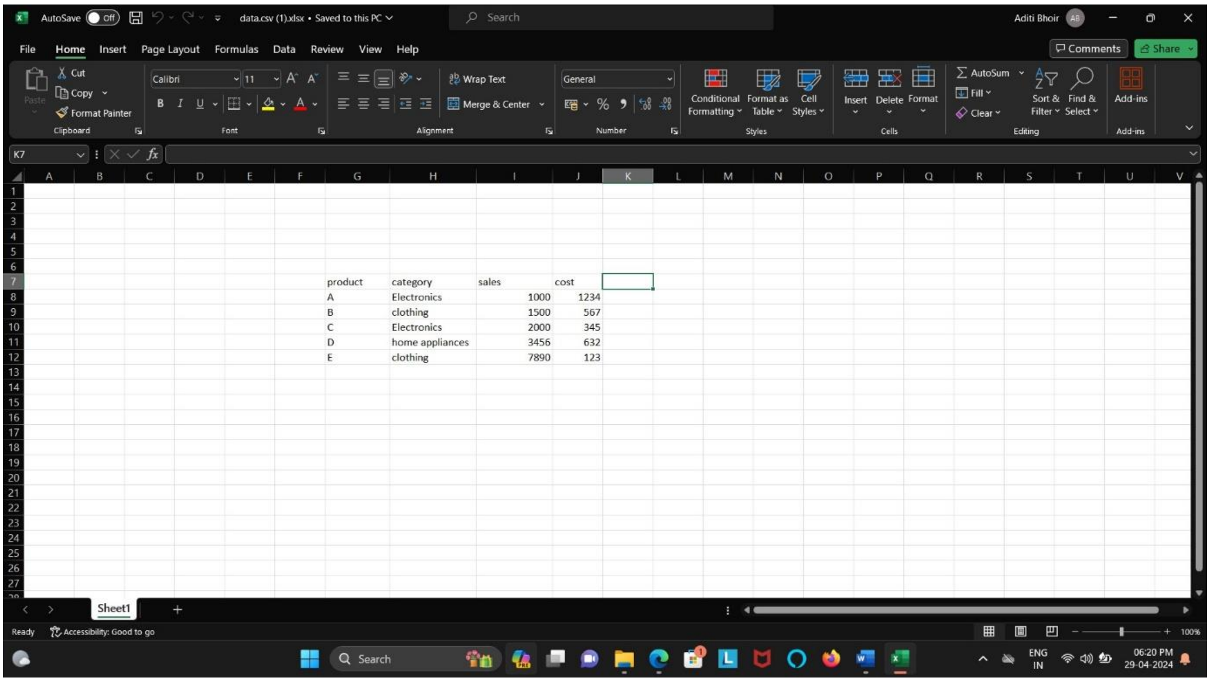
Data science practical

Practical no 1

Introduction to Excel

Open excel

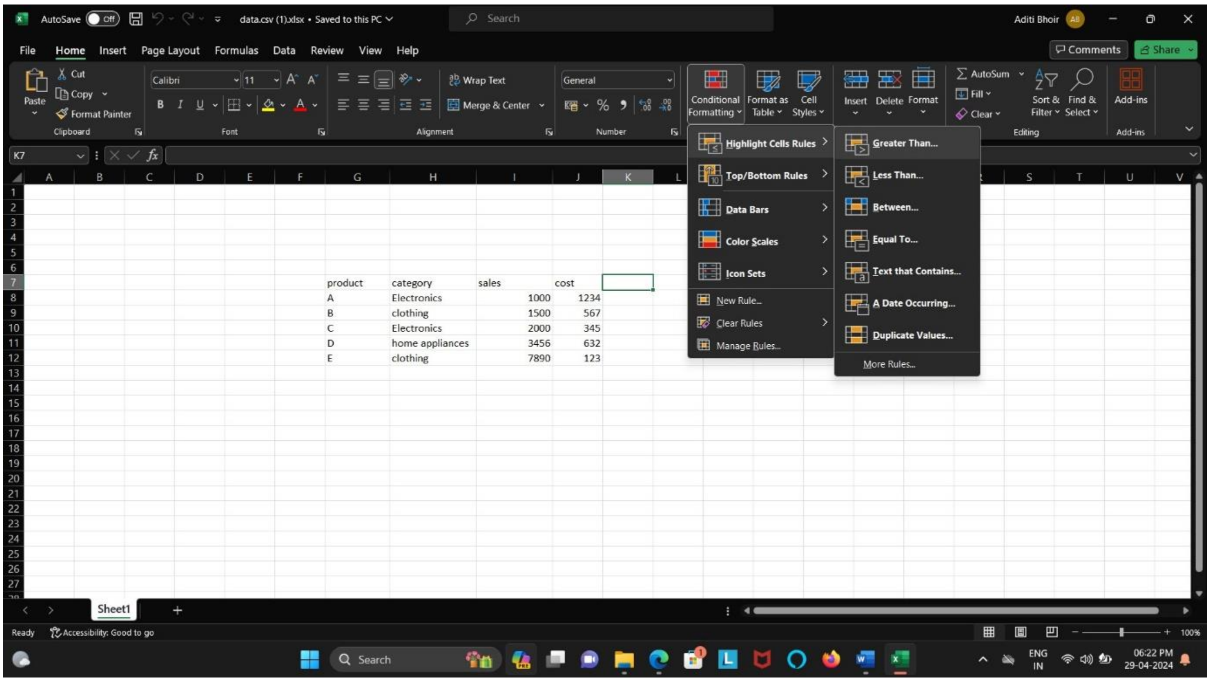
open dataset.csv



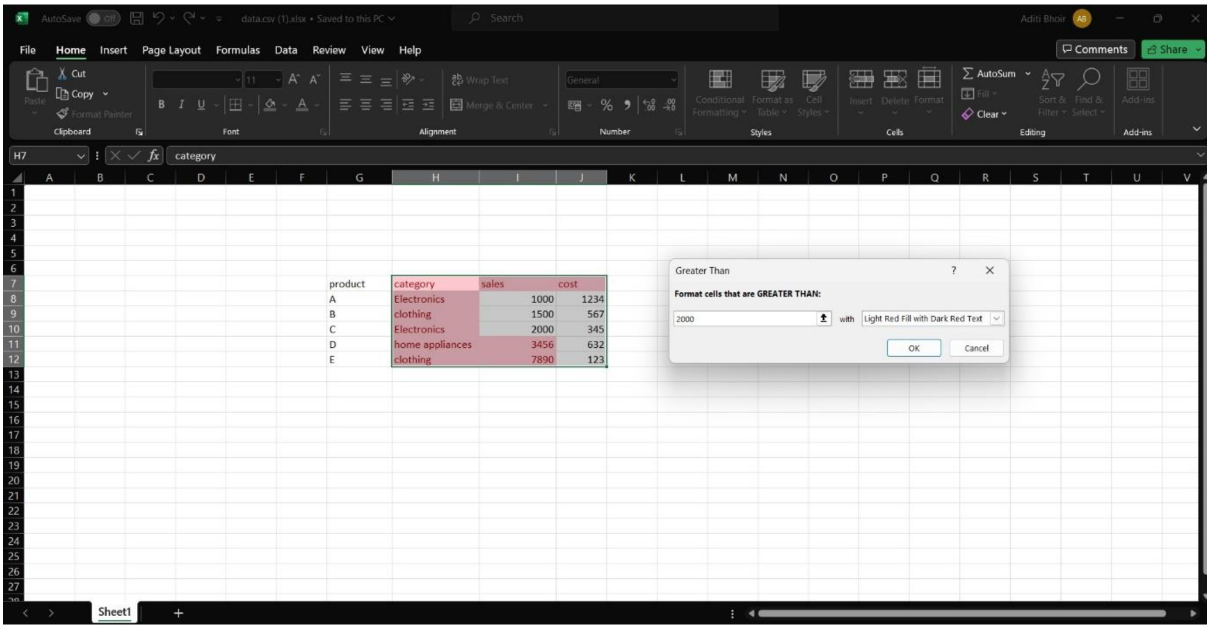
Select all

Then go to conditional formatting

Then select Greater than

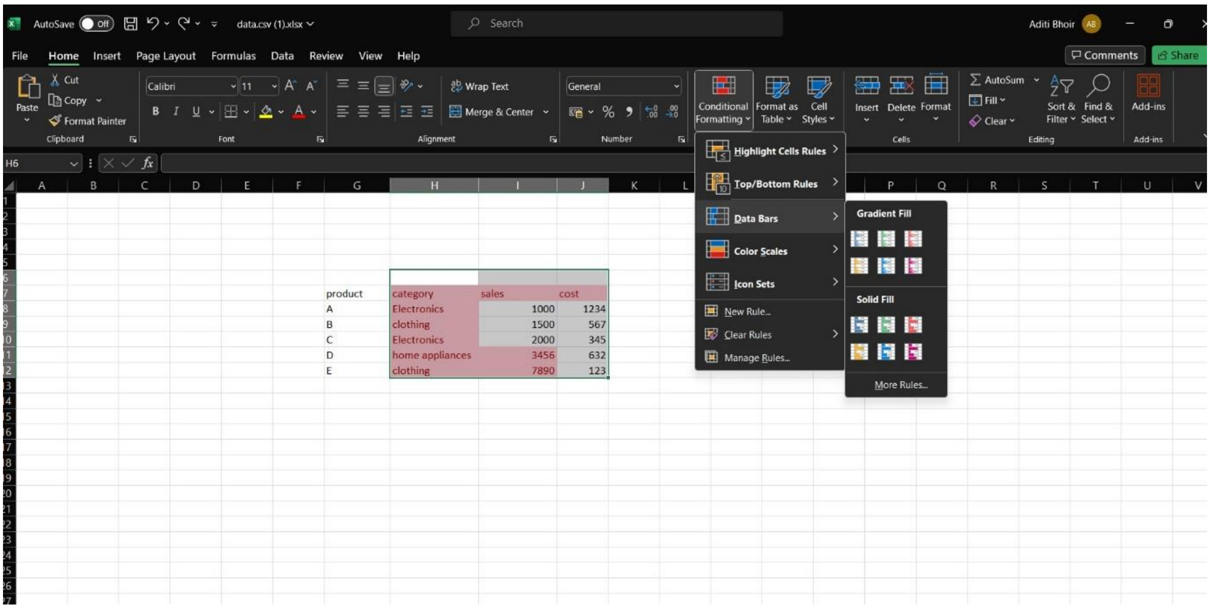


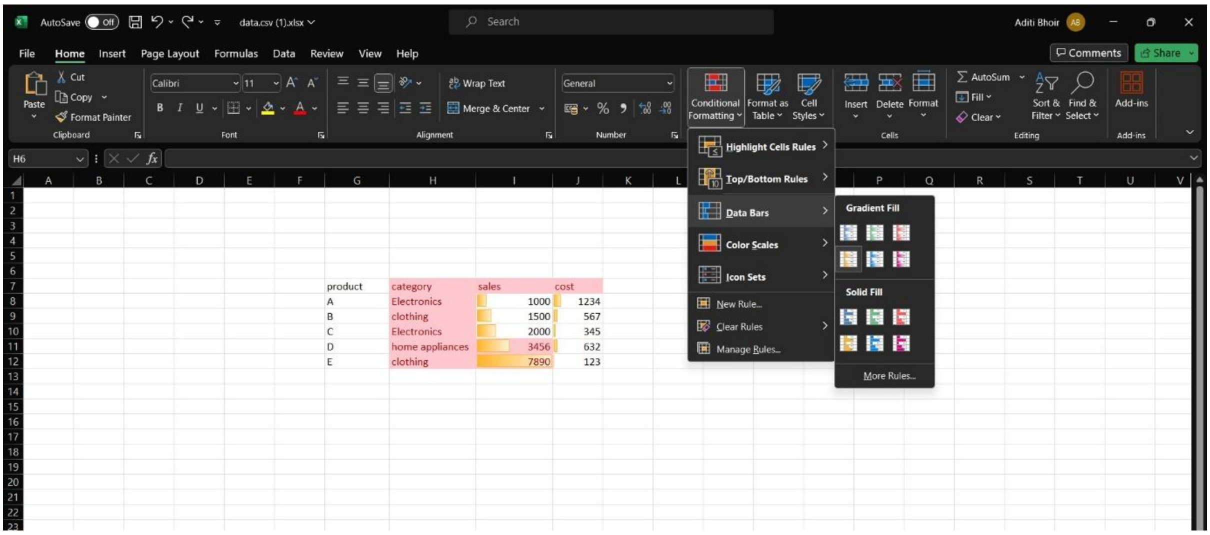
Enter the greater than filters value for example 2000



Go to data bars

Then select solid fill in conditional formatting

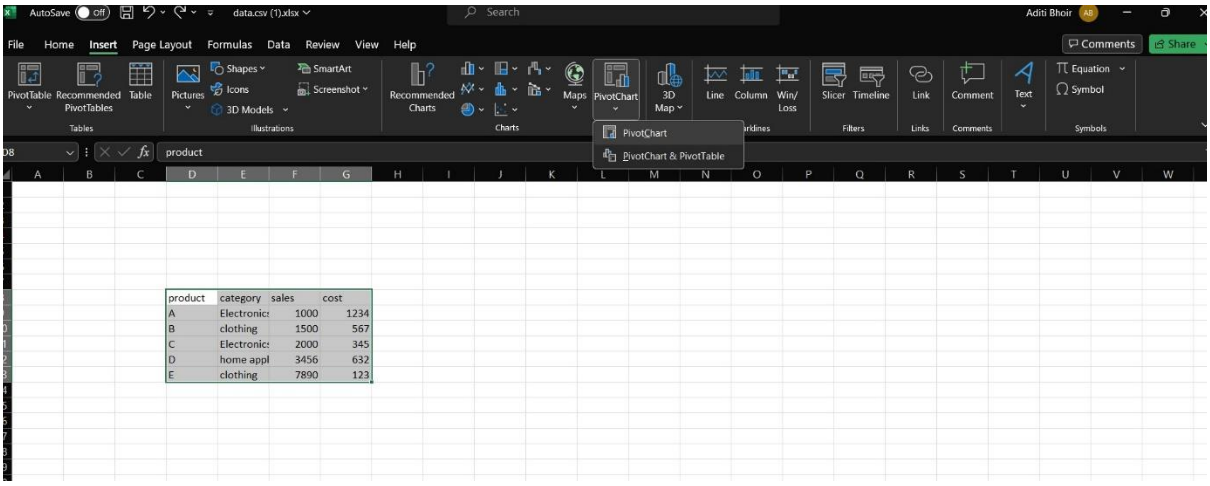


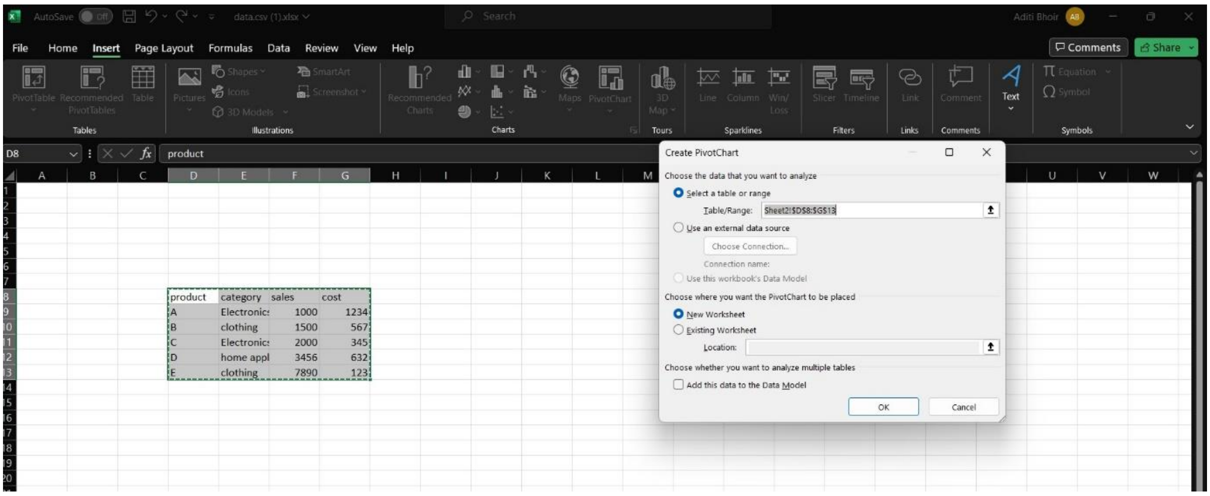


B. Create a pivot table to analyse and summarize data.

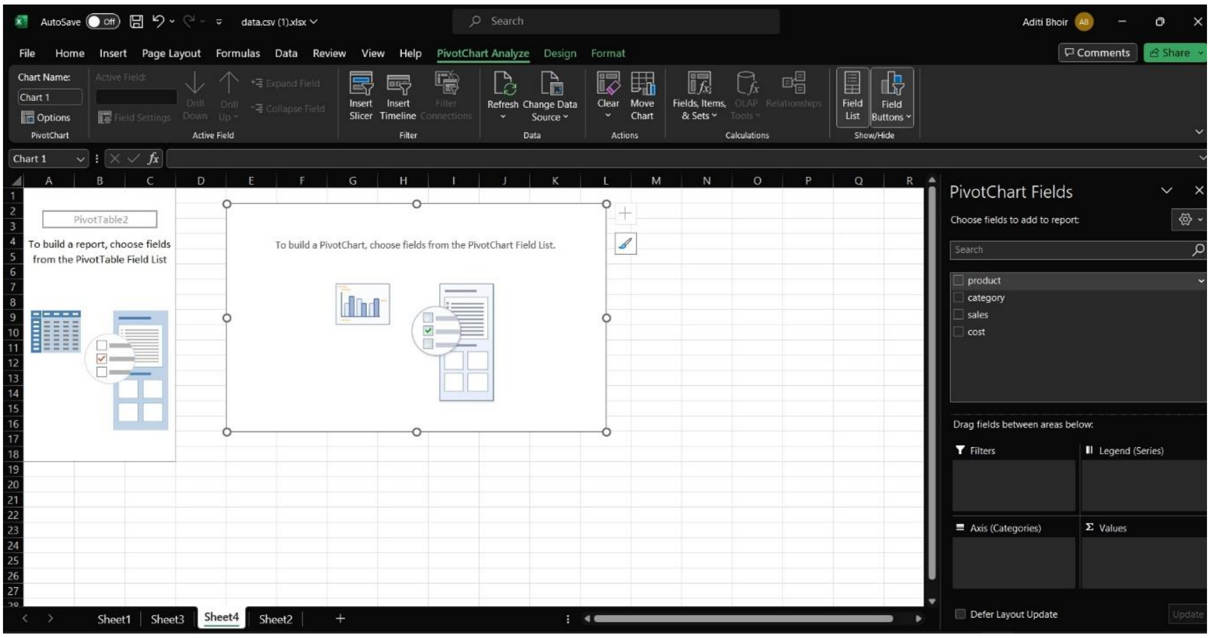
Select entire table and go to insert tab PivotChart

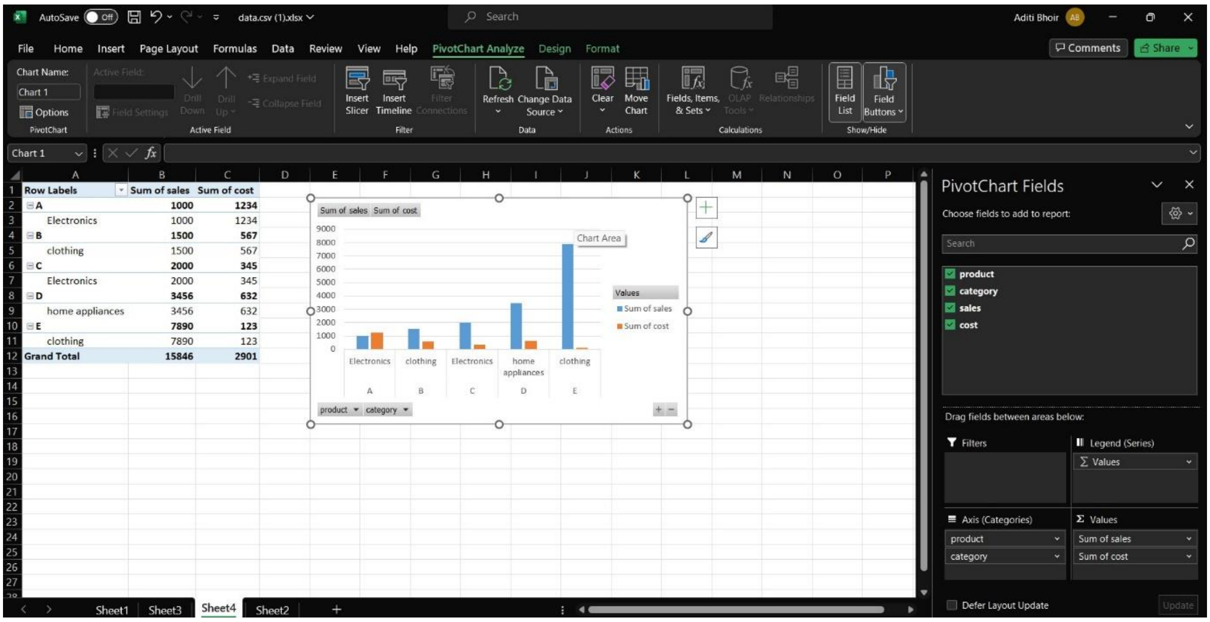
Then select New Workstation in the create pivot chart window





Then select and drag the attributes in the below boxes

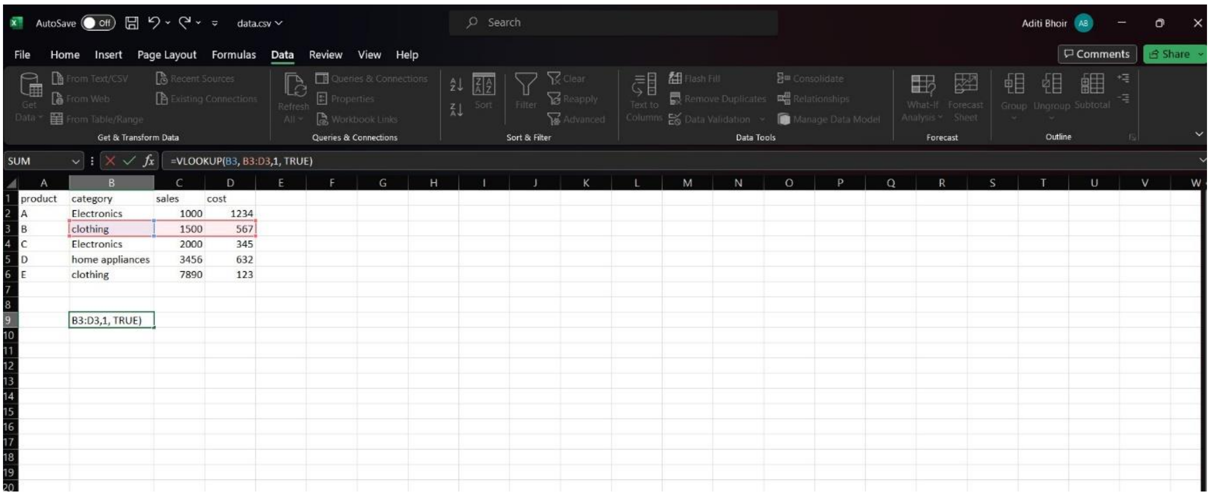




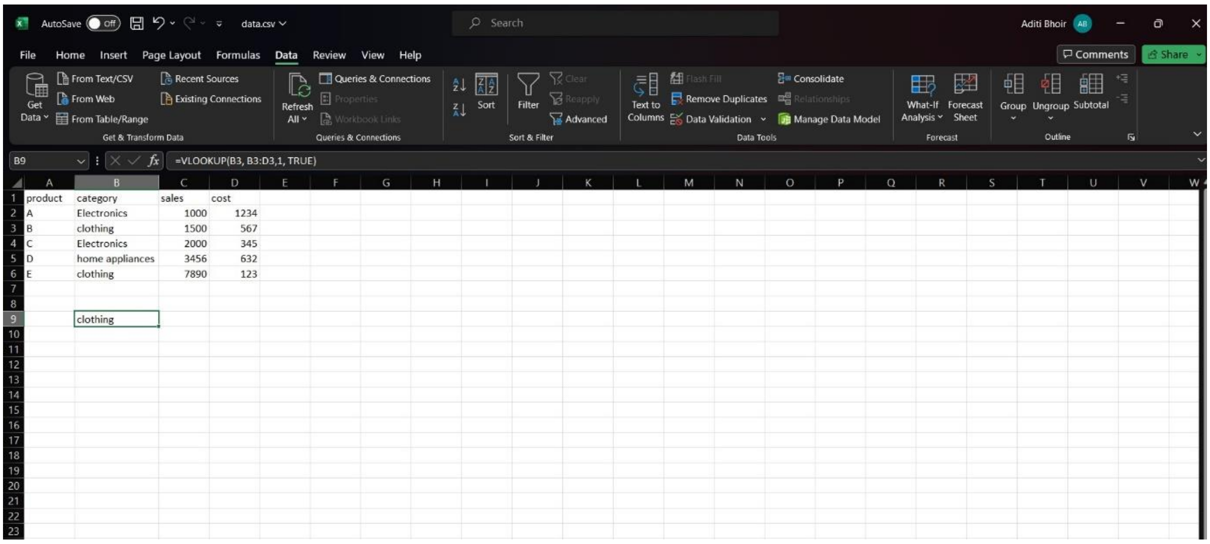
C. Use VLOOKUP function to retrieve information from a different worksheet or table

Step 1: click on an empty cell and type the following command in the command line

VLOOKUP(B3, B3:D3,1, TRUE)

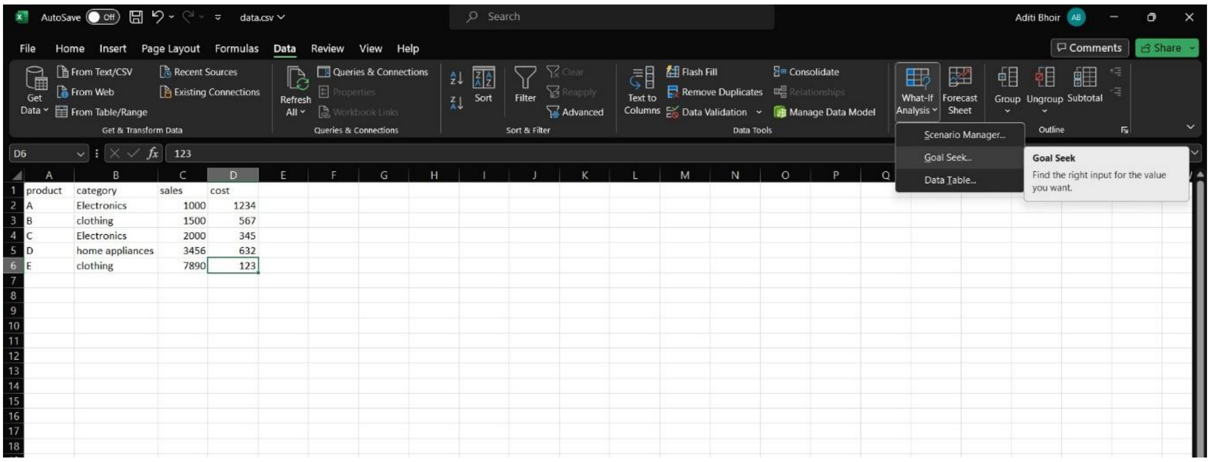


Click enter button

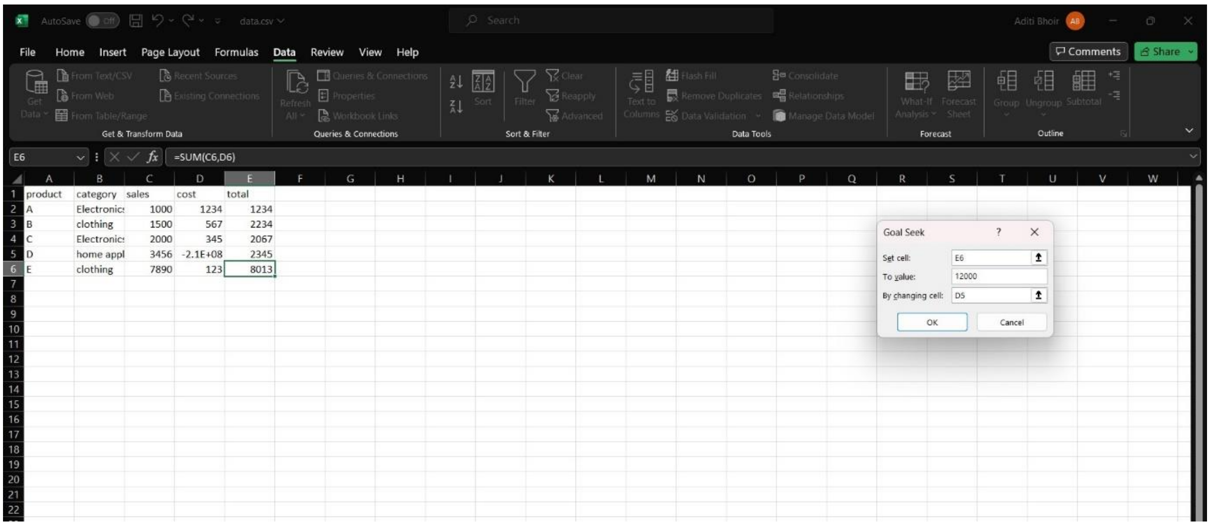


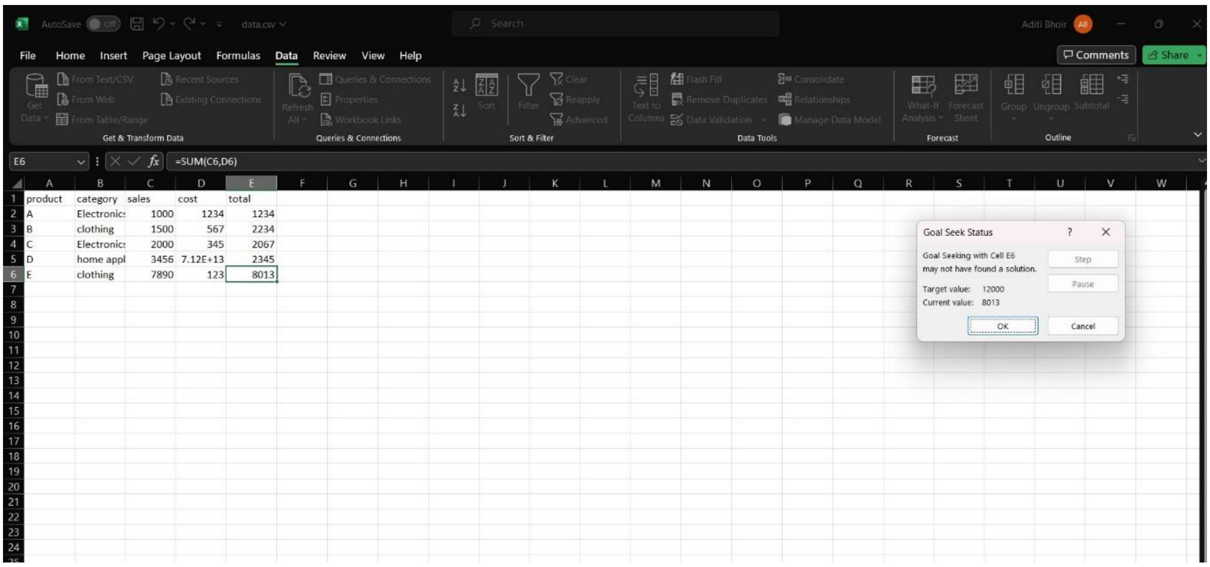
D. Perform what-if analysis using Goal Seek to determine input values for desired

Step 1: In the Data tab go to the what if analysis>Goal seek.



Step 2: Fill the information in the window accordingly and click ok.





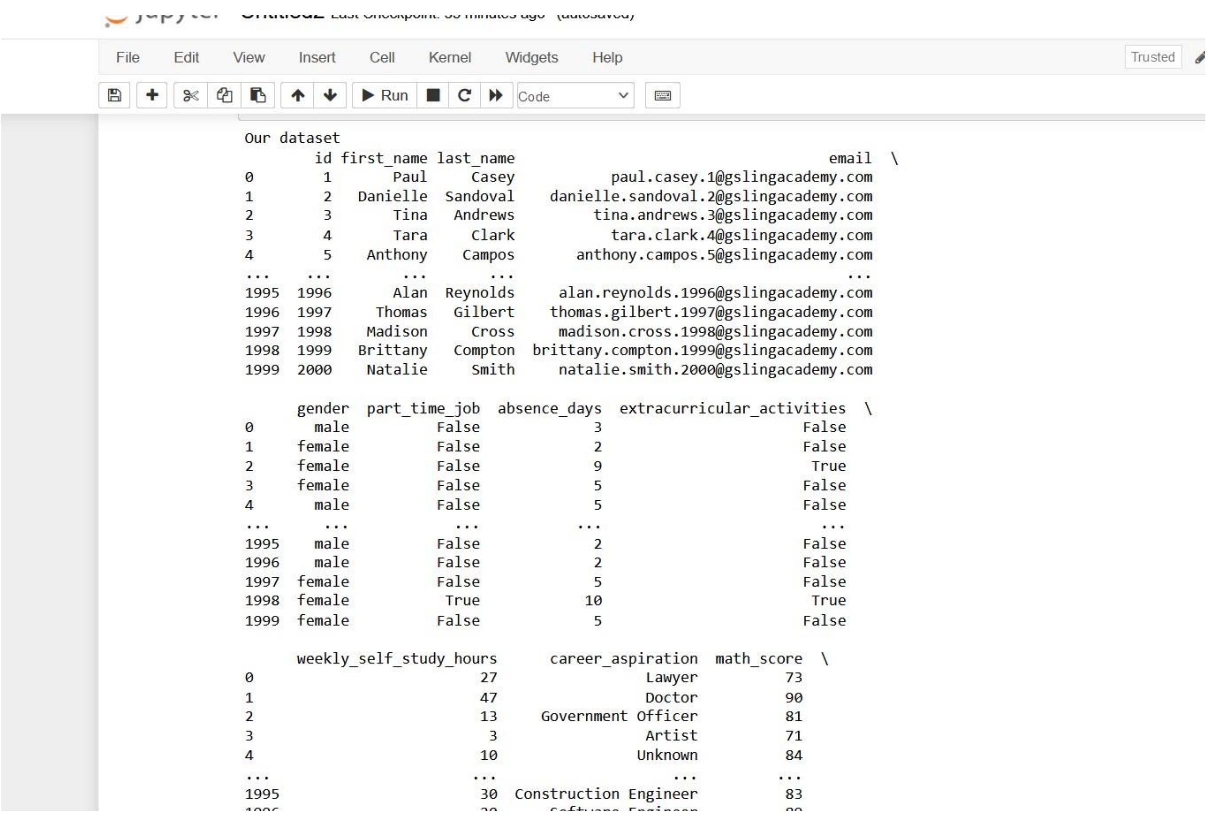
Practical no 2

Aim:- Data Frames and Basic Data Pre-processing

A. Read data from CSV and JSON files into a data frame

1) #Read data from a csv file

```
import pandas as pd
df = pd.read_csv('D:\data science\student-scores.csv')
print("Our dataset ")
print(df)
```



```
Our dataset
        id first_name last_name                                 email  \
0        1       Paul     Casey        paul.casey.1@gslingacademy.com
1        2   Danielle  Sandoval   danielle.sandoval.2@gslingacademy.com
2        3       Tina   Andrews      tina.andrews.3@gslingacademy.com
3        4       Tara     Clark        tara.clark.4@gslingacademy.com
4        5    Anthony    Campos     anthony.campos.5@gslingacademy.com
...    ...        ...       ...                                   ...
1995  1996       Alan  Reynolds    alan.reynolds.1996@gslingacademy.com
1996  1997     Thomas   Gilbert   thomas.gilbert.1997@gslingacademy.com
1997  1998    Madison     Cross    madison.cross.1998@gslingacademy.com
1998  1999   Brittany   Compton brittany.compton.1999@gslingacademy.com
1999  2000    Natalie     Smith    natalie.smith.2000@gslingacademy.com

      gender  part_time_job  absence_days  extracurricular_activities  \
0       male          False             3                       False
1     female          False             2                       False
2     female          False             9                        True
3     female          False             5                       False
4       male          False             5                       False
...      ...            ...           ...                         ...
1995    male          False             2                       False
1996    male          False             2                       False
1997  female          False             5                       False
1998  female           True            10                        True
1999  female          False             5                       False

      weekly_self_study_hours      career_aspiration  math_score  \
0                          27                 Lawyer          73
1                          47                 Doctor          90
2                          13     Government Officer          81
3                           3                 Artist          71
4                          10                Unknown          84
...                       ...                    ...         ...
1995                       30  Construction Engineer          83
```

jupyter Untitled2 Last Checkpoint: 36 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
1997                  14     Software Engineer         97
1998                   5        Business Owner         51
1999                  27            Accountant         82

      history_score  physics_score  chemistry_score  biology_score  \
0                81             93               97             63
1                86             96              100             90
2                97             95               96             65
3                74             88               80             89
4                77             65               65             80
...             ...            ...              ...            ...
1995             77             84               73             75
1996             65             73               80             87
1997             85             63               93             68
1998             96             72               89             95
1999             99             91               69             83

      english_score  geography_score
0                80               87
1                88               90
2                77               94
3                63               86
4                74               76
...             ...              ...
1995             84               82
1996             67               73
1997             94               78
1998             88               75
1999             93              100

[2000 rows x 17 columns]
```

2) **# Reading data from a JSON file**

import pandas as pd

data = pd.read_json(' D:\\data science\\ sales.json')

print(data)

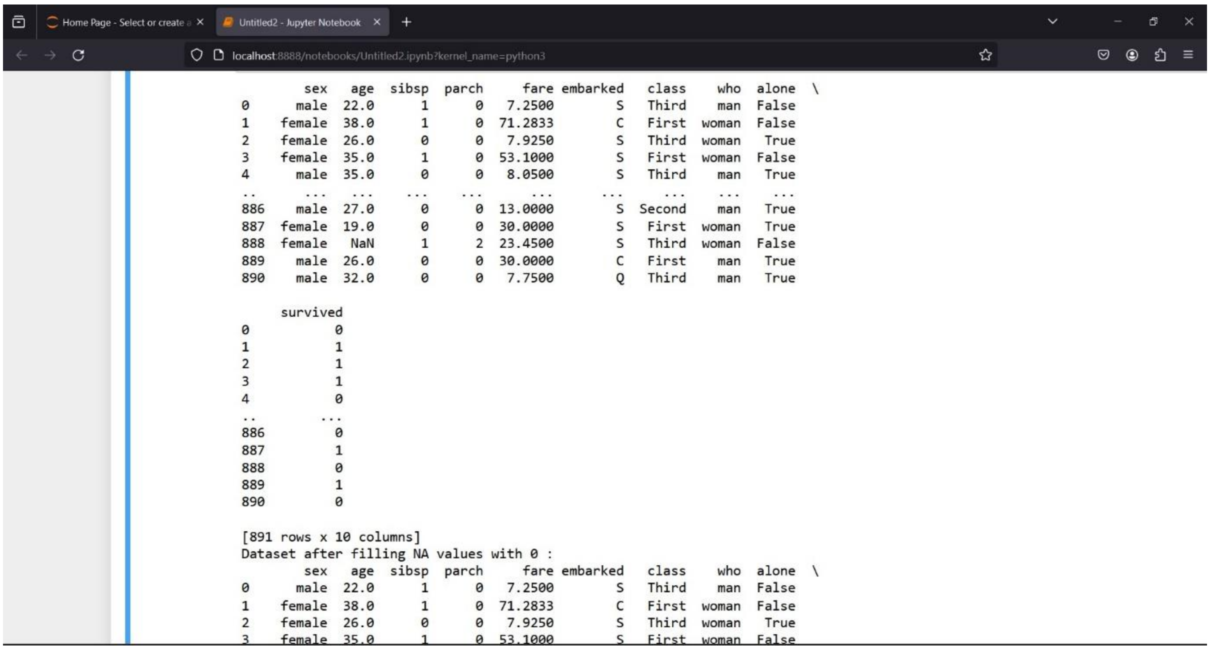
```
>>>
================ RESTART: D:/Notes/sem-6/data science/p:
    fruit    size   color
0   Apple   Large     Red
1  Banana  Medium  Yellow
2  Orange   Small  Orange
>>>
```

B. Perform basic data pre-processing tasks such as handling missing values and outliers.

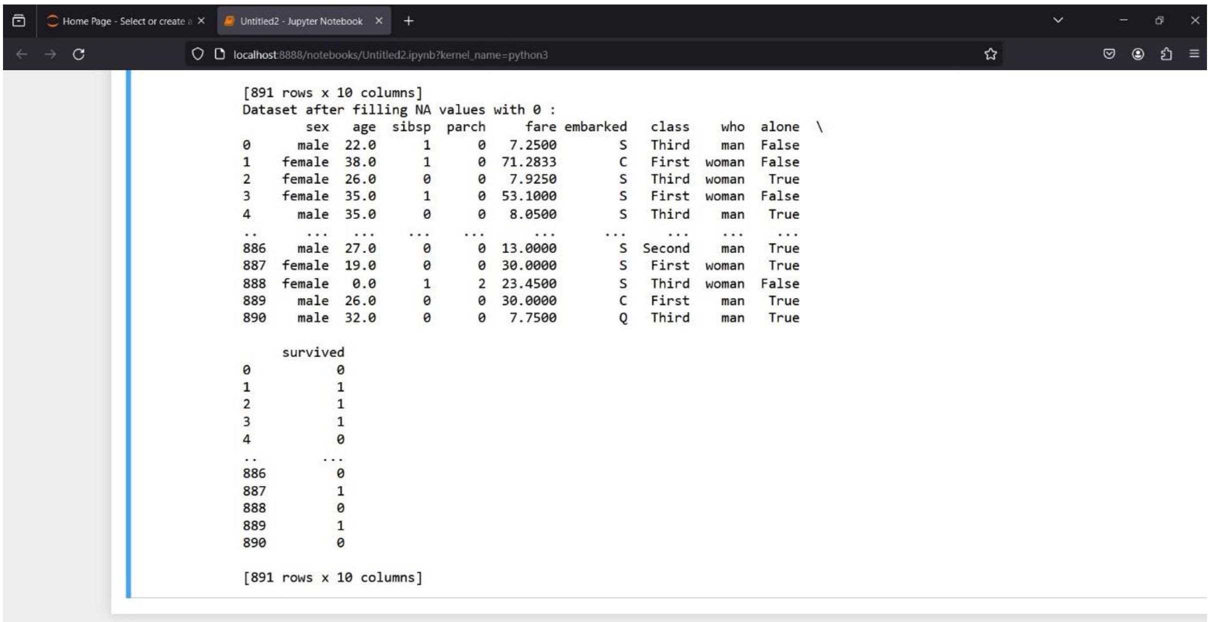
Code:

(1) # Replacing NA values using fillna()

```
import pandas as pd
df = pd.read_csv('titanic.csv')
print(df)
df.head(10)
print("Dataset after filling NA values with 0 : ")
df2=df.fillna(value=0)
print(df2)
```



```
        sex   age  sibsp  parch     fare embarked   class    who  alone  \
0      male  22.0      1      0   7.2500        S   Third    man  False
1    female  38.0      1      0  71.2833        C   First  woman  False
2    female  26.0      0      0   7.9250        S   Third  woman   True
3    female  35.0      1      0  53.1000        S   First  woman  False
4      male  35.0      0      0   8.0500        S   Third    man   True
..      ...   ...    ...    ...      ...      ...     ...    ...    ...
886    male  27.0      0      0  13.0000        S  Second    man   True
887  female  19.0      0      0  30.0000        S   First  woman   True
888  female   NaN      1      2  23.4500        S   Third  woman  False
889    male  26.0      0      0  30.0000        C   First    man   True
890    male  32.0      0      0   7.7500        Q   Third    man   True

     survived
0           0
1           1
2           1
3           1
4           0
..        ...
886         0
887         1
888         0
889         1
890         0

[891 rows x 10 columns]
Dataset after filling NA values with 0 :
        sex   age  sibsp  parch     fare embarked   class    who  alone  \
0      male  22.0      1      0   7.2500        S   Third    man  False
1    female  38.0      1      0  71.2833        C   First  woman  False
2    female  26.0      0      0   7.9250        S   Third  woman   True
3    female  35.0      1      0  53.1000        S   First  woman  False
```




```
[891 rows x 10 columns]
Dataset after filling NA values with 0 :
        sex   age  sibsp  parch     fare embarked   class    who  alone  \
0      male  22.0      1      0   7.2500        S   Third    man  False
1    female  38.0      1      0  71.2833        C   First  woman  False
2    female  26.0      0      0   7.9250        S   Third  woman   True
3    female  35.0      1      0  53.1000        S   First  woman  False
4      male  35.0      0      0   8.0500        S   Third    man   True
..      ...   ...    ...    ...      ...      ...     ...    ...    ...
886    male  27.0      0      0  13.0000        S  Second    man   True
887  female  19.0      0      0  30.0000        S   First  woman   True
888  female   0.0      1      2  23.4500        S   Third  woman  False
889    male  26.0      0      0  30.0000        C   First    man   True
890    male  32.0      0      0   7.7500        Q   Third    man   True

     survived
0           0
1           1
2           1
3           1
4           0
..        ...
886         0
887         1
888         0
889         1
890         0

[891 rows x 10 columns]
```

(2) **# Dropping NA values using dropna()**

```
import pandas as pd
df = pd.read_csv('titanic.csv')
print(df)
df.head(10)
print("Dataset after dropping NA values: ")
df.dropna(inplace = True)
print(df)
```

Home Page - Select or create a | Untitled2 - Jupyter Notebook

localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3

```
        sex   age  sibsp  parch     fare embarked   class    who  alone  \
0      male  22.0      1      0   7.2500        S   Third    man  False
1    female  38.0      1      0  71.2833        C   First  woman  False
2    female  26.0      0      0   7.9250        S   Third  woman   True
3    female  35.0      1      0  53.1000        S   First  woman  False
4      male  35.0      0      0   8.0500        S   Third    man   True
..      ...   ...    ...    ...      ...      ...     ...    ...    ...
886    male  27.0      0      0  13.0000        S  Second    man   True
887  female  19.0      0      0  30.0000        S   First  woman   True
888  female   NaN      1      2  23.4500        S   Third  woman  False
889    male  26.0      0      0  30.0000        C   First    man   True
890    male  32.0      0      0   7.7500        Q   Third    man   True

     survived
0           0
1           1
2           1
3           1
4           0
..        ...
886         0
887         1
888         0
889         1
890         0

[891 rows x 10 columns]
Dataset after dropping NA values:
        sex   age  sibsp  parch     fare embarked   class    who  alone  \
0      male  22.0      1      0   7.2500        S   Third    man  False
1    female  38.0      1      0  71.2833        C   First  woman  False
2    female  26.0      0      0   7.9250        S   Third  woman   True
3    female  35.0      1      0  53.1000        S   First  woman  False
```

Home Page - Select or create a | Untitled2 - Jupyter Notebook

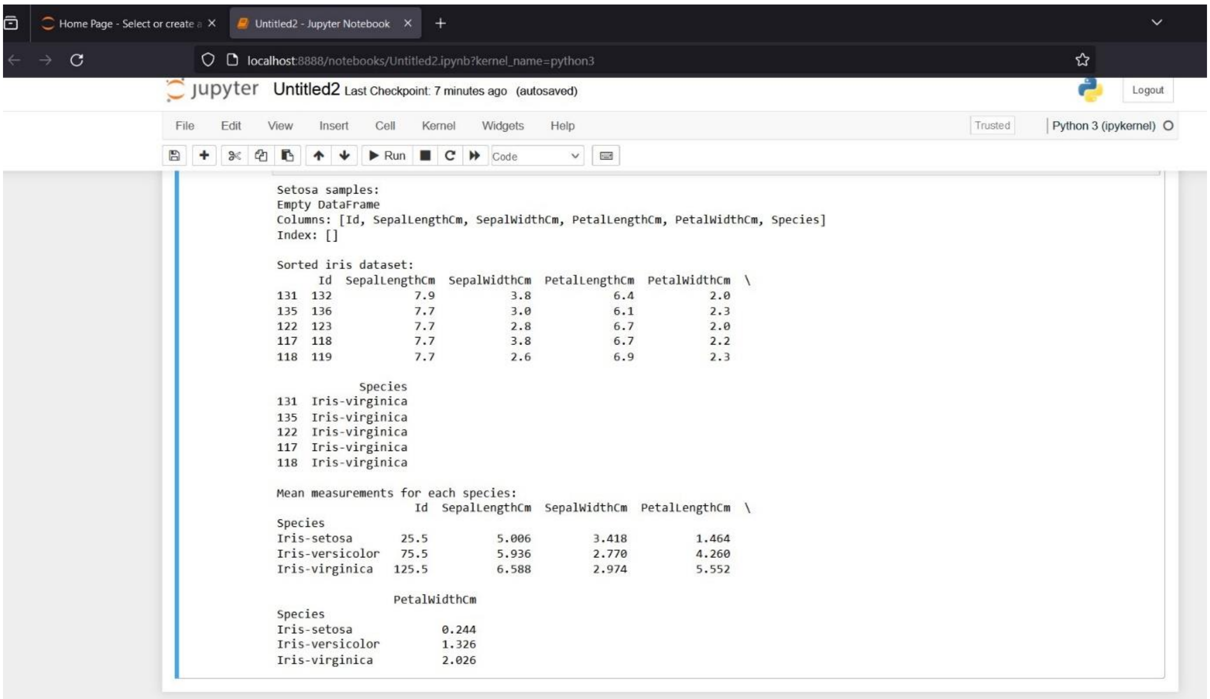
localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3

```
[891 rows x 10 columns]
Dataset after dropping NA values:
        sex   age  sibsp  parch     fare embarked   class    who  alone  \
0      male  22.0      1      0   7.2500        S   Third    man  False
1    female  38.0      1      0  71.2833        C   First  woman  False
2    female  26.0      0      0   7.9250        S   Third  woman   True
3    female  35.0      1      0  53.1000        S   First  woman  False
4      male  35.0      0      0   8.0500        S   Third    man   True
..      ...   ...    ...    ...      ...      ...     ...    ...    ...
885  female  39.0      0      5  29.1250        Q   Third  woman  False
886    male  27.0      0      0  13.0000        S  Second    man   True
887  female  19.0      0      0  30.0000        S   First  woman   True
889    male  26.0      0      0  30.0000        C   First    man   True
890    male  32.0      0      0   7.7500        Q   Third    man   True

     survived
0           0
1           1
2           1
3           1
4           0
..        ...
885         0
886         0
887         1
889         1
890         0

[712 rows x 10 columns]
```

C. Manipulate and transform data using functions like filtering, sorting, and grouping

Code:

```
import pandas as pd
# Load iris dataset
iris = pd.read_csv('Iris.csv')

# Filtering data based on a condition
setosa = iris[iris['Species'] == 'setosa']
print("Setosa samples:")
print(setosa.head())

# Sorting data
sorted_iris = iris.sort_values(by='SepalLengthCm', ascending=False)
print("\nSorted iris dataset:")
print(sorted_iris.head())

# Grouping data
grouped_species = iris.groupby('Species').mean()
print("\nMean measurements for each species:")
print(grouped_species)
```



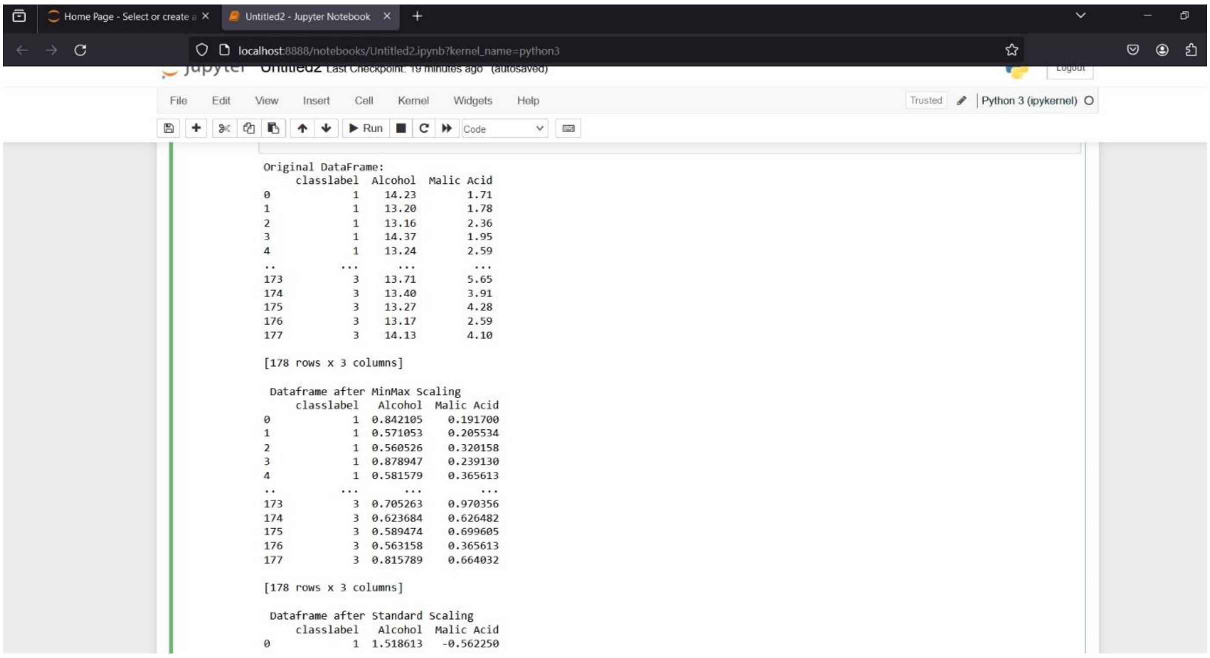
Practical no 3

Feature Scaling and Dummification

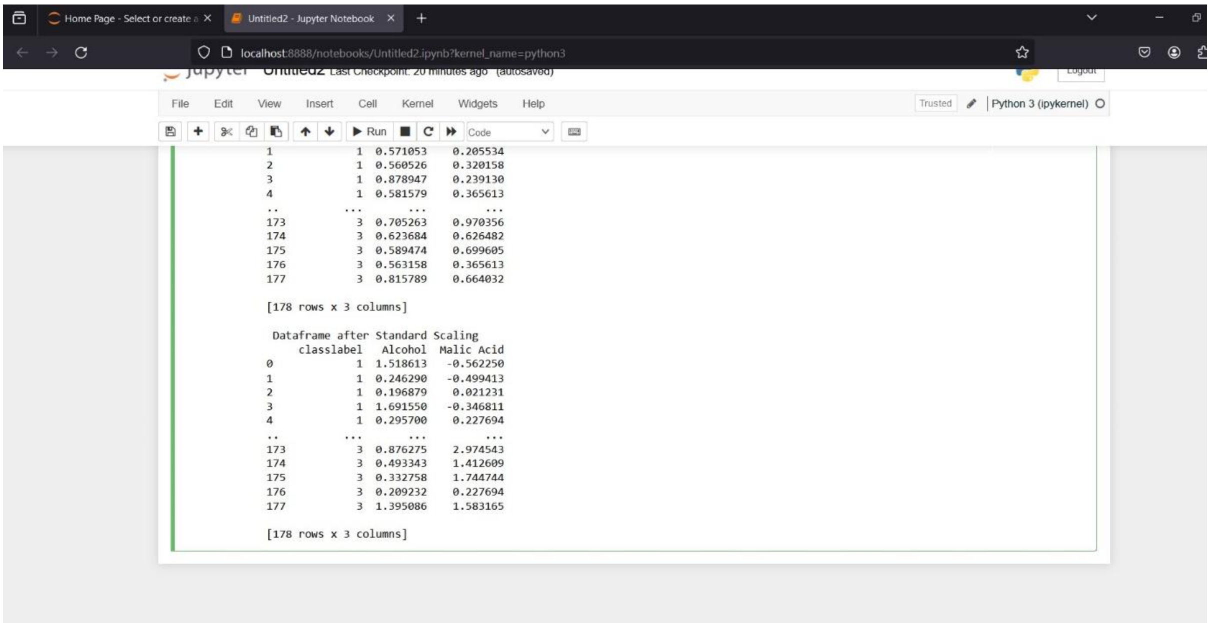
A. Apply feature-scaling techniques like standardization and normalization to numerical features.

Code:

```
# Standardization and normalization
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler,
StandardScaler
df = pd.read_csv('D:\\data science\\wine.csv', header=None,
usecols=[0, 1, 2], skiprows=1)
df.columns = ['classlabel', 'Alcohol', 'Malic Acid']
print("Original DataFrame:")
print(df)
scaling=MinMaxScaler()
scaled_value=scaling.fit_transform(df[['Alcohol','Malic Acid']])
df[['Alcohol','Malic Acid']]=scaled_value
print("\n Dataframe after MinMax Scaling")
print(df)
scaling=StandardScaler()
scaled_standardvalue=scaling.fit_transform(df[['Alcohol','Malic
Acid']])
df[['Alcohol','Malic Acid']]=scaled_standardvalue
print("\n Dataframe after Standard Scaling")
print(df)
```



```
Original DataFrame:
     classlabel  Alcohol  Malic Acid
0             1    14.23        1.71
1             1    13.20        1.78
2             1    13.16        2.36
3             1    14.37        1.95
4             1    13.24        2.59
..          ...      ...         ...
173           3    13.71        5.65
174           3    13.40        3.91
175           3    13.27        4.28
176           3    13.17        2.59
177           3    14.13        4.10

[178 rows x 3 columns]

 Dataframe after MinMax Scaling
     classlabel   Alcohol  Malic Acid
0             1  0.842105    0.191700
1             1  0.571053    0.205534
2             1  0.560526    0.320158
3             1  0.878947    0.239130
4             1  0.581579    0.365613
..          ...       ...         ...
173           3  0.705263    0.970356
174           3  0.623684    0.626482
175           3  0.589474    0.699605
176           3  0.563158    0.365613
177           3  0.815789    0.664032

[178 rows x 3 columns]

 Dataframe after Standard Scaling
     classlabel   Alcohol  Malic Acid
0             1  1.518613   -0.562250
```

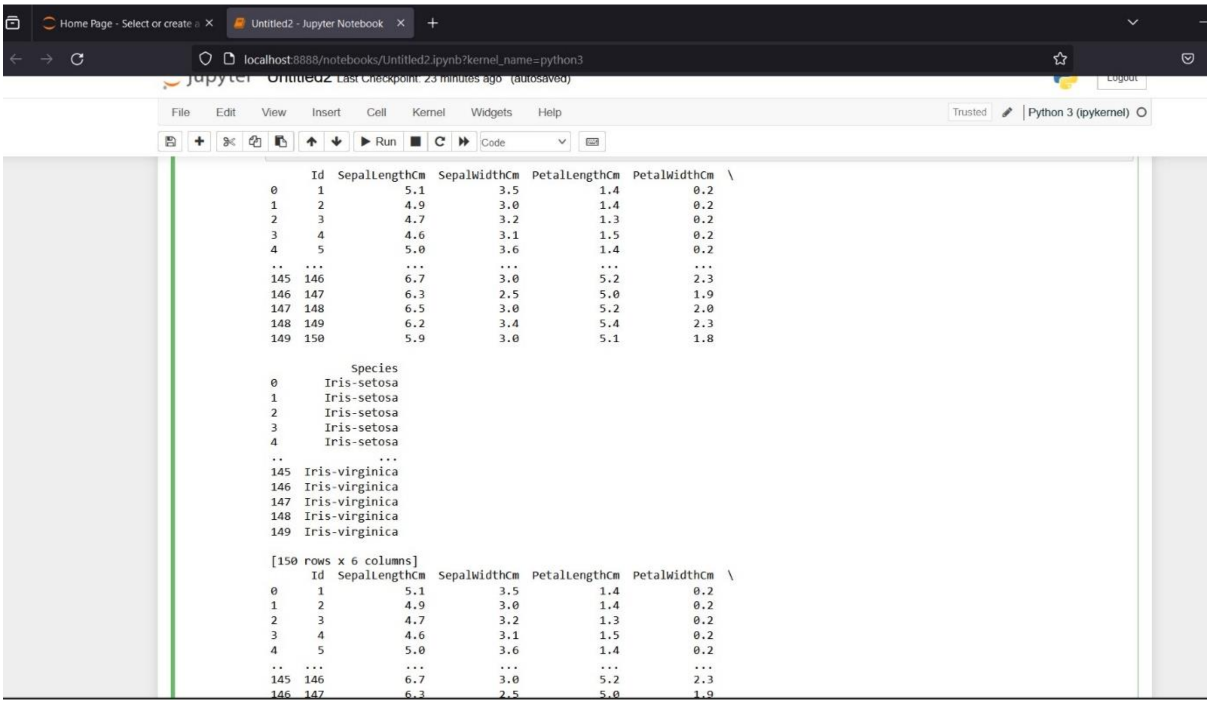


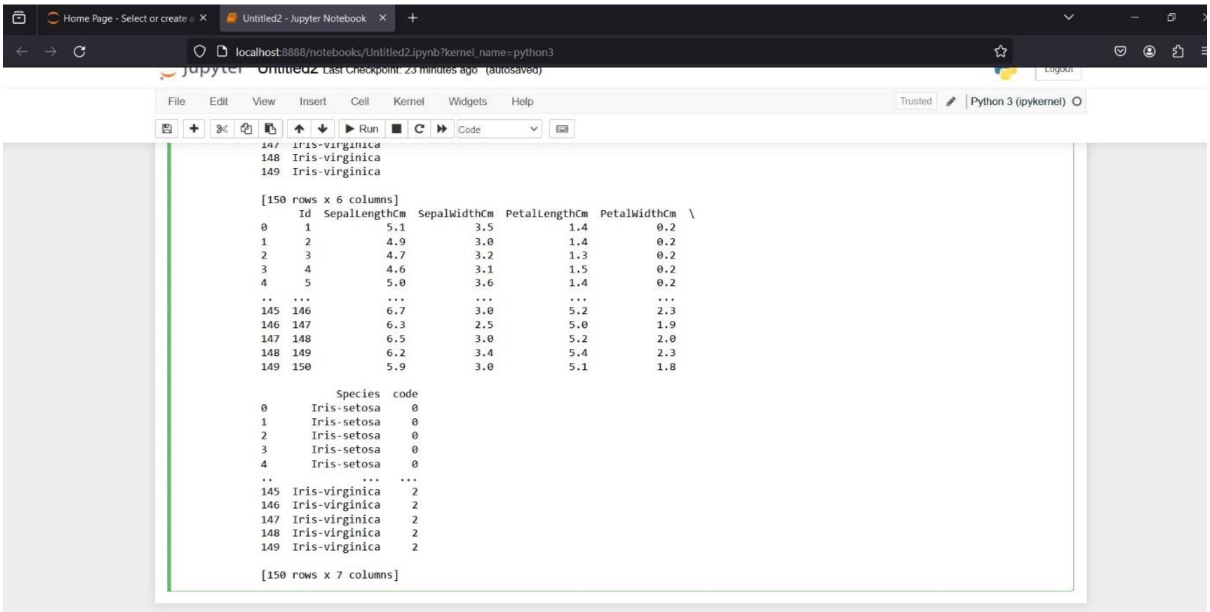
```
1             1  0.571053    0.205534
2             1  0.560526    0.320158
3             1  0.878947    0.239130
4             1  0.581579    0.365613
..          ...       ...         ...
173           3  0.705263    0.970356
174           3  0.623684    0.626482
175           3  0.589474    0.699605
176           3  0.563158    0.365613
177           3  0.815789    0.664032

[178 rows x 3 columns]

 Dataframe after Standard Scaling
     classlabel   Alcohol  Malic Acid
0             1  1.518613   -0.562250
1             1  0.246290   -0.499413
2             1  0.196879    0.021231
3             1  1.691550   -0.346811
4             1  0.295700    0.227694
..          ...       ...         ...
173           3  0.876275    2.974543
174           3  0.493343    1.412609
175           3  0.332758    1.744744
176           3  0.209232    0.227694
177           3  1.395086    1.583165

[178 rows x 3 columns]
```

B. Perform feature Dummification to convert categorical variables into numerical representations.

Code:

```
import pandas as pd
iris=pd.read_csv("D:\\data science\\Iris.csv")
print(iris)
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
iris['code']=le.fit_transform(iris.Species)
print(iris)
```





```
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species  code
0       Iris-setosa     0
1       Iris-setosa     0
2       Iris-setosa     0
3       Iris-setosa     0
4       Iris-setosa     0
..              ...   ...
145  Iris-virginica     2
146  Iris-virginica     2
147  Iris-virginica     2
148  Iris-virginica     2
149  Iris-virginica     2

[150 rows x 7 columns]
```

Practical no 4

Hypothesis Testing

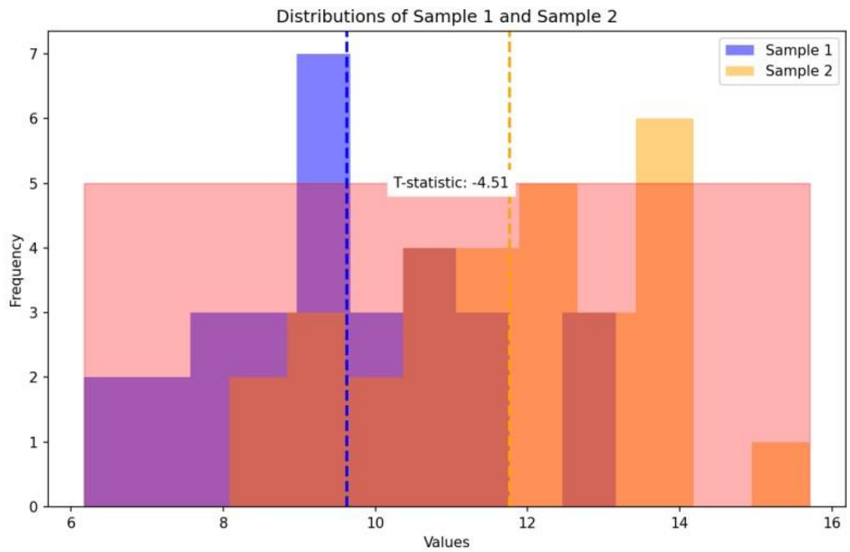
Conduct a hypothesis test using appropriate statistical tests (e.g., t-test, chi-square test)

```
# t-test
import numpy as np
```

```
from scipy import stats
import matplotlib.pyplot as plt

# Generate two samples for demonstration purposes
np.random.seed(42)
sample1 = np.random.normal(loc=10, scale=2, size=30)
sample2 = np.random.normal(loc=12, scale=2, size=30)

# Perform a two-sample t-test
t_statistic, p_value = stats.ttest_ind(sample1, sample2)

# Set the significance level
alpha = 0.05

print("Results of Two-Sample t-test:")
print(f'T-statistic: {t_statistic}')
print(f'P-value: {p_value}')
print(f"Degrees of Freedom: {len(sample1) + len(sample2) - 2}")

# Plot the distributions
plt.figure(figsize=(10, 6))
plt.hist(sample1, alpha=0.5, label='Sample 1', color='blue')
plt.hist(sample2, alpha=0.5, label='Sample 2', color='orange')
plt.axvline(np.mean(sample1), color='blue', linestyle='dashed',
linewidth=2)
plt.axvline(np.mean(sample2), color='orange', linestyle='dashed',
linewidth=2)
plt.title('Distributions of Sample 1 and Sample 2')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.legend()

# Highlight the critical region if null hypothesis is rejected
if p_value < alpha:
   critical_region = np.linspace(min(sample1.min(), sample2.min()),
max(sample1.max(),sample2.max()), 1000)
```

```
    plt.fill_between(critical_region, 0, 5, color='red', alpha=0.3,
label='Critical Region')
    plt.text(11, 5, f'T-statistic: {t_statistic:.2f}', ha='center', va='center',
color='black',
backgroundcolor='white')

# Draw Conclusions
if p_value < alpha:
    if np.mean(sample1) > np.mean(sample2):
        print("Conclusion: There is significant evidence to reject the null
hypothesis.")
        print("Interpretation: The mean of Sample 1 is significantly higher
than that of Sample2.")
    else:
        print("Conclusion: There is significant evidence to reject the null
hypothesis.")
        print("Interpretation: The mean of Sample 2 is significantly higher
than that of Sample1.")
else:
    print("Conclusion: Fail to reject the null hypothesis.")
    print("Interpretation: There is not enough evidence to claim a
significant differencebetween the means.")
```

```
Results of Two-Sample t-test:
T-statistic: -4.512913234547555
P-value: 3.176506547470154e-05
Degrees of Freedom: 58
```



#chi-test (run this code in IDLE)

```
import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sb
import warnings
from scipy import stats
warnings.filterwarnings('ignore')
df=sb.load_dataset('mpg')
print(df)
print(df['horsepower'].describe())
print(df['model_year'].describe())
bins=[0,75,150,240]
df['horsepower_new']=pd.cut(df['horsepower'],bins=bins,labels=['l','m','
h'])
c=df['horsepower_new']
print(c)
ybins=[69,72,74,84]
label=['t1','t2','t3']
df['modelyear_new']=pd.cut(df['model_year'],bins=ybins,labels=label)
```

newyear=df['modelyear_new']
print(newyear)
df_chi=pd.crosstab(df['horsepower_new'],df['modelyear_new'])
print(df_chi)
print(stats.chi2_contingency(df_chi))

```
      mpg  cylinders  ...  origin                       name
0    18.0          8  ...     usa  chevrolet chevelle malibu
1    15.0          8  ...     usa          buick skylark 320
2    18.0          8  ...     usa         plymouth satellite
3    16.0          8  ...     usa              amc rebel sst
4    17.0          8  ...     usa                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...     usa            ford mustang gl
394  44.0          4  ...  europe                  vw pickup
395  32.0          4  ...     usa              dodge rampage
396  28.0          4  ...     usa                ford ranger
397  31.0          4  ...     usa                 chevy s-10

[398 rows x 9 columns]
count    392.000000
mean     104.469388
std       38.491160
min       46.000000
25%       75.000000
50%       93.500000
75%      126.000000
max      230.000000
Name: horsepower, dtype: float64
count    398.000000
mean      76.010050
std        3.697627
min       70.000000
```

```
min       46.000000
25%       75.000000
50%       93.500000
75%      126.000000
max      230.000000
Name: horsepower, dtype: float64
count    398.000000
mean      76.010050
std        3.697627
min       70.000000
25%       73.000000
50%       76.000000
75%       79.000000
max       82.000000
Name: model_year, dtype: float64
0      m
1      h
2      m
3      m
4      m
      ..
393    m
394    l
395    m
396    m
397    m
Name: horsepower_new, Length: 398, dtype: category
Categories (3, object): ['l' < 'm' < 'h']
0      t1
1      t1
2      t1
```

```
      ..
393    m
394    l
395    m
396    m
397    m
Name: horsepower_new, Length: 398, dtype: category
Categories (3, object): ['l' < 'm' < 'h']
0      t1
1      t1
2      t1
3      t1
4      t1
       ..
393    t3
394    t3
395    t3
396    t3
397    t3
Name: modelyear_new, Length: 398, dtype: category
Categories (3, object): ['t1' < 't2' < 't3']
modelyear_new   t1  t2   t3
horsepower_new
l                9  14   76
m               49  41  158
h               26  11    8
Chi2ContingencyResult(statistic=54.95485392447537, pvalue=3.320518009555984e-11, dof=4, expected_freq=array([[ 21.
21428571,  16.66836735,  61.11734694],
       [ 53.14285714,  41.75510204, 153.10204082],
       [  9.64285714,   7.57653061,  27.78061224]]))
>>>
```

Practical no 5

ANOVA (Analysis of Variance)

Perform one-way ANOVA to compare means across multiple groups. Conduct post-hoc tests to identify significant differences between group means

```
import pandas as pd
import scipy.stats as stats
from statsmodels.stats.multicomp import pairwise_tukeyhsd

group1 = [23, 25, 29, 34, 30]
group2 = [19, 20, 22, 24, 25]
group3 = [15, 18, 20, 21, 17]
group4 = [28, 24, 26, 30, 29]

all_data = group1 + group2 + group3 + group4
group_labels = ['Group1'] * len(group1) + ['Group2'] *
len(group2) + ['Group3'] * len(group3) + ['Group4'] * len(group4)
```

(keep all group_labels in one line otherwise it will show error)

```
f_statistics, p_value = stats.f_oneway(group1, group2, group3,
group4)
print("one-way ANOVA:")
print("F-statistics:", f_statistics)
print("p-value", p_value)
tukey_results = pairwise_tukeyhsd(all_data, group_labels)
print("\nTukey-Kramer post-hoc test:")
print(tukey_results)
```

```
one-way ANOVA:
F-statistics: 12.139872842870115
p-value 0.00021465200901629603

Tukey-Kramer post-hoc test:
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower    upper  reject
-----------------------------------------------------
Group1 Group2     -6.2  0.024 -11.6809 -0.7191   True
Group1 Group3    -10.0 0.0004 -15.4809 -4.5191   True
Group1 Group4     -0.8 0.9747  -6.2809  4.6809  False
Group2 Group3     -3.8 0.2348  -9.2809  1.6809  False
Group2 Group4      5.4 0.0542  -0.0809 10.8809  False
Group3 Group4      9.2  0.001   3.7191 14.6809   True
-----------------------------------------------------
```

Practical no 6

Regression and its Types.

```
import numpy as np
import pandas as pd
from sklearn.datasets import
fetch_california_housing
from sklearn.model_selection import
train_test_split
from sklearn.linear_model import
LinearRegression
from sklearn.metrics import
mean_squared_error, r2_score

housing = fetch_california_housing()
housing_df = pd.DataFrame(housing.data,
columns=housing.feature_names)
print(housing_df)

housing_df['PRICE'] = housing.target

X = housing_df[['AveRooms']]
y = housing_df['PRICE']

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
model = LinearRegression()
model.fit(X_train, y_train)

mse = mean_squared_error(y_test,
model.predict(X_test))
r2 = r2_score(y_test, model.predict(X_test))
print("Mean Squared Error:", mse)
print("R-squared:", r2)
print("Intercept:", model.intercept_)
print("Coefficient:", model.coef_)

#multi Linear Regression

X = housing_df.drop('PRICE',axis=1)
y = housing_df['PRICE']

X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state
=42)

model = LinearRegression()

model.fit(X_train,y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test,y_pred)
```

```
r2 = r2_score(y_test,y_pred)

print("Mean Squared Error:",mse)
print("R-squared:",r2)
print("Intercept:",model.intercept_)
print("Coefficient:",model.coef_)
```

```
       MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude
0      8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1      8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2      7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3      5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4      3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85
...       ...       ...       ...        ...         ...       ...       ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37

       Longitude
0        -122.23
1        -122.22
2        -122.24
3        -122.25
4        -122.25
...          ...
20635    -121.09
20636    -121.21
20637    -121.22
20638    -121.32
20639    -121.24

[20640 rows x 8 columns]
```

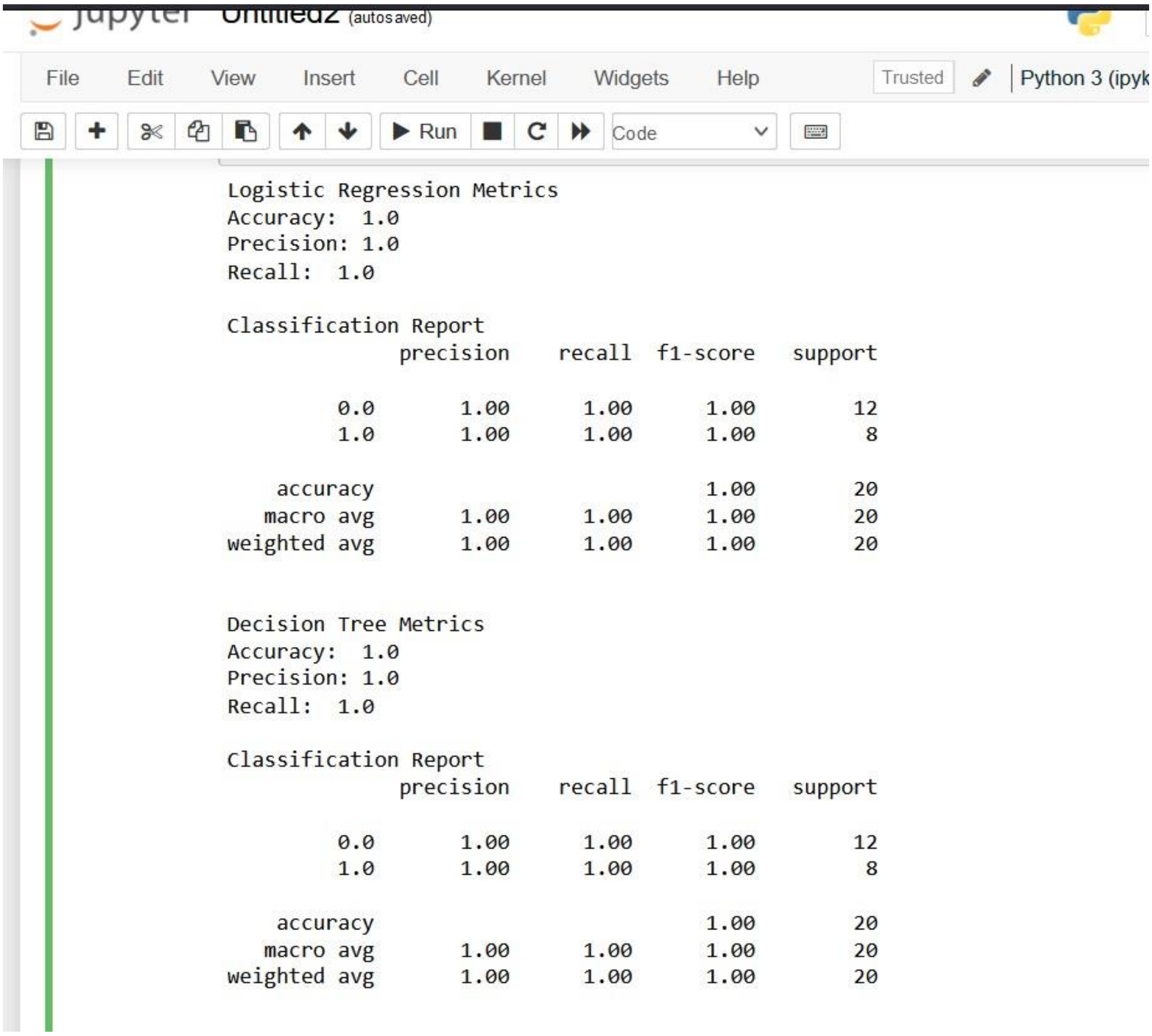
```
[20640 rows x 8 columns]
Mean Squared Error: 1.2923314440807299
R-squared: 0.013795337532284901
Intercept: 1.654762268596842
Coefficient: [0.07675559]
Mean Squared Error: 0.5558915986952441
R-squared: 0.575787706032451
Intercept: -37.02327770606414
Coefficient: [ 4.48674910e-01  9.72425752e-03 -1.23323343e-01  7.83144907e-01
 -2.02962058e-06 -3.52631849e-03 -4.19792487e-01 -4.33708065e-01]
```

Practical no 7

Logistic Regression and Decision Tree

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score,classification_report
# Load the Iris dataset and create a binary classification problem
iris = load_iris()
iris_df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
columns=iris['feature_names'] +
['target'])
binary_df = iris_df[iris_df['target'] != 2]
X = binary_df.drop('target', axis=1)
y = binary_df['target']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Train a logistic regression model and evaluate its performance
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_test)

print("Logistic Regression Metrics")
print("Accuracy: ", accuracy_score(y_test, y_pred_logistic))
print("Precision:", precision_score(y_test, y_pred_logistic))
print("Recall: ", recall_score(y_test, y_pred_logistic))
print("\nClassification Report")
print(classification_report(y_test, y_pred_logistic))
# Train a decision tree model and evaluate its performance
decision_tree_model = DecisionTreeClassifier()
```

```
decision_tree_model.fit(X_train, y_train)
y_pred_tree = decision_tree_model.predict(X_test)
print("\nDecision Tree Metrics")
print("Accuracy: ", accuracy_score(y_test, y_pred_tree))
print("Precision:", precision_score(y_test, y_pred_tree))
print("Recall: ", recall_score(y_test, y_pred_tree))
print("\nClassification Report")
print(classification_report(y_test, y_pred_tree))
```

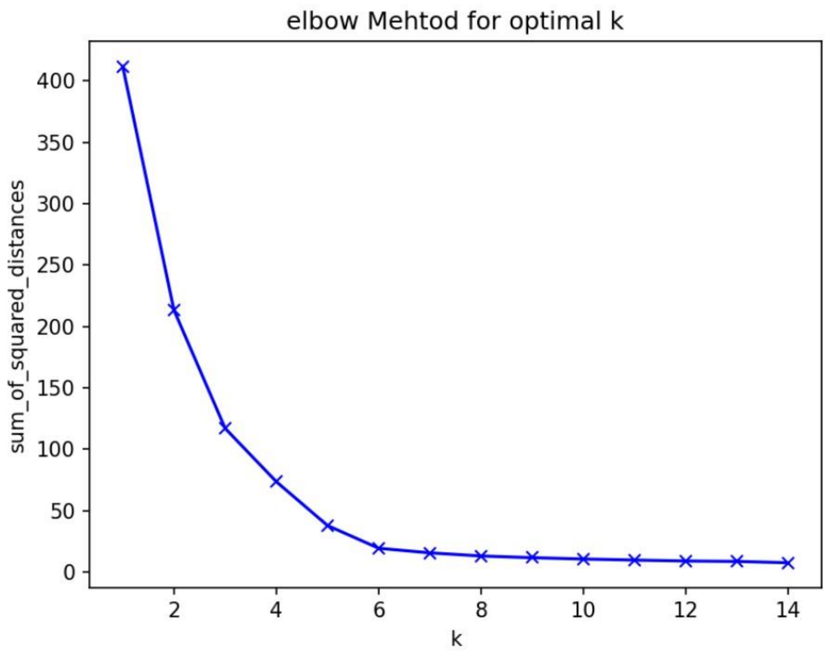


Practical no 8

K-Means clustering

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

data = pd.read_csv("D:\\data science\\wholesale.csv")
data.head()

categorical_features = ['Channel', 'Region']
continuous_features = ['Fresh', 'Milk', 'Grocery',
'Frozen', 'Detergents_Paper', 'Delicassen']
data[continuous_features].describe()

for col in categorical_features:
    dummies = pd.get_dummies(data[col], prefix = col)
    data = pd.concat([data, dummies], axis = 1)
    data.drop(col, axis = 1, inplace = True)
data.head()

mms = MinMaxScaler()
mms.fit(data)
data_transformed = mms.transform(data)

sum_of_squared_distances = []
K = range(1, 15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(data_transformed)
```

```
sum_of_squared_distances.append(km.inertia_)

plt.plot(K, sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('sum_of_squared_distances')
plt.title('elbow Mehtod for optimal k')
plt.show()
```

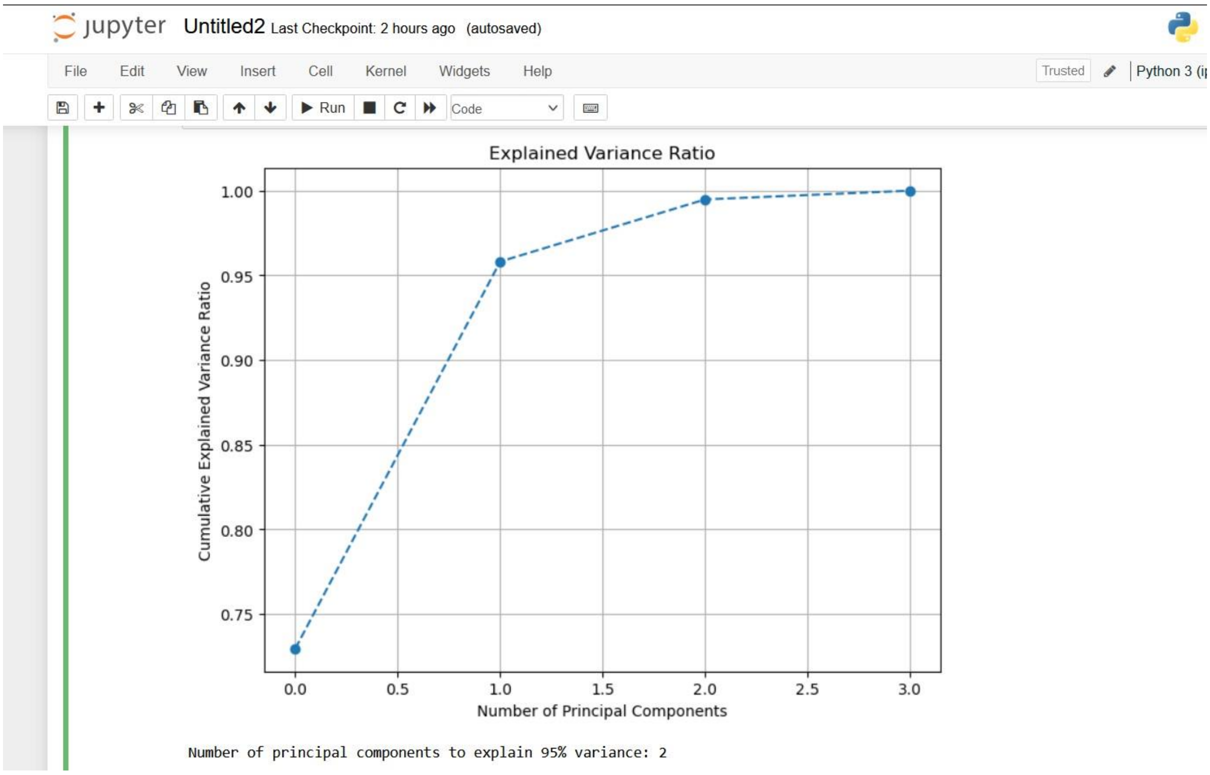


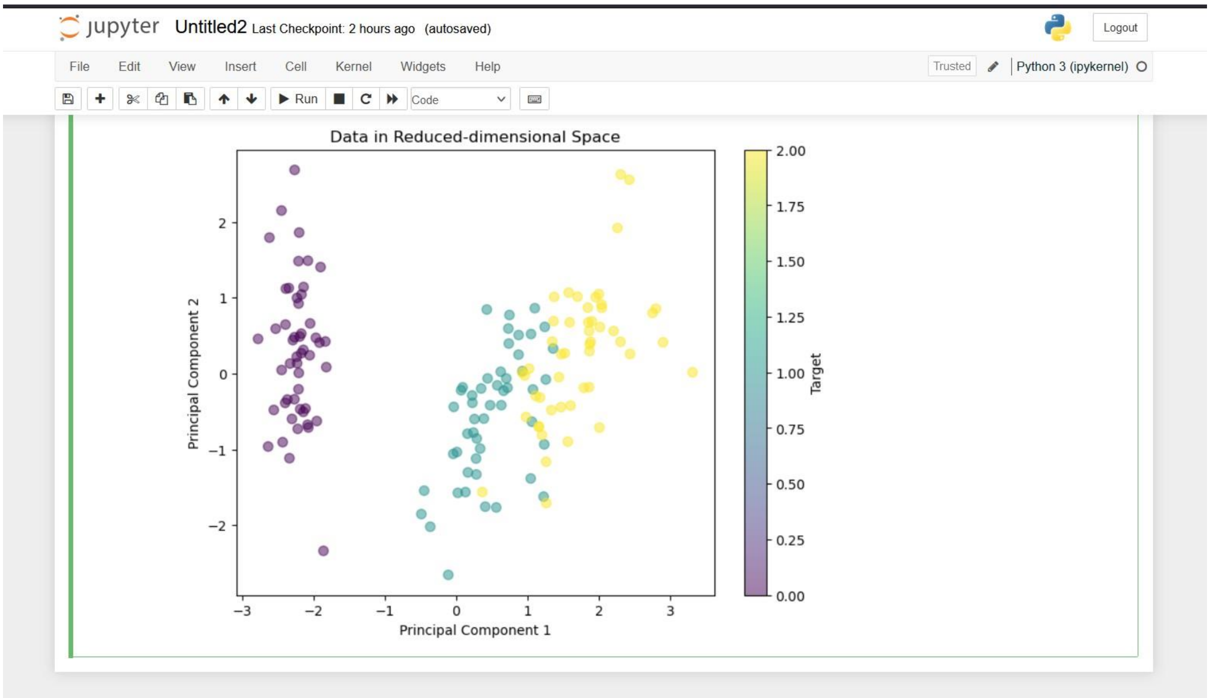
Practical no 9

Principal Component Analysis (PCA)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

iris = load_iris()
iris_df = pd.DataFrame(data=np.c_[iris['data'],
iris['target']], columns=iris['feature_names'] +
['target'])
X = iris_df.drop('target', axis=1)
y = iris_df['target']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA()
X_pca = pca.fit_transform(X_scaled)
explained_variance_ratio =
pca.explained_variance_ratio_

plt.figure(figsize=(8, 6))
plt.plot(np.cumsum(explained_variance_ratio),
marker='o', linestyle='--')
```

```
plt.title('Explained Variance Ratio')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.grid(True)

plt.show()

cumulative_variance_ratio =
np.cumsum(explained_variance_ratio)
n_components =
np.argmax(cumulative_variance_ratio >= 0.95) + 1
print(f"Number of principal components to explain
95% variance: {n_components}")

pca = PCA(n_components=n_components)
X_reduced = pca.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y,
cmap='viridis', s=50, alpha=0.5)
plt.title('Data in Reduced-dimensional Space')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Target')
plt.show()
```


jupyter Untitled2 Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (i
Run
Code
Explained Variance Ratio
Cumulative Explained Variance Ratio
1.00
0.95
0.90
0.85
0.80
0.75
0.0 0.5 1.0 1.5 2.0 2.5 3.0
Number of Principal Components
Number of principal components to explain 95% variance: 2


jupyter Untitled2 Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Data in Reduced-dimensional Space
Principal Component 2
2
1
0
−1
−2
−3 −2 −1 0 1 2 3
Principal Component 1
Target
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

Practical no 10

Data Visualization and Storytelling

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Generate random data
np.random.seed(42) # Set a seed for reproducibility

# Create a DataFrame with random data
data = pd.DataFrame({
'variable1': np.random.normal(0, 1, 1000),
'variable2': np.random.normal(2, 2, 1000) + 0.5 *
np.random.normal(0, 1, 1000),
 'variable3': np.random.normal(-1, 1.5, 1000),
 'category': pd.Series(np.random.choice(['A', 'B', 'C', 'D'],
size=1000, p=[0.4, 0.3, 0.2, 0.1]),
dtype='category')
})

# Create a scatter plot to visualize the relationship between
two variables
plt.figure(figsize=(10, 6))
plt.scatter(data['variable1'], data['variable2'], alpha=0.5)
plt.title('Relationship between Variable 1 and Variable 2',
fontsize=16)
plt.xlabel('Variable 1', fontsize=14)
plt.ylabel('Variable 2', fontsize=14)
plt.show()

# Create a bar chart to visualize the distribution of a
categorical variable
plt.figure(figsize=(10, 6))
```

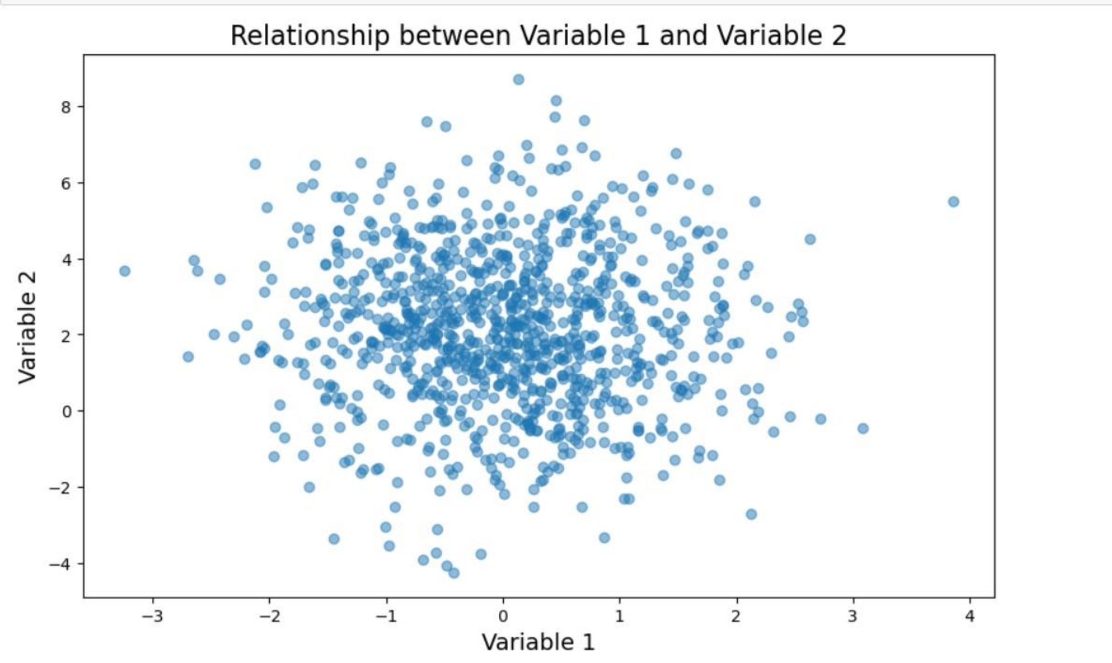
```
sns.countplot(x='category', data=data)
plt.title('Distribution of Categories', fontsize=16)
plt.xlabel('Category', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=45)
plt.show()

# Create a heatmap to visualize the correlation between
numerical variables
plt.figure(figsize=(10, 8))
numerical_cols = ['variable1', 'variable2', 'variable3']
sns.heatmap(data[numerical_cols].corr(), annot=True,
cmap='coolwarm')
plt.title('Correlation Heatmap', fontsize=16)
plt.show()

# Data Storytelling
print("Title: Exploring the Relationship between Variable 1 and
Variable 2")
print("\nThe scatter plot (Figure 1) shows the relationship
between Variable 1 and Variable 2. We can observe a positive
correlation, indicating that as Variable 1 increases, Variable 2
tends to increase as well. However, there is a considerable
amount of scatter, suggesting that other factors may influence
this relationship.")
print("\nScatter Plot")
print("Figure 1: Scatter Plot of Variable 1 and Variable 2")

print("\nTo better understand the distribution of the
categorical variable 'category', we created a bar chart (Figure
2). The chart reveals that Category A has the highest
frequency, followed by Category B, Category C, and Category
D. This information could be useful for further analysis or
decision-making processes.")
print("\nBar Chart")
print("Figure 2: Distribution of Categories")
print("\nAdditionally, we explored the correlation between
numerical variables using a heatmap (Figure 3). The heatmap
```

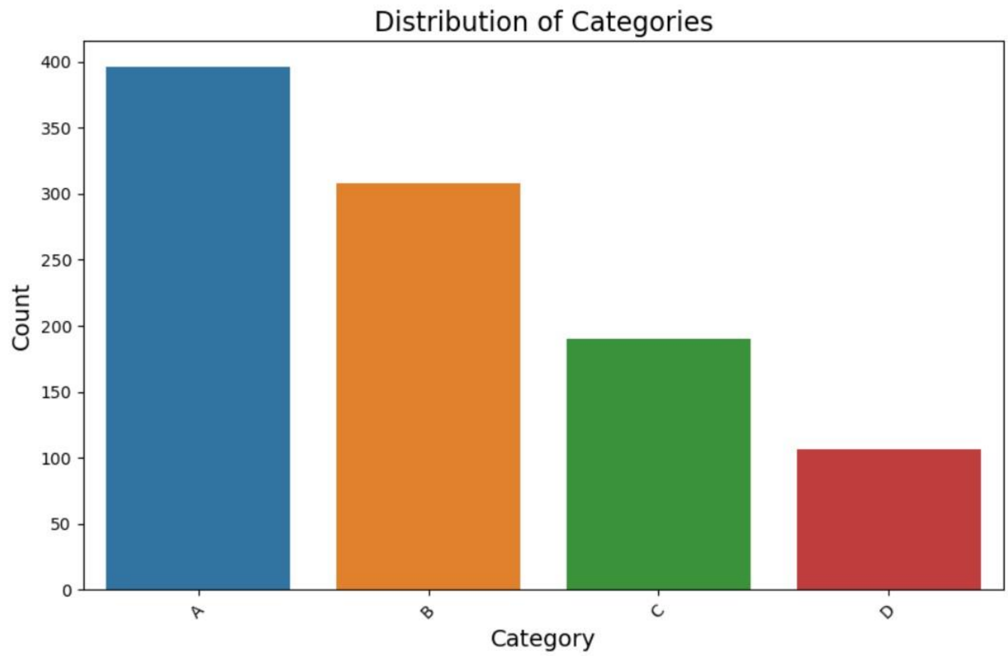
shows that Variable 1 and Variable 2 have a strong positive correlation, confirming the observation from the scatter plot. However, we can also see that Variable 3 has a moderate negative correlation with both Variable 1 and Variable 2, suggesting that it may have an opposing effect on the relationship between the first two variables.")

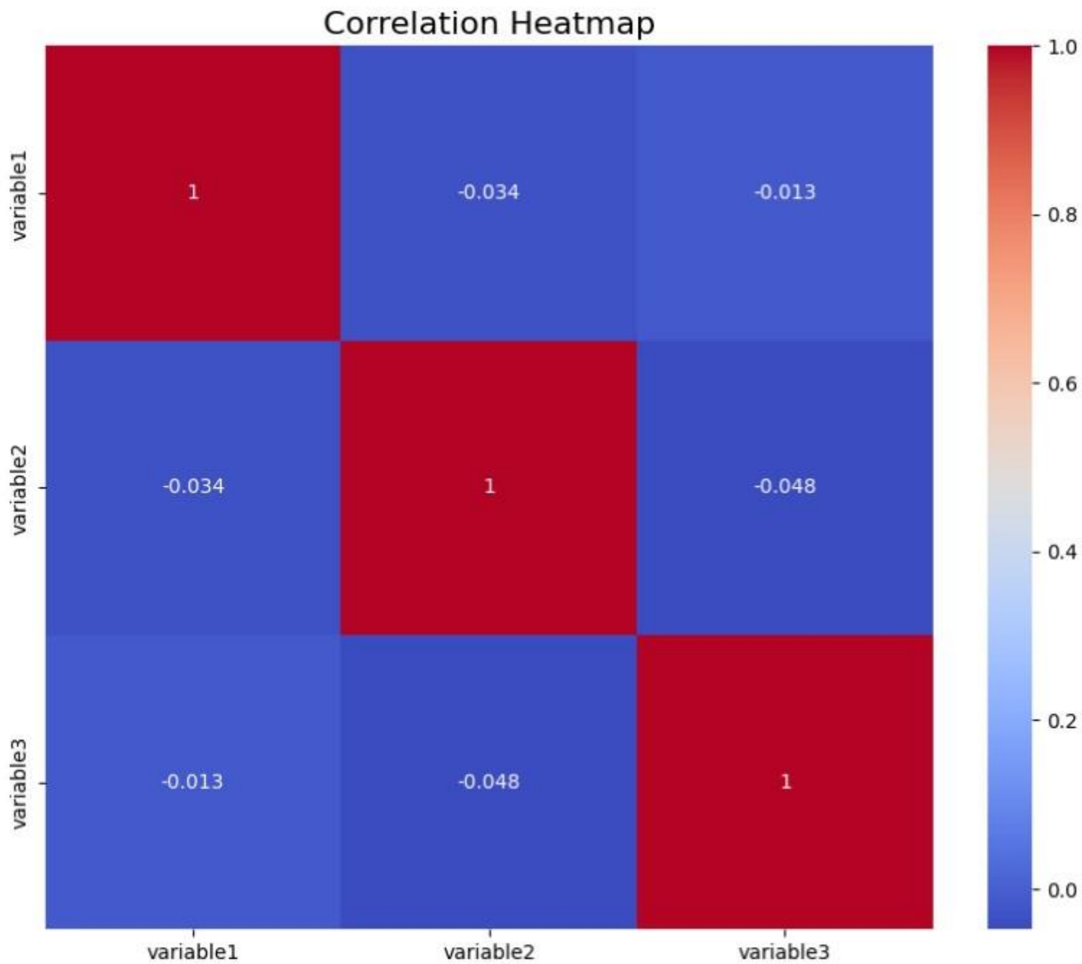
print("\nHeatmap")

print("Figure 3: Correlation Heatmap")

print("\nIn summary, the visualizations and analysis provide insights into the relationships between variables, the distribution of categories, and the correlations between numerical variables. These findings can be used to inform further analysis, decision-making, or to generate new hypotheses for investigation.")




Distribution of Categories
400
350
300
250
200
150
100
50
0
Count
A
B
C
D
Category


Correlation Heatmap
variable1
variable2
variable3
1
-0.034
-0.013
-0.034
1
-0.048
-0.013
-0.048
1
variable1
variable2
variable3
1.0
0.8
0.6
0.4
0.2
0.0

Title: Exploring the Relationship between Variable 1 and Variable 2

The scatter plot (Figure 1) shows the relationship between Variable 1 and Variable 2. We can observe a positive correlation, in
dicating that as Variable 1 increases, Variable 2 tends to increase as well. However, there is a considerable amount of scatter
, suggesting that other factors may influence this relationship.

Scatter Plot
Figure 1: Scatter Plot of Variable 1 and Variable 2

To better understand the distribution of the categorical variable 'category', we created a bar chart (Figure 2). The chart reve
als that Category A has the highest frequency, followed by Category B, Category C, and Category D. This information could be us
eful for further analysis or decision-making processes.

Bar Chart
Figure 2: Distribution of Categories

Additionally, we explored the correlation between numerical variables using a heatmap (Figure 3). The heatmap shows that Variab
le 1 and Variable 2 have a strong positive correlation, confirming the observation from the scatter plot. However, we can also

Additionally, we explored the correlation between numerical variables using a heatmap (Figure 3). The heatmap shows that Variab
le 1 and Variable 2 have a strong positive correlation, confirming the observation from the scatter plot. However, we can also
see that Variable 3 has a moderate negative correlation with both Variable 1 and Variable 2, suggesting that it may have an opp
osing effect on the relationship between the first two variables.

Heatmap
Figure 3: Correlation Heatmap

In summary, the visualizations and analysis provide insights into the relationships between variables, the distribution of cate
gories, and the correlations between numerical variables. These findings can be used to inform further analysis, decision-makin
g, or to generate new hypotheses for investigation.