

#1.A) BASIC PRACTICE EXPERIMENTS 1-4

#230701004

#ABHINAYA LAKSHMI.S

#30/07/24

```
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')
```

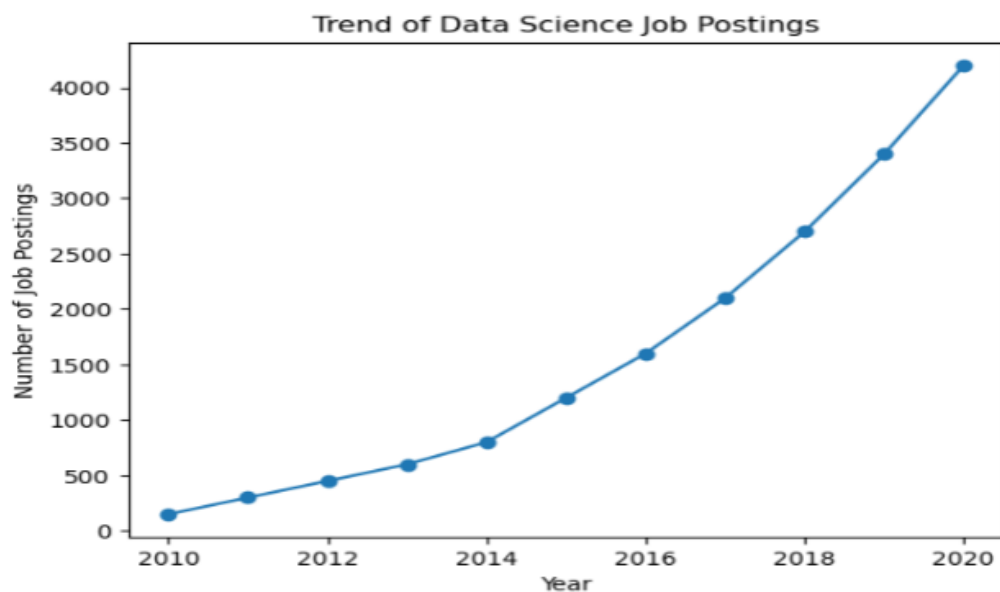
```
plt.title('Trend of Data Science Job Postings')
```

```
plt.xlabel('Year')
```

```
plt.ylabel('Number of Job Postings')
```

```
plt.show()
```

Output:



```
In [2]: import pandas
x=[1,7,2]
y=pandas.DataFrame(x,index=["a","b","c"])
print(y)
```

```

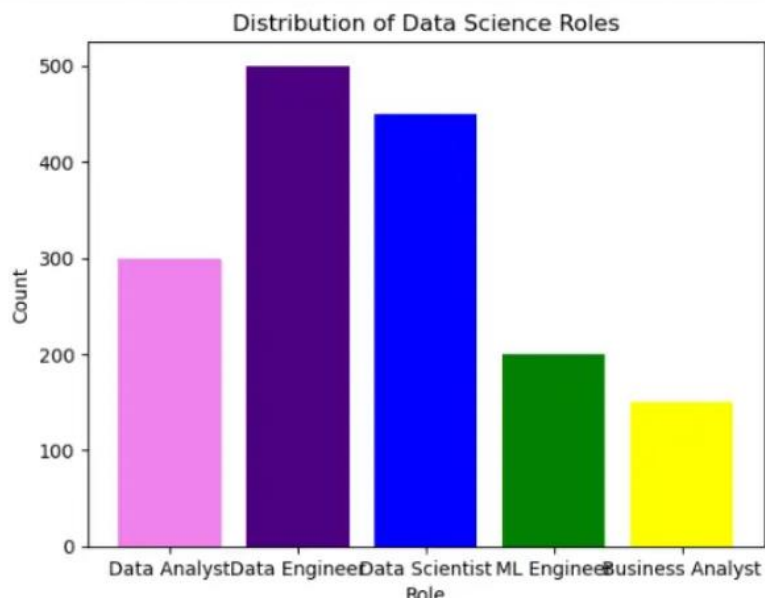
0
a  1
b  7
c  2
```

```
In [3]: import pandas
x={'Subjects':["Math","Physics","English"], 'Marks': [89,92,96]}
print(pandas.DataFrame(x))
```

```

Subjects  Marks
0    Math      89
1  Physics      92
2  English      96
```

```
In [19]: import matplotlib.pyplot as plt
roles=['Data Analyst','Data Engineer','Data Scientist','ML Engineer','Business Analyst']
counts=[300,500,450,200,150]
color=['violet','indigo','blue','green','yellow']
plt.bar(roles,counts,color=color)
plt.title('Distribution of Data Science Roles')
plt.xlabel('Role')
plt.ylabel('Count')
plt.show()
```



#1.B) PANDAS BUIT IN FUNCTION; NUMPY BUIT IN FUCTION- ARRAY SLICING, RAVEL,RESHAPE,NDIM

#230701004

#ABHINAYA LAKSHMI.S

#06.08.2024

#NUMPY FUNCTIONS

import numpy as np

import pandas as pd

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

df

df.info()

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

df.dropna(inplace=True)

df.info()

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

```
df.describe()
```

	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

```
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_st
```

```
from sklearn.linear_model import LinearRegression
```

```
model=LinearRegression()
```

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

```
▼ LinearRegression
LinearRegression()
```

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

```
0.9603182547438908
```

```
model.score(x_test,y_test)
```

```
0.9184170849214232
```

```
model.coef_
```

```
array([[9281.30847068]])
```

```
model.intercept_
```

```
array([27166.73682891])
```

```

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)

Enter Years of Experience: 44

print("Estimated Salary for {} years of experience is {}".format(yr_of_exp,Salary))

Estimated Salary for 44.0 years of experience is [[435544.30953887]]:

```

#PANDAS FUNCTIONS

```

import numpy as np

import pandas as pd

list=[[1,'Smith',50000],[2,'Jones',60000]]

df=pd.DataFrame(list)

df

```

	0	1	2
0	1	Smith	50000
1	2	Jones	60000

```

df.columns=['Empd','Name','Salary']

df

```

	Empd	Name	Salary
0	1	Smith	50000
1	2	Jones	60000

```

df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0   Empd    2 non-null         int64
1   Name    2 non-null         object
2   Salary  2 non-null         int64
dtypes: int64(2), object(1)
memory usage: 176.0+ bytes

```

```

df=pd.read_csv("/content/50_Startups.csv")

```

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   R&D Spend              50 non-null    float64
1   Administration         50 non-null    float64
2   Marketing Spend        50 non-null    float64
3   State                  50 non-null    object
4   Profit                  50 non-null    float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

df.head()

df.tail()

import numpy as np

import pandas as pd

df = pd.read_csv("/content/employee.csv")

df.head()

df.tail()

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0   emp id  7 non-null      int64
1   name    7 non-null      object
2   salary  7 non-null      int64
dtypes: int64(2), object(1)
memory usage: 296.0+ bytes
```

df.salary()

	salary
0	5000
1	6000
2	7000
3	5000
4	8000
5	3000
6	6000

type(df.salary)

df.salary.mean()

```
df.salary.median()
```

```
↪ 6000.0
```

```
df.salary.mode()
```

```
↪
```

	salary
0	5000
1	6000

```
df.salary.var()
```

```
↪ 2571428.5714285714
```

```
df.salary.std()
```

```
↪ 1603.5674514745463
```

```
empCol=df.columns
```

```
empCol
```

```
Index(['emp id', 'name ', 'salary'], dtype='object')
```

```
emparray=df.values
```

```
employee_DF=pd.DataFrame(emparray,columns=empCol)
```

```
#OUTLIER DETECTION
```

```
#230701004
```

```
#ABHINAYA LAKSHMI.S
```

```
#13.08.2024
```

```
#sample calculation for low range(lr) , upper range (ur),percentile
```

```
import numpy as np
```

```
array=np.random.randint(1,100,16) # randomly generate 16 numbers between 1 to 100
```

```
array
```

```
#array([21, 72, 69, 45, 61, 43, 43, 59, 62, 42, 90, 25, 54, 86, 80, 13], dtype=int32)
```

```
array.mean()
```

```
np.percentile(array,25)
```

```
np.percentile(array,50)
```

```
np.percentile(array,75)
```

```
np.percentile(array,100)
```

```
#outliers detection
```

```
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
```

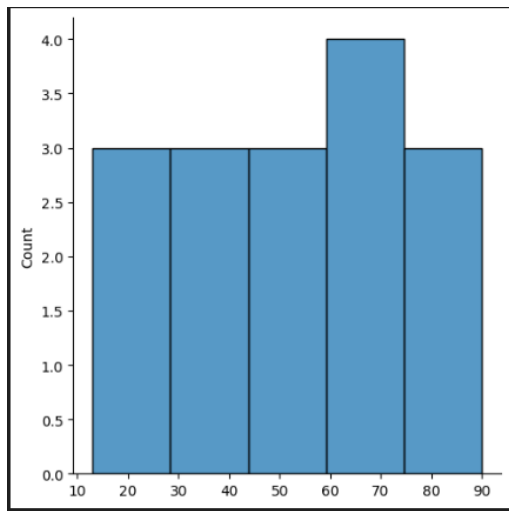
```
lr,ur=outDetection(array)
```

```
lr,ur
```

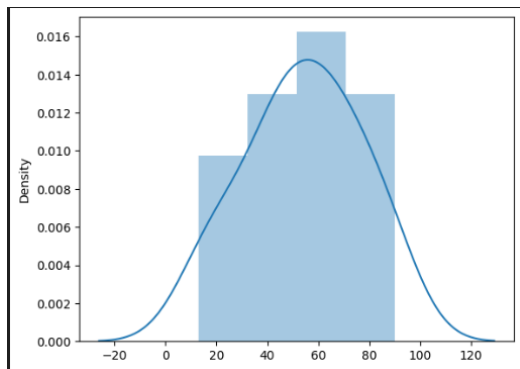
```
import seaborn as sns
```

```
%matplotlib inline
```

```
sns.displot(array)
```

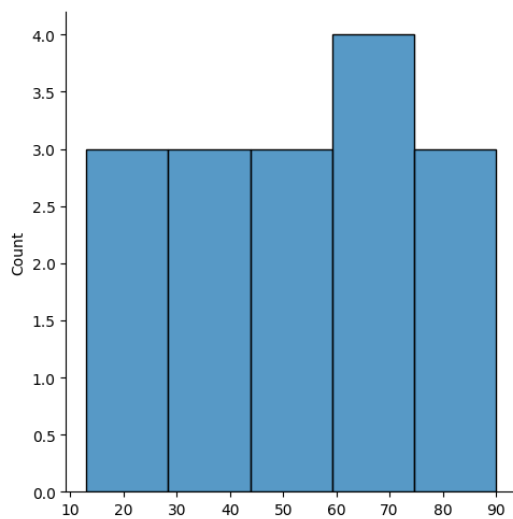
```
sns.distplot(array)
```



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

```
new_array
```

```
sns.distplot(new_array)
```



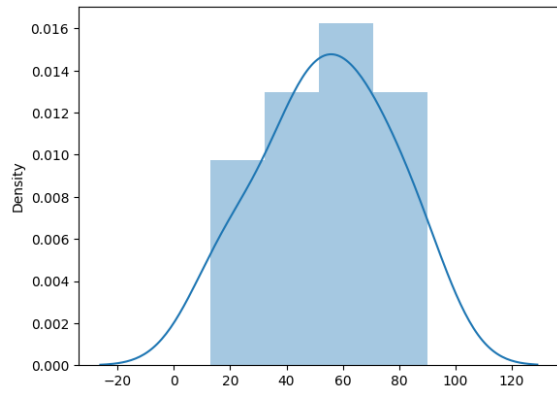
```
lr1,ur1=outDetection(new_array)
```

```
lr1,ur1
```

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

```
final_array
```

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



#3) Missing and inappropriate data

#230701004

#ABHINAYA LAKSHMI.S

#20.08.2024

import numpy as np

import pandas as pd

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

df

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

df.duplicated()

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

df.info()

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

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

```
df
```

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

```
len(df)
```

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

```
df.set_index(index,inplace=True)
```

```
index
```

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

```
df
```

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

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

```
df
```

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

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

```
df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan
```

```
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4.0	Ibis	veg	1300.0	2	40000.0
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3	59000.0
2	3.0	25-30	NaN	RedFox	Veg	1322.0	2	30000.0
3	4.0	20-25	NaN	LemonTree	Veg	1234.0	2	120000.0
4	5.0	35+	3.0	Ibis	Vegetarian	989.0	2	45000.0
5	6.0	35+	3.0	Ibys	Non-Veg	1909.0	2	122220.0
6	7.0	35+	4.0	RedFox	Vegetarian	1000.0	-1	21122.0
7	8.0	20-25	NaN	LemonTree	Veg	2999.0	-10	345673.0
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3	NaN
9	10.0	30-35	5.0	RedFox	non-Veg	NaN	4	87777.0

```
df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
```

```
df
```

```
df.Age_Group.unique()
```

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

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

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

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

```
df.FoodPreference.unique
```

```
<bound method Series.unique of 0          veg
1      Non-Veg
2        Veg
3        Veg
4    Vegetarian
5      Non-Veg
6    Vegetarian
7        Veg
8      Non-Veg
9      non-Veg
Name: FoodPreference, dtype: object>
```

```
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)
```

```
df
```

CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
0	1.0	20-25	4.0	Ibis	Veg	1300.0	2.0	40000.0
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	4.0	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	4.0	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3.0	Ibis	Veg	989.0	2.0	45000.0
5	6.0	35+	3.0	Ibis	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4.0	RedFox	Veg	1000.0	2.0	21122.0
7	8.0	20-25	4.0	LemonTree	Veg	2999.0	2.0	345673.0
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3.0	96755.0
9	10.0	30-35	5.0	RedFox	Non-Veg	1801.0	4.0	87777.0

#4)Data Preprocessing

#230701004

#ABHINAYA LAKSHMI.S

#27.08.2024

import numpy as np

import pandas as pd

df=pd.read_csv("/content/pre-process_datasample.csv")

df

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

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Country     9 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: 448.0+ bytes
```

df.Country.mode()

Country
0 France

df.Country.mode()[0]

type(df.Country.mode())

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

df.Age.fillna(df.Age.median(),inplace=True)

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

df

	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	France	50.0	83000.0	No

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

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

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 True False False 50.0 83000.0 No

9 True False False 37.0 67000.0 Yes

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Country     10 non-null    object
1   Age         10 non-null    float64
2   Salary      10 non-null    float64
3   Purchased   10 non-null    object
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```



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

```
updated_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	True	False	False	50.0	83000.0	0
9	True	False	False	37.0	67000.0	1

EDA-Quantitative and Qualitative plots - Experiments 1

#230701004

#ABHINAYA LAKSHMI.S

03.09.2024

```
import seaborn as sns
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

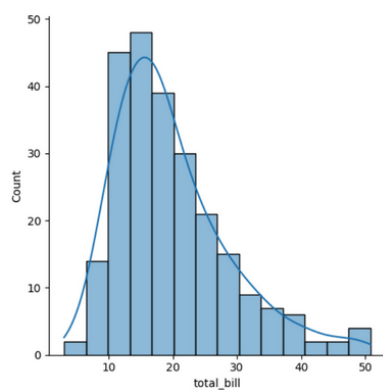
```
%matplotlib inline
```

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

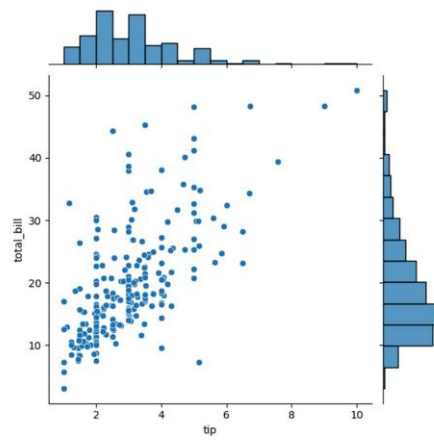
```
tips.head()
```

	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

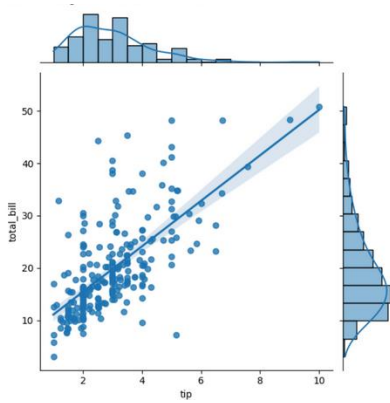
```
sns.displot(tips.total_bill,kde=True)
```



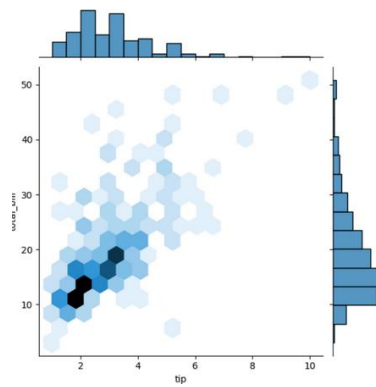
```
sns.jointplot(x=tips.tip,y=tips.total_bill)
```



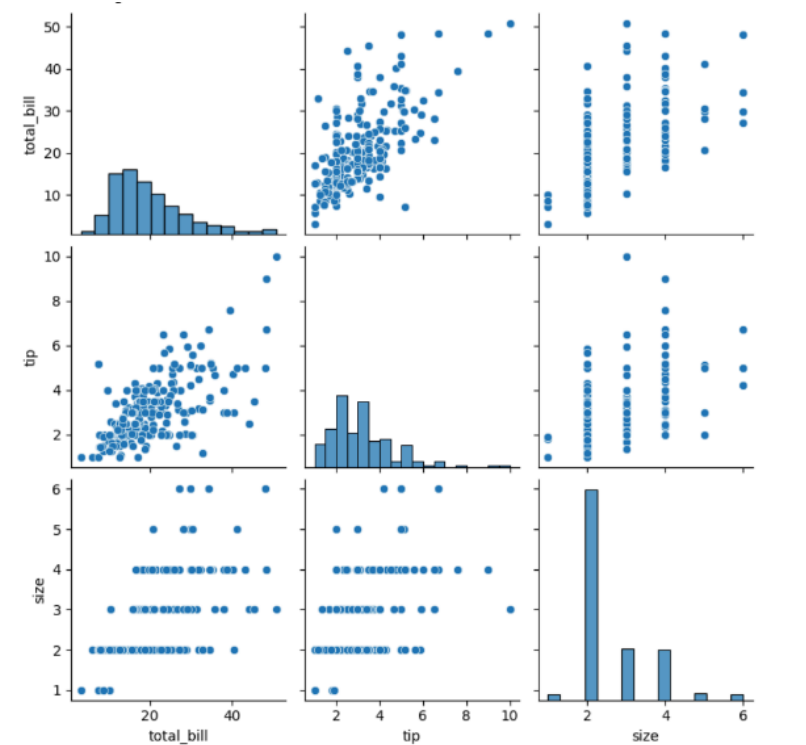
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
```



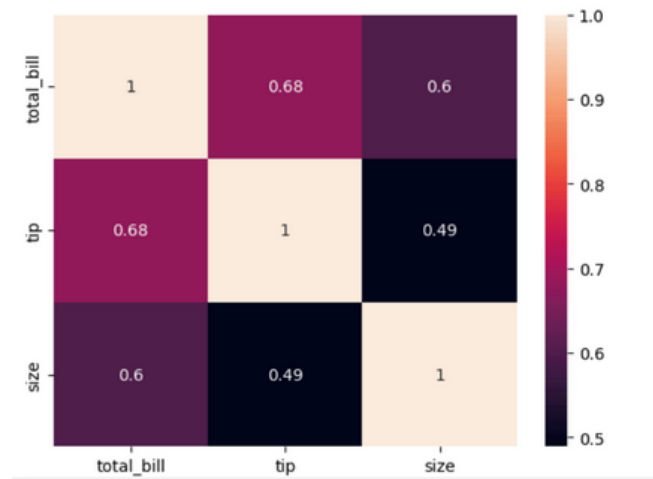
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
```



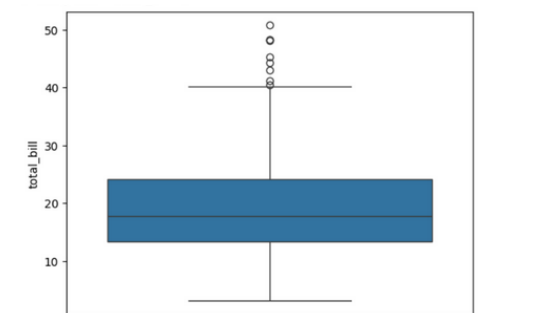
```
sns.pairplot(tips)
```



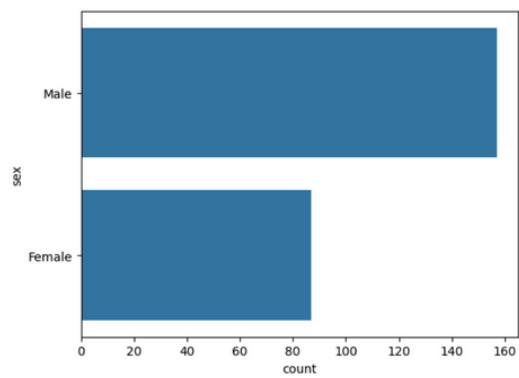
```
sns.heatmap(tips.corr(numeric_only=True),annot=True)
```



```
sns.boxplot(tips.total_bill)
```

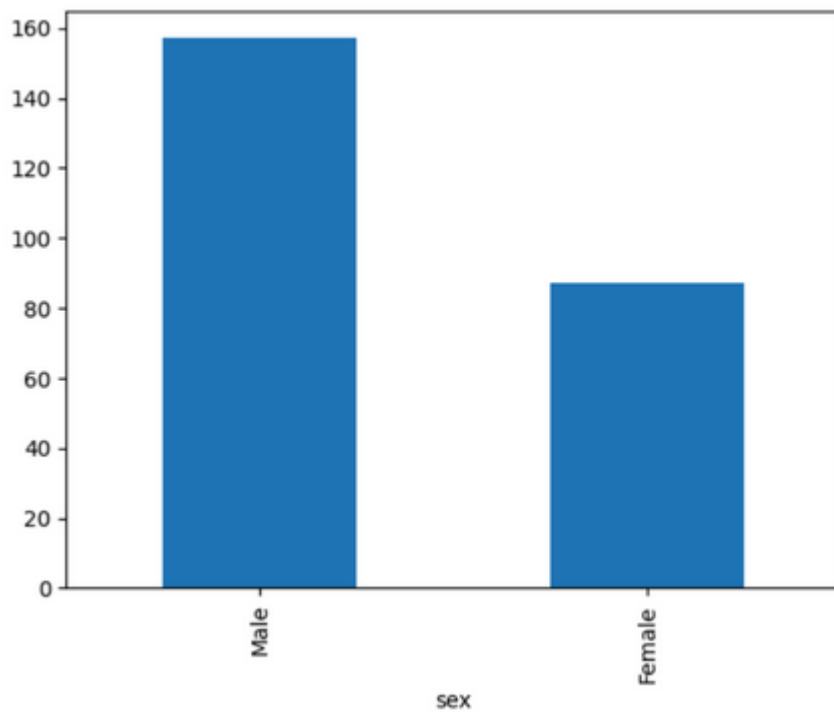
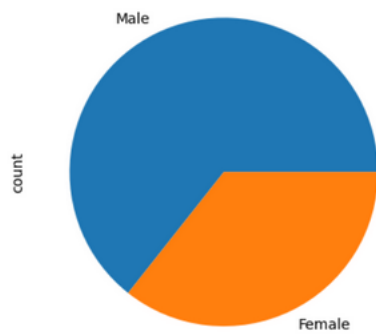


```
sns.countplot(tips.sex)
```



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

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



```
#Random Sampling and Sampling Distribution

#230701004

#ABHINAYA LAKSHMI.S

# 10.09.2024

import numpy as np
import matplotlib.pyplot as plt

population_mean = 50
population_std = 10
population_size = 100000
population = np.random.normal(population_mean, population_std, population_size)

plt.figure(figsize=(8, 5))

plt.hist(population, bins=50, color='skyblue', edgecolor='black', alpha=0.7)

plt.title('Population Distribution')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.axvline(population_mean, color='red', linestyle='dashed', linewidth=1.5, label='Population Mean')

plt.legend()

plt.show()

sample_sizes = [30, 50, 100]

num_samples = 1000

sample_means = {}

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, color='orange', edgecolor='black',
             label=f'Sample Size {size}')
```

```
plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5, label='Population Mean')
```

```
plt.title(f'Sampling Distribution of the Sample Mean (Sample Size {size})')
```

```
plt.xlabel('Sample Mean')
```

```
plt.ylabel('Frequency')
```

```
plt.legend()
```

```
plt.tight_layout()
```

```
plt.show()
```

```
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, color='purple', edgecolor='black',
```

```
label=f'Sample Size {size}', density=True)
```

```
plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5, label='Population Mean')
```

```
plt.title(f'Sampling Distribution (Sample Size {size}) - CLT Demonstration')
```

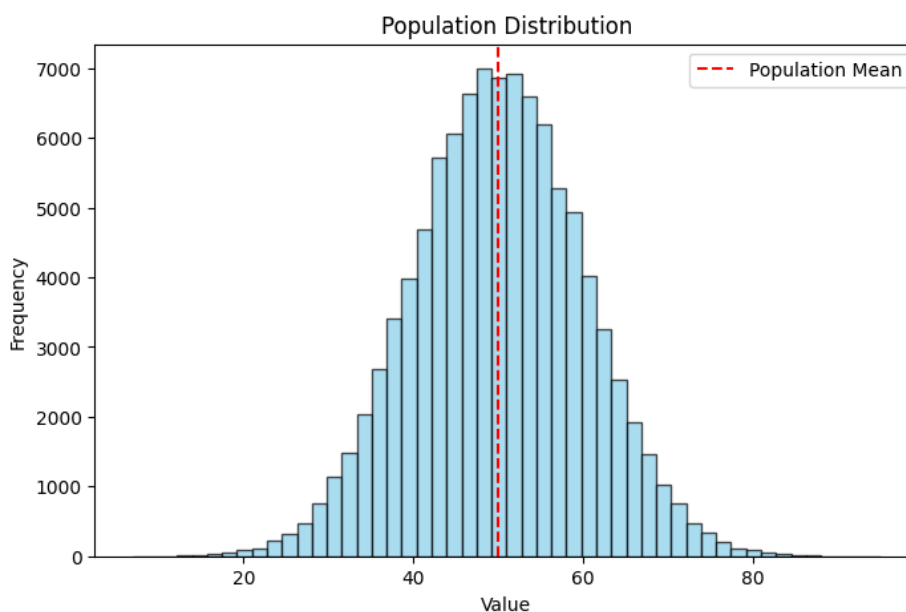
```
plt.xlabel('Sample Mean')
```

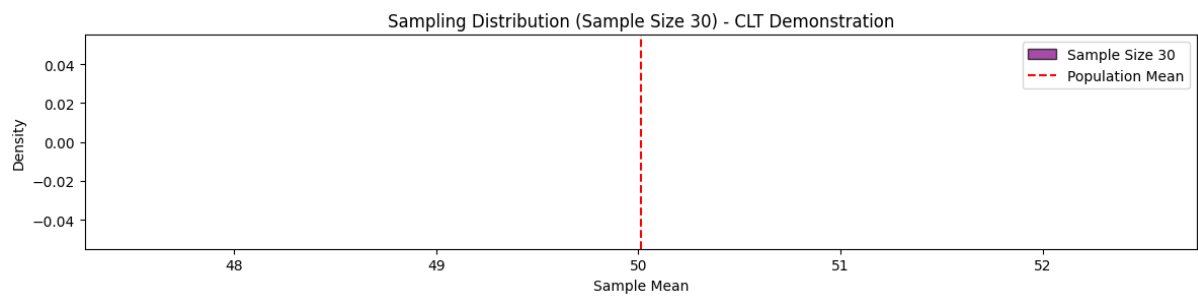
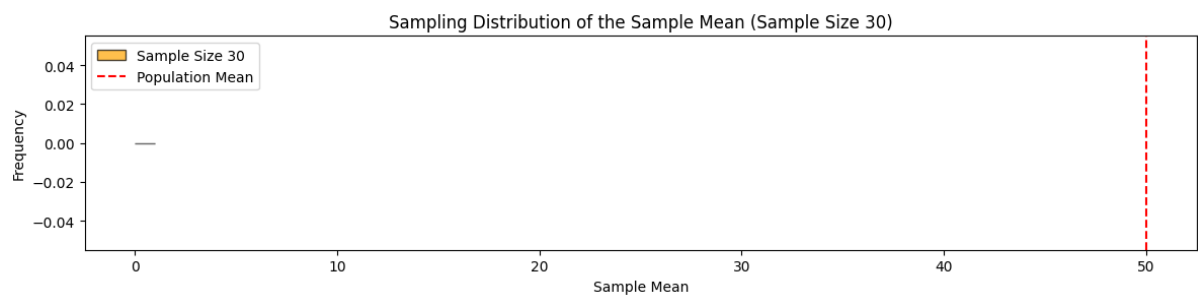
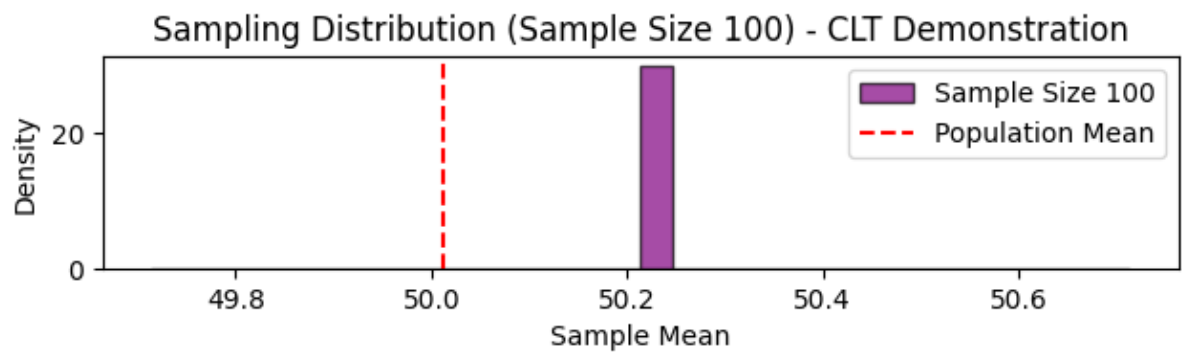
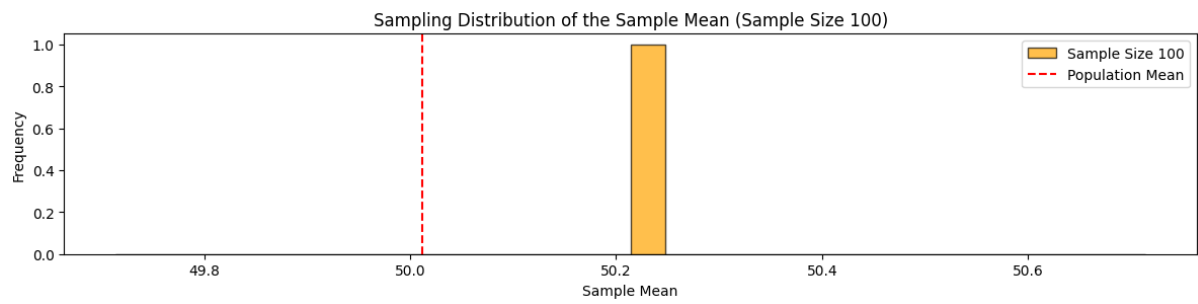
```
plt.ylabel('Density')
```

```
plt.legend()
```

```
plt.tight_layout()
```

```
plt.show()
```






```
# Z-Test
# 230701004
# ABHINAYA LAKSHMI.S
# 10.09.2024
```

```
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 = len(sample_data)
z_statistic = (sample_mean - population_mean) / (sample_std / np.sqrt(n))
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 different from 150 grams.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in average weight from 150 grams.")
```

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

```

# T-Test

# 230701004

# ABHINAYA LAKSHMI.S

# 08.10.2024


import numpy as np
import scipy.stats as stats

np.random.seed(42)

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 different from 100.")

else:

    print("Fail to reject the null hypothesis: There is no significant difference in average IQ score from 100.")

```

```

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

```

```
# Anova TEST
# 230701004
# ABHINAYA LAKSHMI.S
# 08.10.2024
```

```
import numpy as np
import scipy.stats as stats
np.random.seed(42)
n_plants = 25
growth_A = np.random.normal(loc=10, scale=2, size=n_plants)
growth_B = np.random.normal(loc=12, scale=3, size=n_plants)
growth_C = np.random.normal(loc=15, scale=2.5, size=n_plants)
f_statistic, p_value = stats.f_oneway(growth_A, growth_B, growth_C)
print("Treatment A Mean Growth:", np.mean(growth_A))
print("Treatment B Mean Growth:", np.mean(growth_B))
print("Treatment C Mean Growth:", np.mean(growth_C))
print()
print(f"F-Statistic: {f_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in mean growth rates among the
    three
    treatments.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in mean growth rates
    among the
    three treatments.")
if p_value < alpha:
    all_data = np.concatenate([growth_A, growth_B, growth_C])
    treatment_labels = ['A'] * n_plants + ['B'] * n_plants + ['C'] * n_plants
    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 rates among the three treatments.
```

```
# Feature Scaling
```

```
# 230701004
```

```
# ABHINAYA LAKSHMI.S
```

```
# 22.10.2024
```

```
import numpy as np
```

```
import pandas as pd
```

```
df = pd.read_csv('/content/pre-process_datasample.csv')
```

```
df
```

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

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

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

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

```
from sklearn.impute import SimpleImputer
```

```
age_imputer = SimpleImputer(strategy="mean")
```

```
salary_imputer = SimpleImputer(strategy="mean")
```

```
age_imputer.fit(features[:, [1]])
```

```
salary_imputer.fit(features[:, [2]])
```

```
features[:, [1]] = age_imputer.transform(features[:, [1]])
```

```
features[:, [2]] = salary_imputer.transform(features[:, [2]])
```

```
print("Features after handling missing values:")
```

features

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

```
from sklearn.preprocessing import OneHotEncoder
```

```
oh = OneHotEncoder(sparse_output=False)
```

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

```
print("OneHotEncoded 'Country' column:")
```

Country

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

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

```
print("Final dataset with OneHotEncoded 'Country' and other features:")
```

final_set

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

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```

sc.fit(final_set)

feat_standard_scaler = sc.transform(final_set)

print("Standardized features:")

feat_standard_scaler

array([[ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        7.58874362e-01, 7.49473254e-01],
       [-1.00000000e+00, -5.00000000e-01, 1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-1.00000000e+00, 2.00000000e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-1.00000000e+00, -5.00000000e-01, 1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-1.00000000e+00, 2.00000000e+00, -6.54653671e-01,
        1.77608893e-01, 6.63219199e-16],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-1.00000000e+00, -5.00000000e-01, 1.52752523e+00,
        0.00000000e+00, -1.07356980e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        1.34013983e+00, 1.38753832e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        1.63077256e+00, 1.75214693e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        -2.58340208e-01, 2.93712492e-01]])

```

```

from sklearn.preprocessing import MinMaxScaler

mms = MinMaxScaler(feature_range=(0, 1))

mms.fit(final_set)

feat_minmax_scaler = mms.transform(final_set)

print("Normalized features:")

print(feat_minmax_scaler)

```

```

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

```

```
# Linear Regression
```

```
# 230701004
```

```
# ABHINAYA LAKSHMI.S
```

```
# 29.10.2024
```

```
import numpy as np
```

```
import pandas as pd
```

```
df=pd.read_csv('Salary_data.csv')
```

```
df
```

```
df.info()
```

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

```
df.info()
```

```
df.describe()
```

```
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
```

```
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=23)
```

```
from sklearn.linear_model import LinearRegression
```

```
model=LinearRegression()
```

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

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

```
model.score(x_test,y_test)
```

```
model.coef_
```

```
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("Estimated Salary for {} years of experience is {}: " .format(yr_of_exp,Salary))
```



```
# Logistic Regression
```

```
# 230701004
```

```
#ABHINAYA LAKSHMI.S
```

```
# 05.11.2024
```

```
import numpy as np
```

```
import pandas as pd
```

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

```
df
```

```
User ID Gender Age EstimatedSalary Purchased 0 15624510
```

```
Male 19 19000 0 1 15810944 Male 35 20000 0 2 15668575
```

```
Female 26 43000 0 3 15603246 Female 27 57000 0 4 15804002
```

```
Male 19 76000 0 ... ..
```

```
395 15691863 Female 46 41000 1 396 15706071 Male 51 23000
```

```
1 397 15654296 Female 50 20000 1 398 15755018 Male 36
```

```
33000 0 399 15594041 Female 49 36000 1
```

```
400 rows × 5 columns
```

```
df.head()
```

```
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
```

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

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

```
features
```

```
label
```

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

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

for i in range(1,401):
    x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,random_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,i))
        x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,random_state=314)
        finalModel=LogisticRegression()
        finalModel.fit(x_train,y_train)
        print(finalModel.score(x_train,y_train))
        print(finalModel.score(x_test,y_test))
    from sklearn.metrics import classification_report
    print(classification_report(label,finalModel.predict(features)))
```

```
# K-MEANS CLUSTERING
```

```
# 230701004
```

```
#ABHINAYA LAKSHMI.S
```

```
# 05.11.2024
```

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
%matplotlib inline
```

```
df=pd.read_csv('Mall_Customers.csv')
```

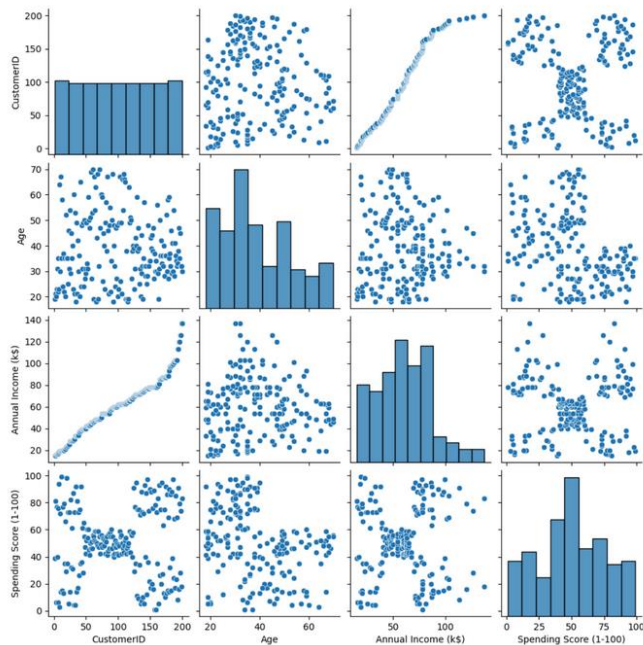
```
df.info()
```

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

```
df.head()
```

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

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



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

```
from sklearn.cluster import KMeans
```

```
model=KMeans(n_clusters=5)
```

```
model.fit(features)
```

```
KMeans(n_clusters=5)
```

```
KMeans(n_clusters=5)
```

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

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

```
Final=df.iloc[:,[3,4]]
```

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

```
Final.head()
```

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

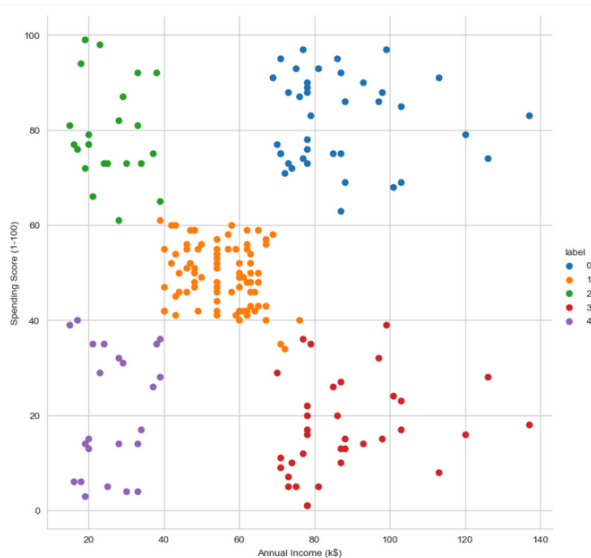
```
sns.set_style("whitegrid")
```

```
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
```

```
from sklearn.cluster import KMeans
```

```
wcss=[]
```

```
for i in range(1,10):
```

```
    model=KMeans(n_clusters=i)
```

```
    model.fit(features_el)
```

```
    wcss.append(model.inertia_)
```

```
plt.plot(range(1,10),wcss)
```

