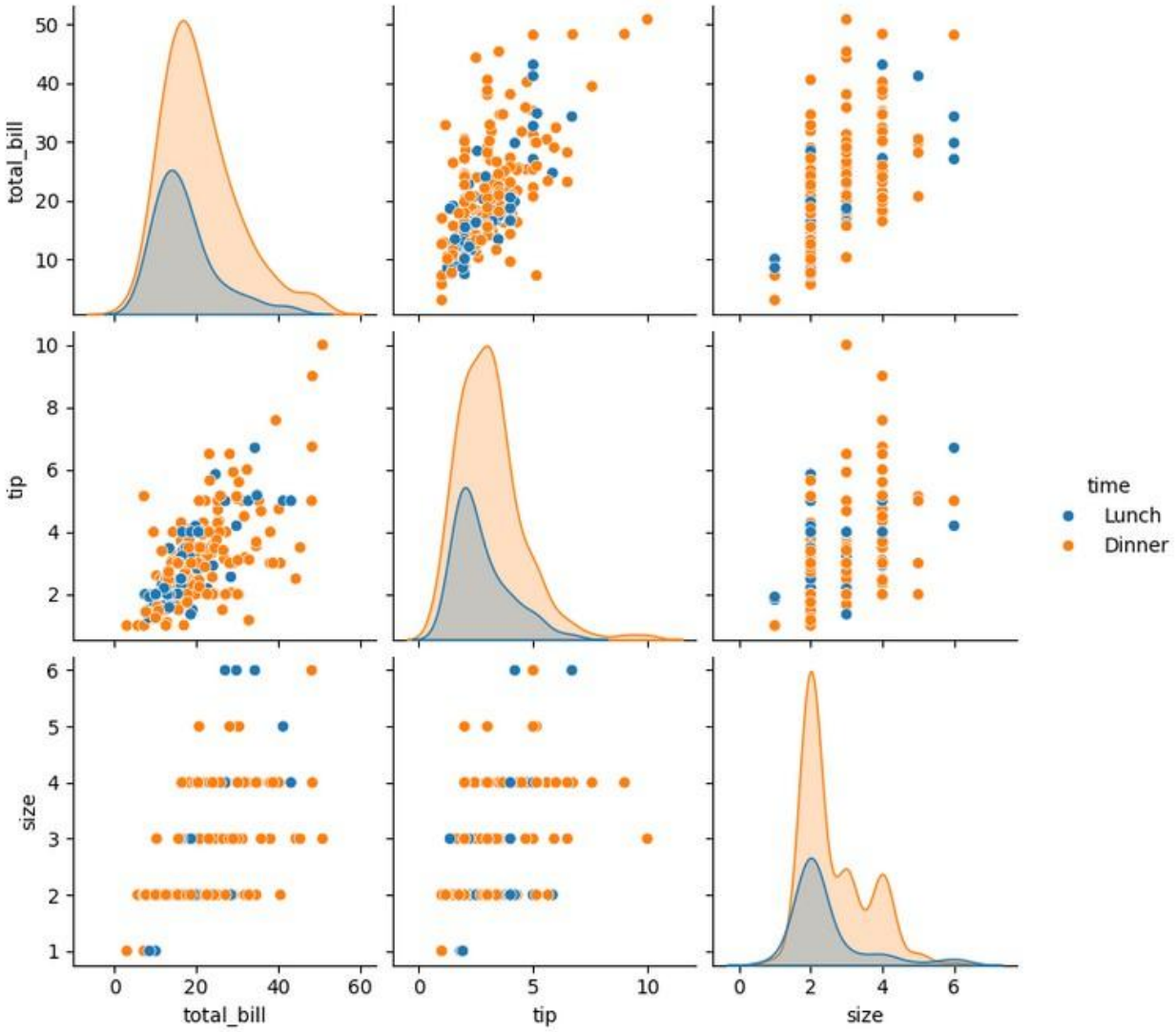

50
40
30
20
10
0
Count
10
20
30
40
50
total_bill

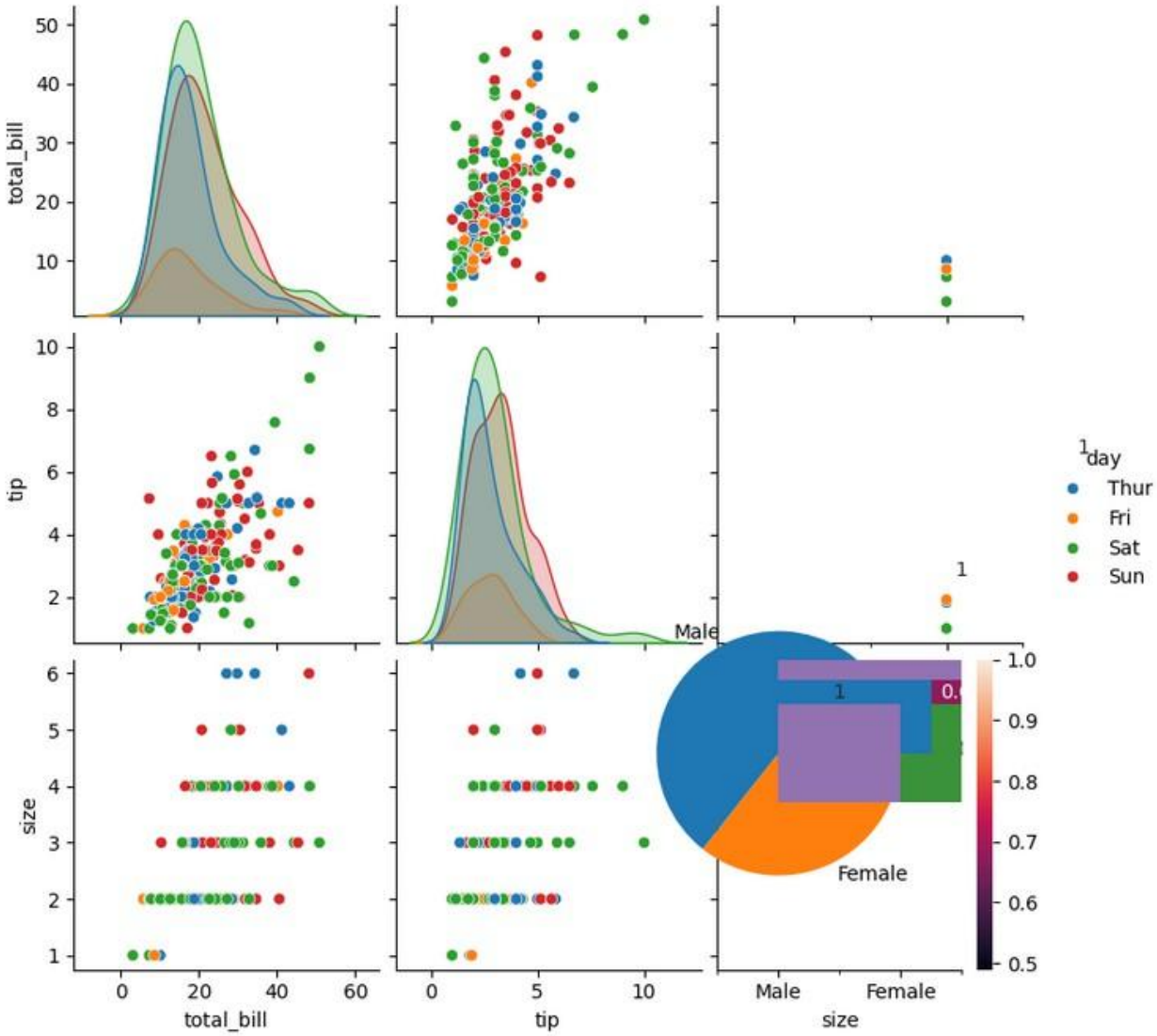

total_bill
tip
10
20
30
40
50
2
4
6
8
10


total_bill
tip
50
40
30
20
10
2
4
6
8
10


total_bill
tip
10
20
30
40
50
2
4
6
8
10


total_bill
tip
size
50
40
30
20
10
10
8
6
4
2
6
5
4
3
2
1
20
40
2
4
6
8
10
2
4
6
total_bill
tip
size


total_bill
tip
size
time
Lunch
Dinner
0
20
40
60
0
5
10
0
2
4
6
10
20
30
40
50
2
4
6
8
10
1
2
3
4
5
6

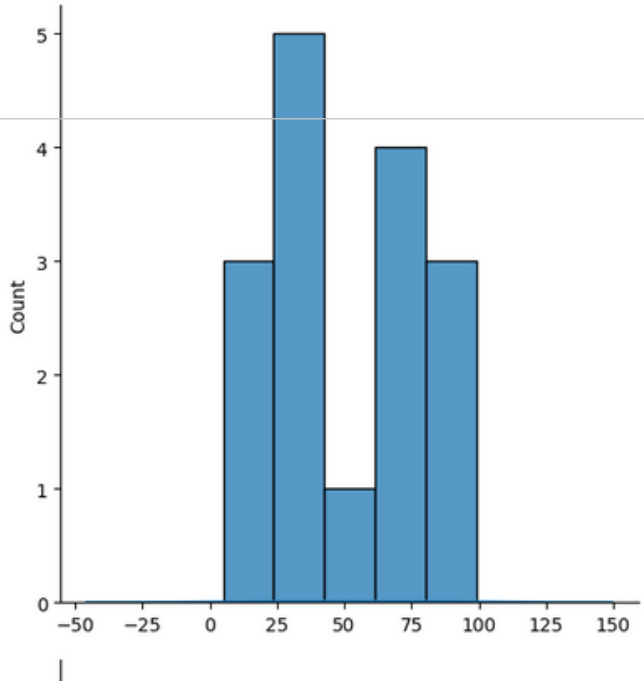

total_bill
tip
size
day
Thur
Fri
Sat
Sun
Male
Female
1
0.
0
20
40
60
5
10
Male
Female
1.0
0.9
0.8
0.7
0.6
0.5

```
#4

import       numpy       as       np
array=np.random.randint(1,100,16)
array                  array.mean()
np.percentile(array,25)
np.percentile(array,50)
np.percentile(array,75)
np.percentile(array,100)        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

import    seaborn    as    sns    %matplotlib    inline
sns.displot(array)                     sns.distplot(array)
new_array=array[(array>lr)   &   (array<ur)]   new_array
sns.displot(new_array)    lr1,ur1=outDetection(new_array)
lr1,ur1      final_array=new_array[(new_array>lr1)      &
(new_array<ur1)] final_array sns.distplot(final_array)
```

```
/tmp/ipython-input-1810417697.py:25: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

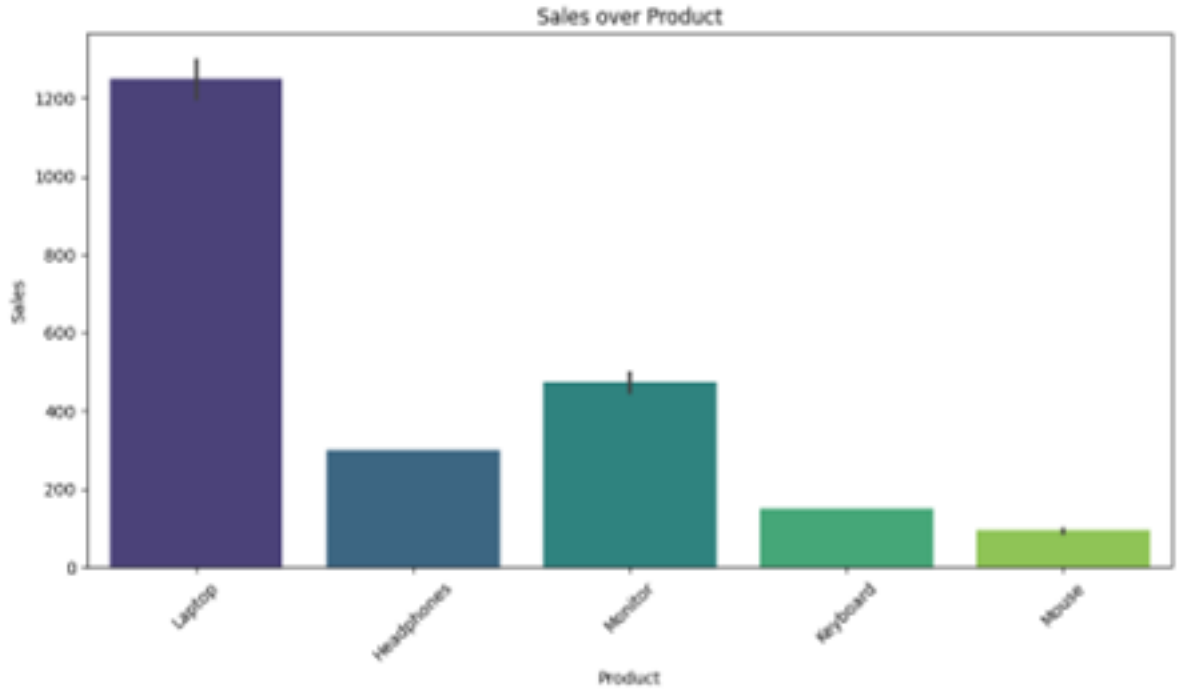
```
 sns.distplot(array)
/tmp/ipython-input-1810417697.py:33: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
 sns.distplot(final_array)
<Axes: ylabel='Count'>
```



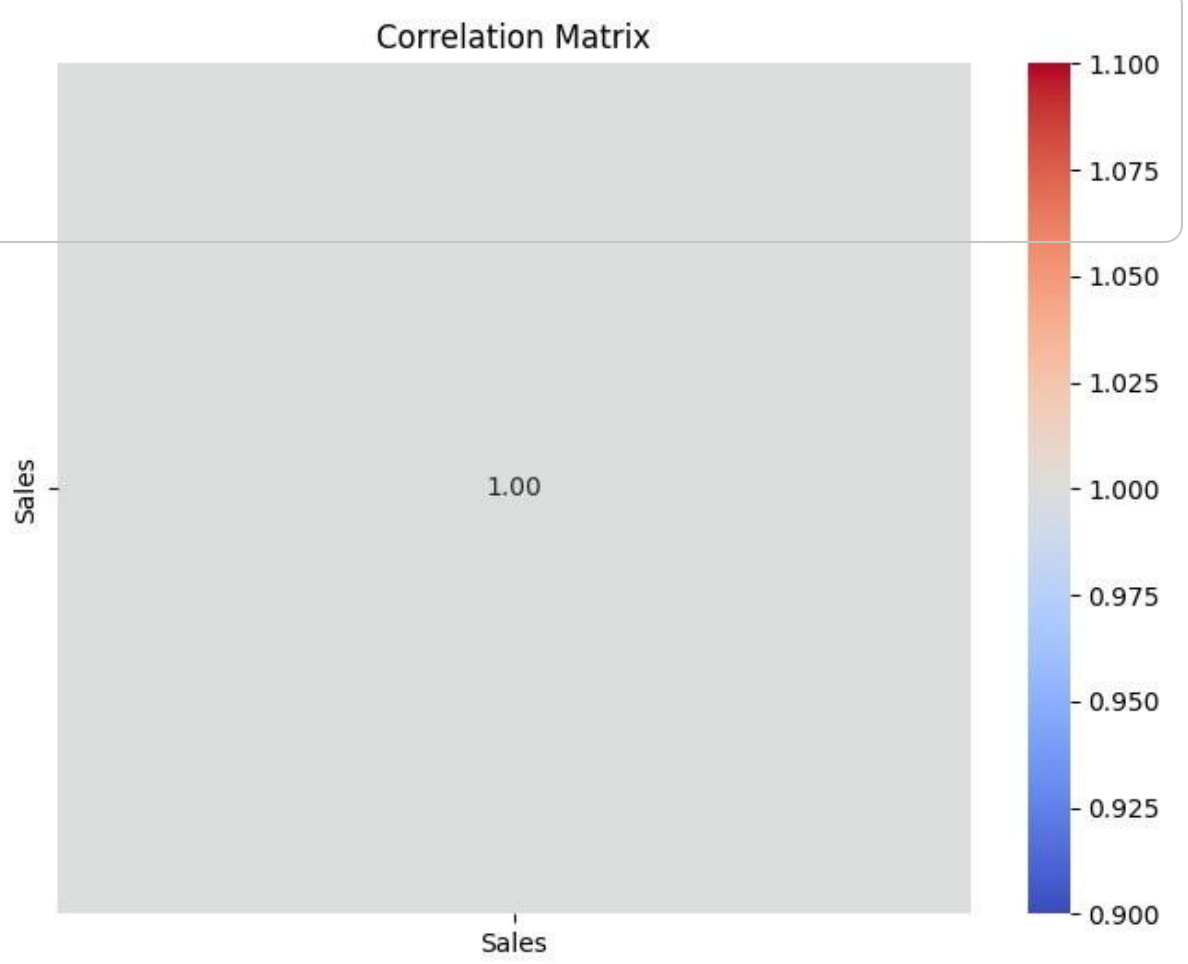
```
#2.a

import pandas as pd

data = {
    'Product': ['Laptop', 'Headphones', 'Monitor', 'Keyboard', 'Mouse', '
        'Sales': [1200, 300, 450, 150, 100, 1300, 500, 90],
            'Date': ['2024-01-01', '2024-01-03', '2024-01-05', '2024-01-07
            }

df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])
df

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,6))
sns.barplot(x='Product', y='Sales', data=df, palette='viridis')
plt.title('Sales over Product')
plt.xlabel('Product')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-338642133.py:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed i

 sns.barplot(x='Product', y='Sales', data=df, palette='viridis')
```



```
#2.b

plt.figure(figsize=(10,6))
sns.lineplot(x='Date', y='Sales', data=df, marker='o', color='teal')
plt.title('Sales over Time')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.tight_layout()
plt.show()
```


```
#2.c

plt.figure(figsize=(8,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
Sales
1.00
Sales
1.100
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900

In [1]:

```
#14   import  numpy
import      scipy.stats
np.random.seed(a4s2n)p
                        as stats



n_plants  = 25


growth_A = gnpwrtnBom.ngrowth(loc=10,all_scale=2,     size=n_plants)  =
treatment_lpbelsndom.nrmatsoic=12,     psvaaleu=e3,size=n_plants)    =
print("Treatmrandom.Anormal(loc=15,  Gscvale=2.5,size=n_plants)   =
print("Treatmconcatenate([growth_A, gGrowth_B", growth_C])
print("Treatment   C   Mean   Growth:",   print()
print(f"F-Statistic:   {f_statistic:.4f}")   print(f"P-
Value: {p_value:.4f}") ['A']    * n_plants   + ['B']    * n_plants   + ['C']    * n_plants

                        = stats.f_oneway(growth_A, growth_B, growth_C)

                                  np.mean(growth_A))
                                  np.mean(growth_B))
                                  np.mean(growth_C))




alpha  = 0.05
if p_value    < alpha:
   print("Reject the null hypothesis: There is a significant difference in mean
else:
   print("Fail to reject the null hypothesis: There is no significant differenc

if p_value    < alpha:
      from statsmodels.stats.multicomp     import pairwise_tukeyhsd
      tukey_results    = pairwise_tukeyhsd(all_data, treatment_labels, alpha=0.05)
      print("\nTukey's HSD Post-hoc Test:")

print(tukey_results)
```

Treatment A Mean Growth: 9.672983882683818
Treatment B Mean Growth: 11.137680744437432
Treatment C Mean Growth: 15.265234904828972
F-Statistic: 36.1214
P-Value: 0.0000
Reject the null hypothesis: There is a significant difference in mean growth ra
tes among the three treatments.
Tukey's HSD Post-hoc Test:
Multiple Comparison of Means - Tukey HSD, FWER=0.05
==
reject

Fgarodsuep 1 group2 meandiff p-adj lower upper
True
True A B 1.4647 0.0877 -0.1683 3.0977
------A-------C---5.5923----0.0--- 3.9593 7.2252
B C 4.1276 0.0 2.4946 5.7605