



DATA SCIENCE LAB MANUAL

TYCS SEMESTER-VI



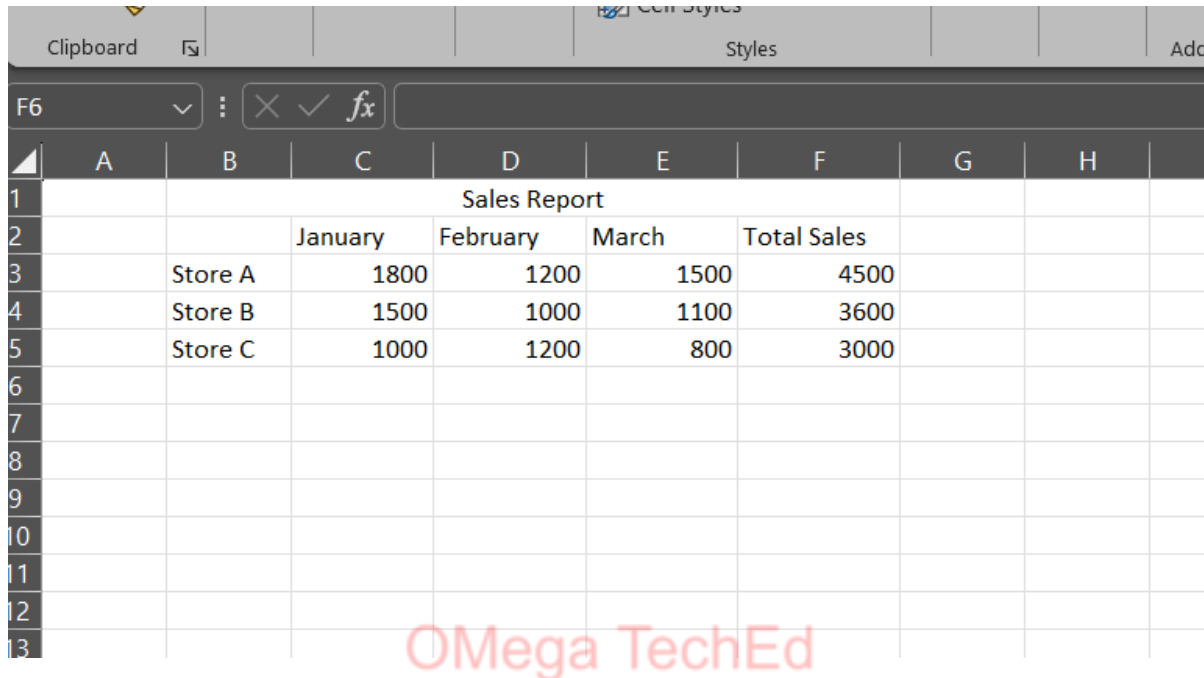
“Special thanks to, Nikhil Singh, Ashi Chauhan and Dinesh Chaudhary for their co-operation to compile this document.”

*Compiled by:
Asst. Prof. Megha Sharma*

PRACTICAL 1

Introduction to Excel

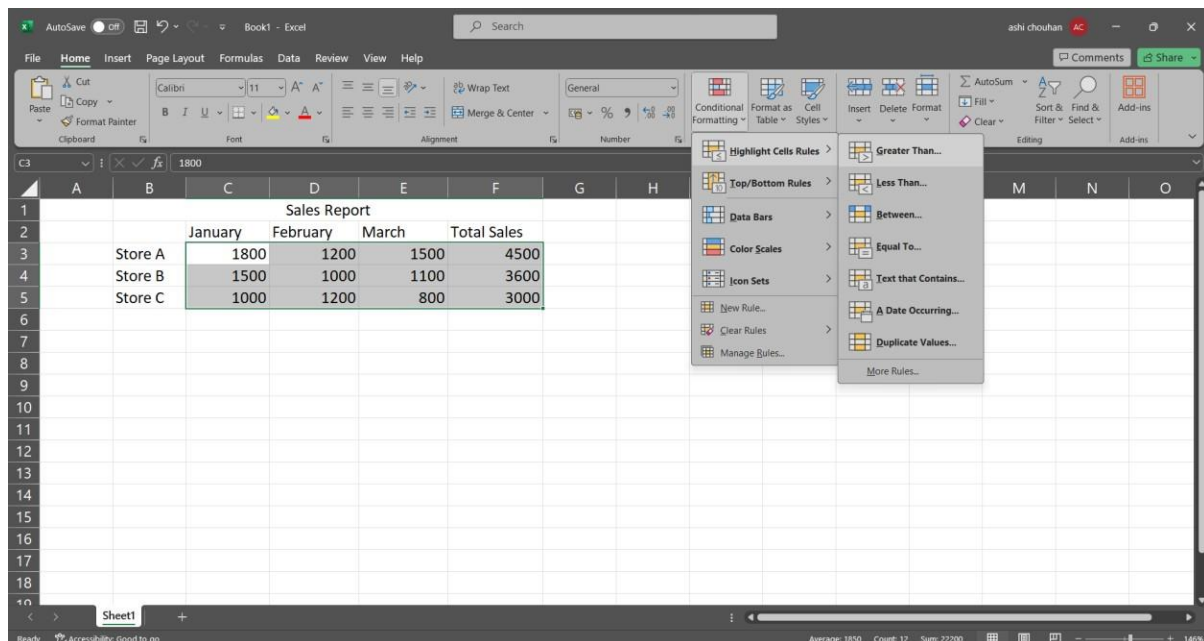
A. Perform conditional formatting on a dataset using various criteria.



Sales Report					
	January	February	March	Total Sales	
Store A	1800	1200	1500	4500	
Store B	1500	1000	1100	3600	
Store C	1000	1200	800	3000	

Steps

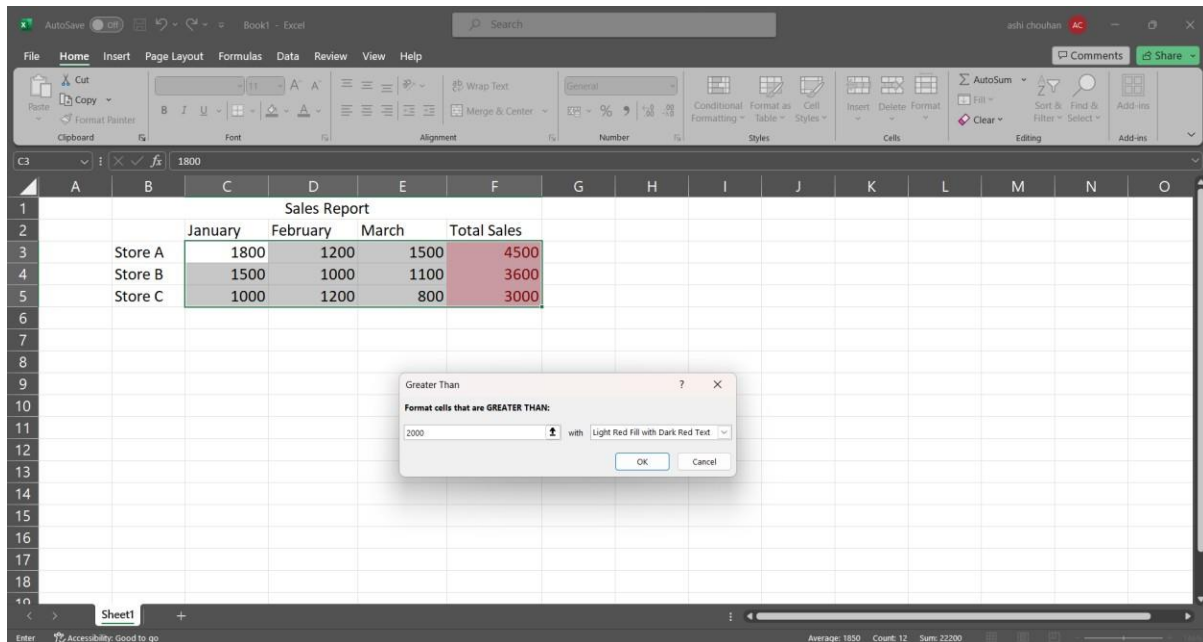
Step 1: Go to conditional formatting > Greater Than



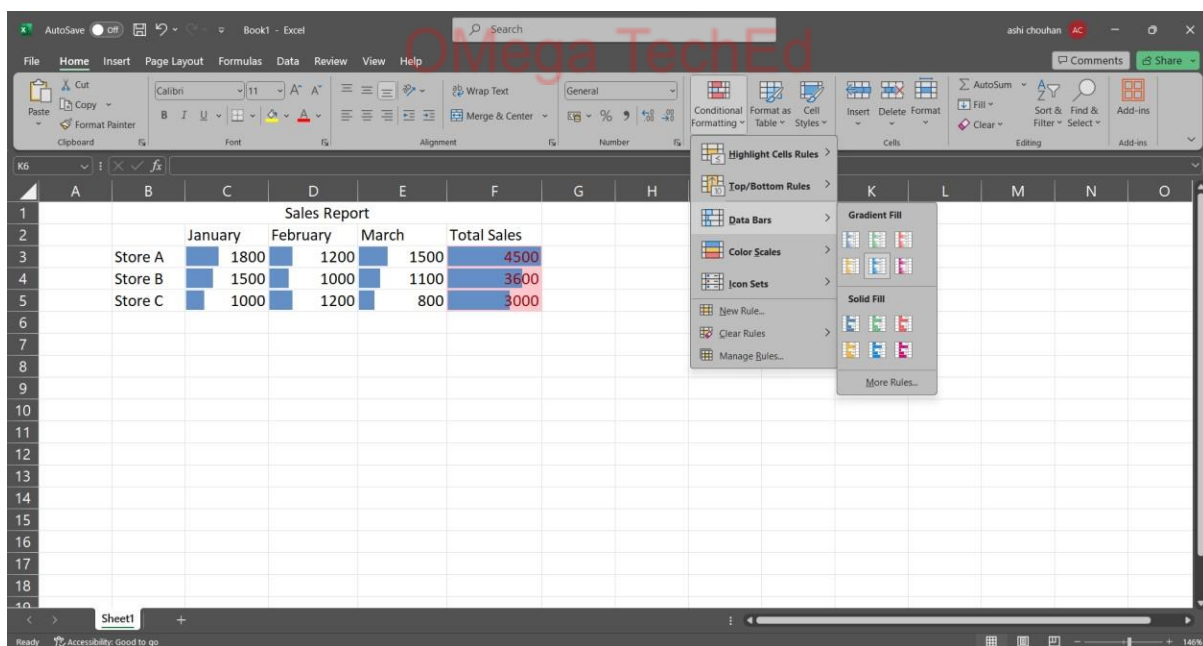
Step 2: Enter the greater than filter value for example 2000.

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)



Step 3: Go to Data Bars > Solid Fill in conditional formatting.



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

For video demonstration of the practical click on the below link:

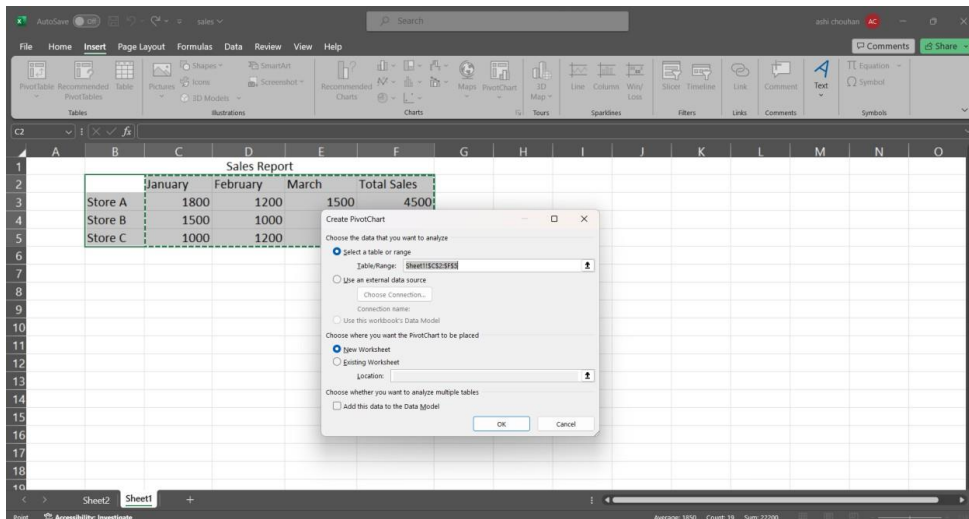
[Data Science Practical Playlist](#)

http://www.youtube.com/@omega_teched

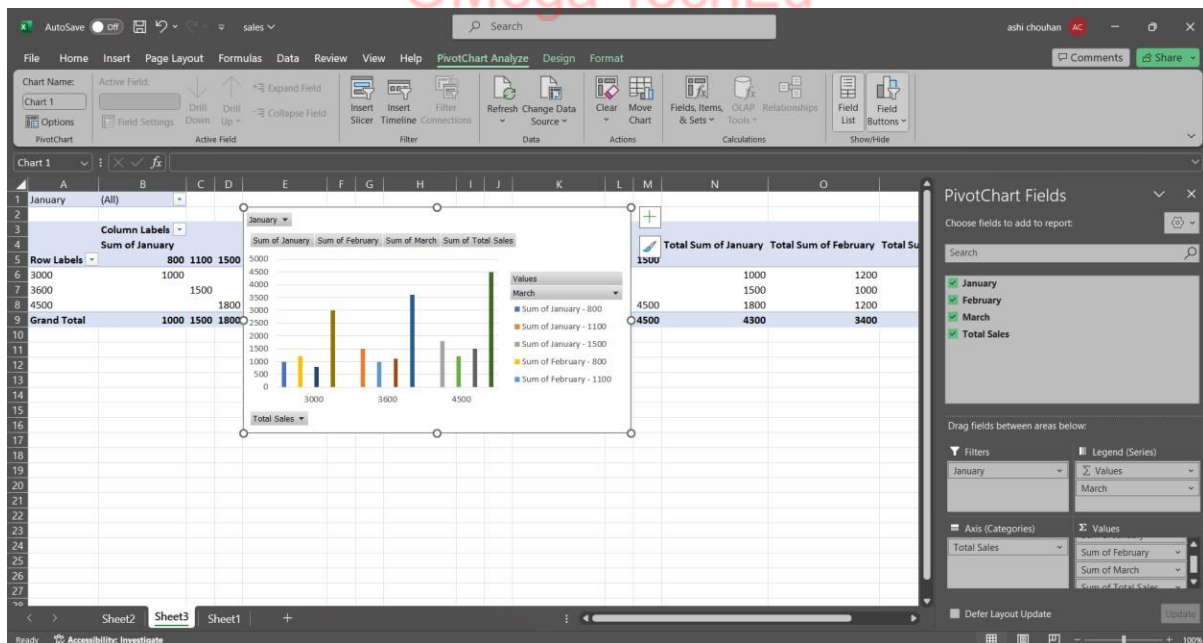
Steps

Step 1: select the entire table and go to Insert tab PivotChart > Pivotchart.

Step 2: Select “New worksheet” in the create pivot chart window.



Step 3: Select and drag attributes in the below boxes.



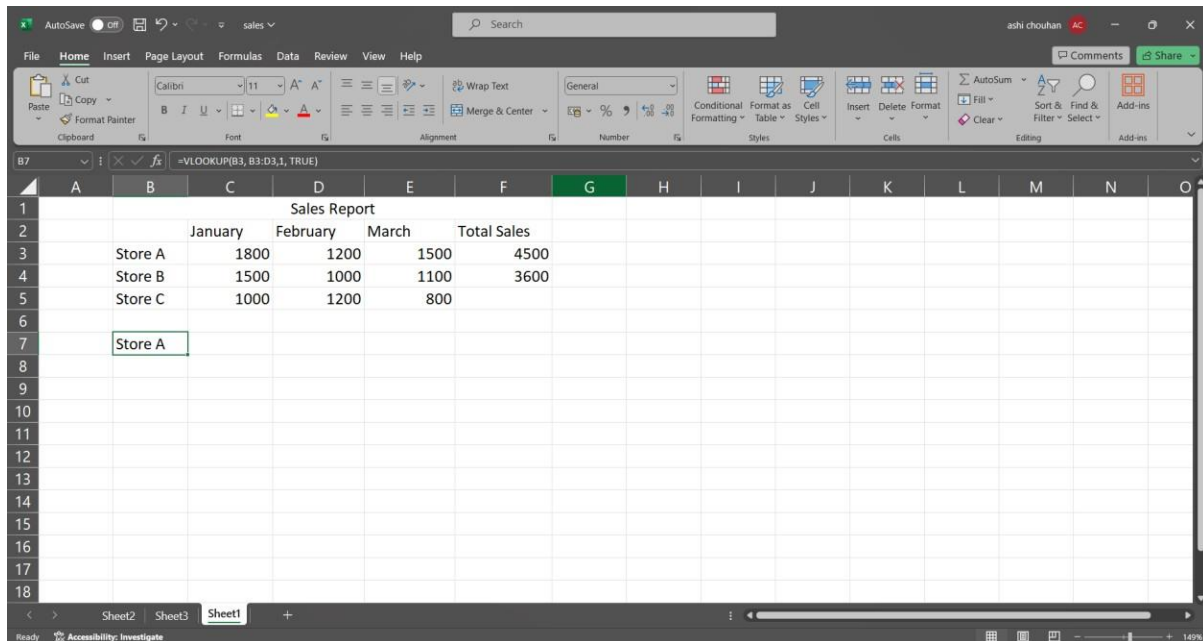
A. Use VLOOKUP function to retrieve information from a different worksheet or table.Steps:

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

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

For video demonstration of the practical click on the below link:

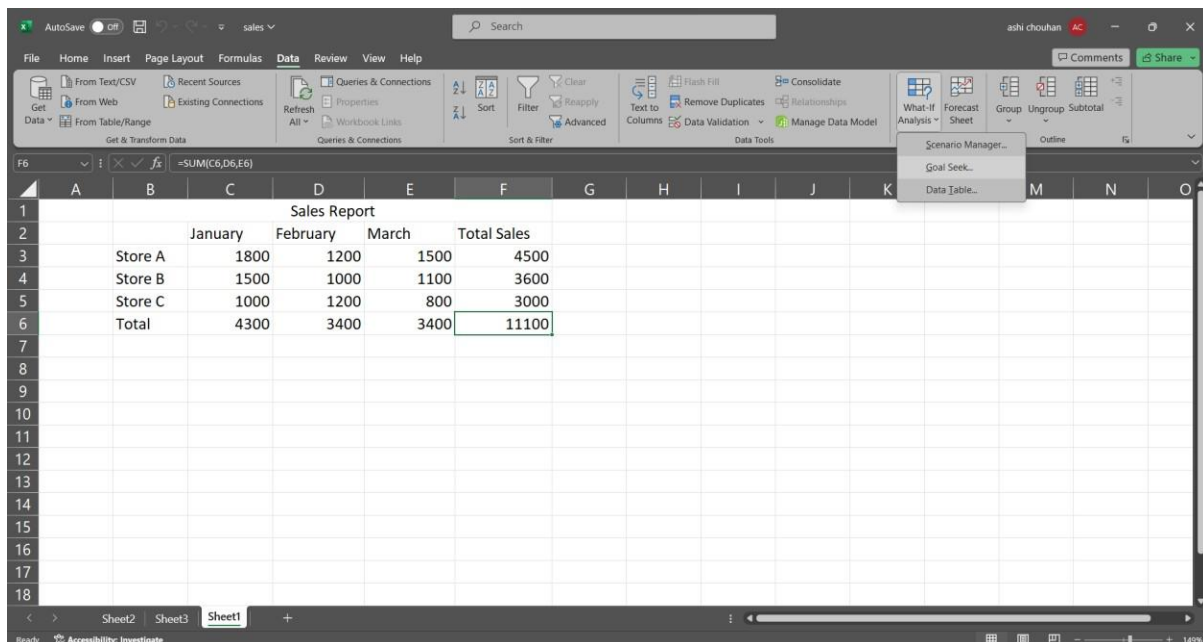
[Data Science Practical Playlist](#)



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

Steps-

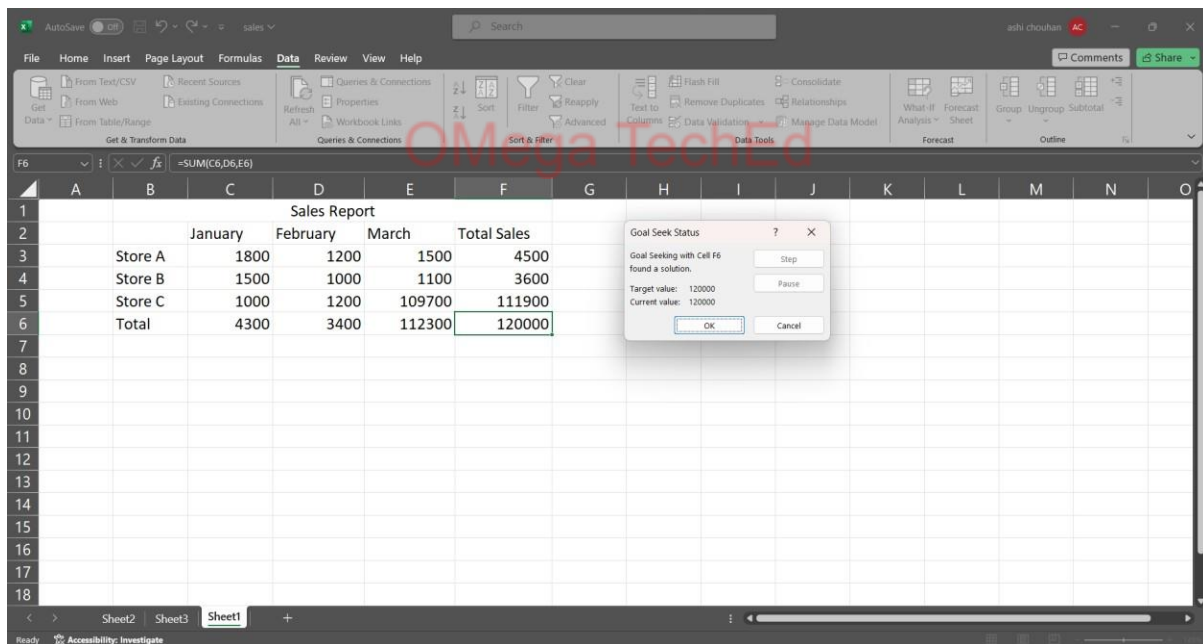
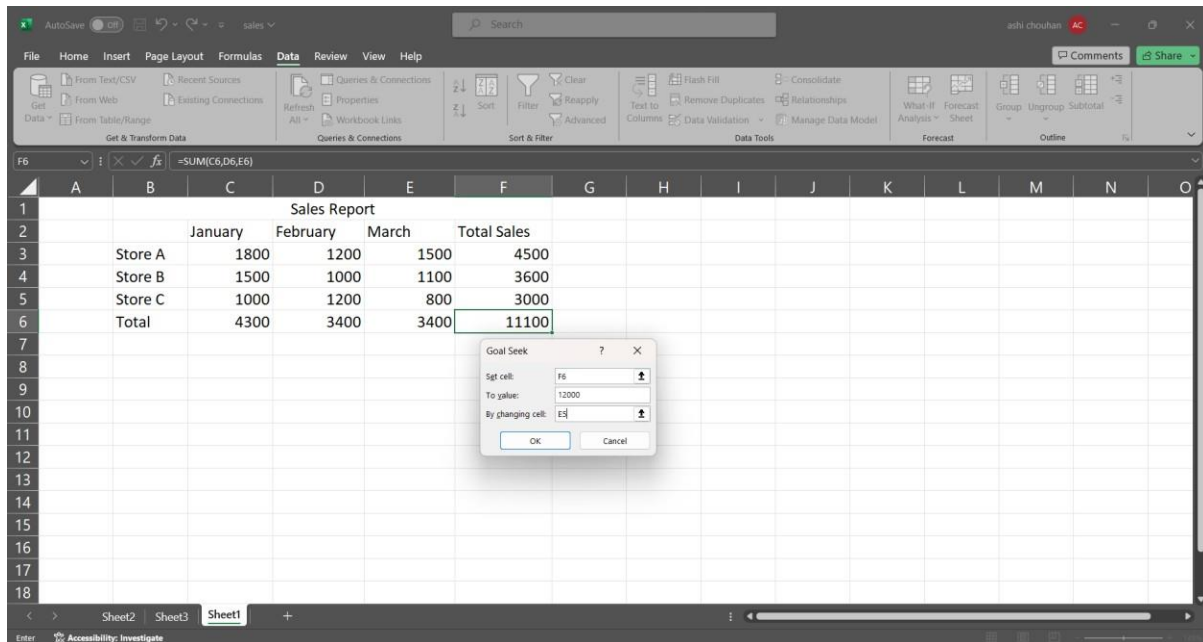
Step 1: In the Data tab go to the what if analysis>Goal seek.



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

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)



For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

The screenshot shows a Microsoft Excel window with a 'Sales Report' table. The table has columns for months (January, February, March) and a 'Total Sales' column. The data is as follows:

	January	February	March	Total Sales
Store A	1800	1200	1500	4500
Store B	1500	1000	1100	3600
Store C	1000	1200	109700	111900
Total	4300	3400	112300	120000

The formula bar shows the formula for the total sales: `=SUM(C5,D5,E5)`.

OMega TechEd

PRACTICAL 2

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
fileimport pandas as
pd
df =
pd.read_csv('Student_Marks.csv')
print("Our dataset ")
print(df)
```

```
===== RESTART: D:\Notes\sem-6\data science\prac2
Our dataset
   number_courses  time_study  Marks
0                3      4.508  19.202
1                4      0.096   7.734
2                4      3.133  13.811
3                6      7.909  53.018
4                8      7.811  55.299
..             ...         ...    ...
95               6      3.561  19.128
96               3      0.301   5.609
97               4      7.163  41.444
98               7      0.309  12.027
99               3      6.335  32.357

[100 rows x 3 columns]
>>>
```

(2)

```
# Reading data from a
JSON fileimport pandas as
pd
data =
pd.read_json('dataset.json')
print(data)
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

http://www.youtube.com/@omega_teched

```
>>> ===== 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)
```

OMega TechEd

```
===== RESTART: D:/Notes/sem-6/data science/prac2c.py =====
      PassengerId  Survived  ... Cabin Embarked
0              892         0  ...   NaN        Q
1              893         0  ...   NaN        S
2              894         0  ...   NaN        Q
3              895         0  ...   NaN        S
4              896         0  ...   NaN        S
..          ...          ...  ...   ...        ...
413            1305         0  ...   NaN        S
414            1306         0  ...  C105        C
415            1307         0  ...   NaN        S
416            1308         0  ...   NaN        S
417            1309         0  ...   NaN        C

[418 rows x 11 columns]
Dataset after filling NA values with 0 :
      PassengerId  Survived  ... Cabin Embarked
0              892         0  ...     0        Q
1              893         0  ...     0        S
2              894         0  ...     0        Q
3              895         0  ...     0        S
4              896         0  ...     0        S
..          ...          ...  ...   ...        ...
413            1305         0  ...     0        S
414            1306         0  ...  C105        C
415            1307         0  ...     0        S
416            1308         0  ...     0        S
417            1309         0  ...     0        C

[418 rows x 11 columns]
.>>>
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

(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)
```

OMega TechEd

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

```

===== RESTART: D:/Notes/sem-6/data science/prac2c.py =====
   PassengerId  Survived  Pclass    Cabin Embarked
0            892         3.0    NaN      Q
1            893         3.0    NaN      S
2            894         2.0    NaN      Q
3            895         3.0    NaN      S
4            896         NaN    NaN      S
..          ...         ...    ...    ...
413          1305         3.0    NaN      S
414          1306         1.0    C105      C
415          1307         3.0    NaN      S
416          1308         3.0    NaN      S
417          1309         3.0    NaN      C

[418 rows x 11 columns]
Dataset after dropping NA values:
   PassengerId  Survived  Pclass    Cabin Embarked
12            904         1.0    B45      S
14            906         1.0    E31      S
24            916         1.0    B57 B59 B63 B66      C
26            918         1.0    B36      C
28            920         1.0    A21      S
..          ...         ...    ...    ...
404           1296         1.0    D40      C
405           1297         2.0    D38      C
407           1299         1.0    C80      C
411           1303         1.0    C78      Q
414           1306         1.0    C105      C

[87 rows x 11 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
```

```
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:")
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

```
print(sorted_iris.head())
```

```
# Grouping data
```

```
grouped_species =
```

```
iris.groupby('Species').mean()
```

```
print("\nMean measurements for each
```

```
species:") print(grouped_species)
```

```
===== RESTART: D:/Notes/sem-6/data science/prac2b.py =====
Setosa samples:
Empty DataFrame
Columns: [Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species]
Index: []

Sorted iris dataset:
   Id  SepalLengthCm  ...  PetalWidthCm  Species
131  132           7.9  ...           2.0  Iris-virginica
135  136           7.7  ...           2.3  Iris-virginica
122  123           7.7  ...           2.0  Iris-virginica
117  118           7.7  ...           2.2  Iris-virginica
118  119           7.7  ...           2.3  Iris-virginica

[5 rows x 6 columns]

Mean measurements for each species:
   Id  SepalLengthCm  ...  PetalLengthCm  PetalWidthCm
Species
Iris-setosa      25.5           5.006  ...           1.464           0.244
Iris-versicolor  75.5           5.936  ...           4.260           1.326
Iris-virginica   125.5           6.588  ...           5.552           2.026

[3 rows x 5 columns]
>>
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

PRACTICAL 3

Feature Scaling and Dummification

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

Code:

```
# Standardization and
normalizationimport pandas as
pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, StandardScaler
df = pd.read_csv('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)
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

```
= RESTART: D:/Notes/sem-6/data science/prac3b.py
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
173         3  0.305130  0.305130
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]
>>>
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

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

Code:

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

```
===== RESTART: D:/Notes/sem-6/data science/prac3a.py =====
   Id  SepalLengthCm  ...  PetalWidthCm  Species
0    1             5.1  ...           0.2  Iris-setosa
1    2             4.9  ...           0.2  Iris-setosa
2    3             4.7  ...           0.2  Iris-setosa
3    4             4.6  ...           0.2  Iris-setosa
4    5             5.0  ...           0.2  Iris-setosa
..  ...           ...  ...           ...  ...
145 146             6.7  ...           2.3  Iris-virginica
146 147             6.3  ...           1.9  Iris-virginica
147 148             6.5  ...           2.0  Iris-virginica
148 149             6.2  ...           2.3  Iris-virginica
149 150             5.9  ...           1.8  Iris-virginica

[150 rows x 6 columns]
   Id  SepalLengthCm  SepalWidthCm  ...  PetalWidthCm  Species  code
0    1             5.1           3.5  ...           0.2  Iris-setosa    0
1    2             4.9           3.0  ...           0.2  Iris-setosa    0
2    3             4.7           3.2  ...           0.2  Iris-setosa    0
3    4             4.6           3.1  ...           0.2  Iris-setosa    0
4    5             5.0           3.6  ...           0.2  Iris-setosa    0
..  ...           ...           ...  ...           ...  ...  ...
145 146             6.7           3.0  ...           2.3  Iris-virginica    2
146 147             6.3           2.5  ...           1.9  Iris-virginica    2
147 148             6.5           3.0  ...           2.0  Iris-virginica    2
148 149             6.2           3.4  ...           2.3  Iris-virginica    2
149 150             5.9           3.0  ...           1.8  Iris-virginica    2

[150 rows x 7 columns]
>>>
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Practical 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
```

```
purposesnp.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
```

```
levelalpha = 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}")
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)


```
# 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')
```

Show the plot

plt.show()

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 Sample 2.")

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 Sample 1.")

1.")

else:

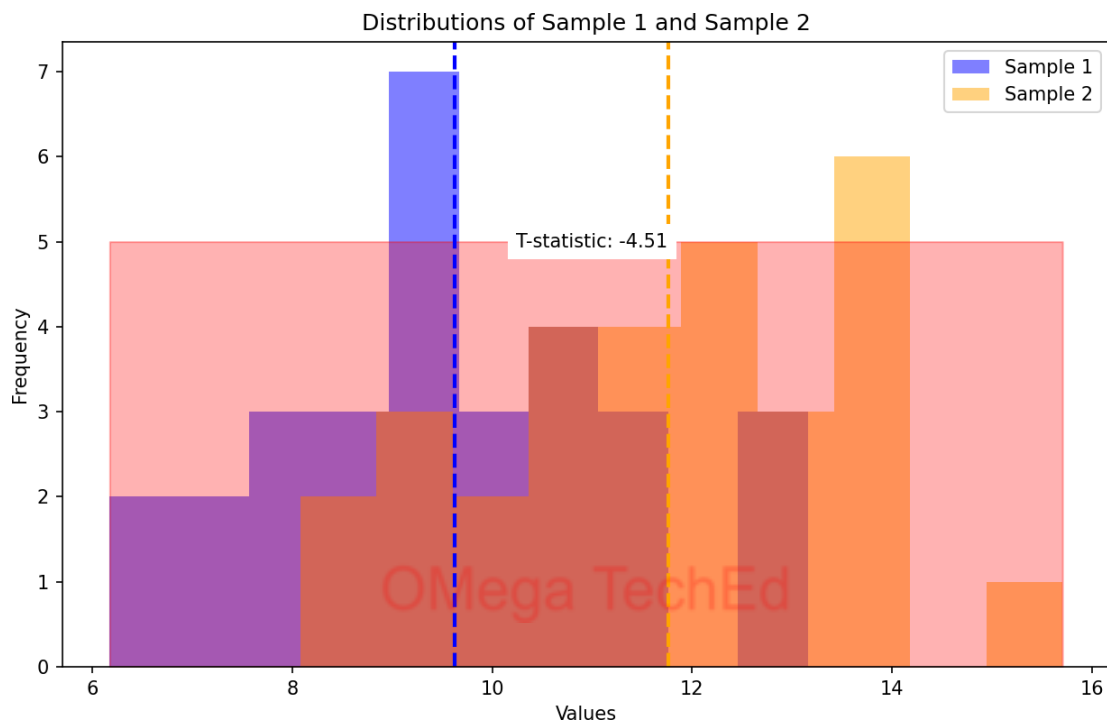
OMega TechEd

print("Conclusion: Fail to reject the null hypothesis.")

print("Interpretation: There is not enough evidence to claim a significant difference between the means.")

```
----- RESULTS: P-VALUE NOT
Results of Two-Sample t-test:
T-statistic: -4.512913234547555
P-value: 3.176506547470154e-05
Degrees of Freedom: 58
```

Output:



#chi-test

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib as
```

```
pltimport seaborn
```

```
as sb import
```

```
warnings
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

```
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))
```

Output:

```

----- RESTART: E:/all notes/DS/prac_4.1.py -----
      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

```

OMega TechEd

```

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

```

OMega TechEd

```

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
(54.95485392447537, 3.320518009555984e-11, 4, array([[ 21.21428571,  16.66836735,  61.11734694]
,
[ 53.14285714,  41.75510204, 153.10204082],
[  9.64285714,   7.57653061,  27.78061224]])

```

Conclusion:

There is sufficient evidence to reject the null hypothesis, indicating that there is a significant association between 'horsepower_new' and 'modelyear_new' categories.

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Practical 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

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]

# Combine data into a DataFrame
data = pd.DataFrame({'value': group1 + group2 + group3 + group4,
                    'group': ['Group1'] * len(group1) + ['Group2'] * len(group2) +
                        ['Group3'] * len(group3) + ['Group4'] * len(group4)})

# Perform one-way ANOVA
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)

# Perform Tukey-Kramer post-hoc test
tukey_results = pairwise_tukeyhsd(data['value'], data['group'])
print("\nTukey-Kramer post-hoc test:")
print(tukey_results)
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Output:-

```

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
=====

```

Conclusion

- **F-statistic:** This value indicates the ratio of the variance between groups to the variance within groups. A larger F-statistic suggests that the means of the groups are more different from each other compared to the variability within each group.
- If the p-value is less than the chosen significance level (e.g., 0.05), it suggests that there are significant differences among the group means.
- If the p-value is greater than the significance level, it suggests that there is insufficient evidence to reject the null hypothesis, meaning there are no significant differences among the group means.
- ***There are significant differences among the means of the groups.***

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Practical 6

Regression and its Types.

```
import numpy as
npimport pandas
as pd
from sklearn.datasets import
fetch_california_housingfrom
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()
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

http://www.youtube.com/@omega_teched

```
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_)
```

OMega TechEd

http://www.youtube.com/@omega_teched

output:

```

===
      MedInc HouseAge AveRooms ... AveOccup Latitude Longitude
0      8.3252    41.0  6.984127 ...  2.555556    37.88   -122.23
1      8.3014    21.0  6.238137 ...  2.109842    37.86   -122.22
2      7.2574    52.0  8.288136 ...  2.802260    37.85   -122.24
3      5.6431    52.0  5.817352 ...  2.547945    37.85   -122.25
4      3.8462    52.0  6.281853 ...  2.181467    37.85   -122.25
...      ...      ...      ...      ...      ...
20635  1.5603    25.0  5.045455 ...  2.560606    39.48   -121.09
20636  2.5568    18.0  6.114035 ...  3.122807    39.49   -121.21
20637  1.7000    17.0  5.205543 ...  2.325635    39.43   -121.22
20638  1.8672    18.0  5.329513 ...  2.123209    39.43   -121.32
20639  2.3886    16.0  5.254717 ...  2.616981    39.37   -121.24

[20640 rows x 8 columns]
Mean Squared Error: 1.2923314440807299
R-squared: 0.013795337532284901
Intercept: 1.654762268596842
Coefficient: [0.07675559]
->

```

#####

OMega TechEd

#Multiple Liner 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)

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

http://www.youtube.com/@omega_teched

```
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.interc  
ept_)  
print("Coefficient:",model.co  
ef_)
```

OMega TechEd

Output:

```

----- REGRESSOR: E:\all notes\DS\prac_0_single.py -----
      MedInc  HouseAge  AveRooms  ...  AveOccup  Latitude  Longitude
0      8.3252      41.0  6.984127  ...  2.555556      37.88      -122.23
1      8.3014      21.0  6.238137  ...  2.109842      37.86      -122.22
2      7.2574      52.0  8.288136  ...  2.802260      37.85      -122.24
3      5.6431      52.0  5.817352  ...  2.547945      37.85      -122.25
4      3.8462      52.0  6.281853  ...  2.181467      37.85      -122.25
...      ...      ...      ...      ...      ...      ...
20635  1.5603      25.0  5.045455  ...  2.560606      39.48      -121.09
20636  2.5568      18.0  6.114035  ...  3.122807      39.49      -121.21
20637  1.7000      17.0  5.205543  ...  2.325635      39.43      -121.22
20638  1.8672      18.0  5.329513  ...  2.123209      39.43      -121.32
20639  2.3886      16.0  5.254717  ...  2.616981      39.37      -121.24

[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]

```

Conclusion

OMega TechEd

The Mean Squared Error (MSE) is a commonly used metric to evaluate the performance of regression models. It measures the average squared difference between the predicted values and the actual values of the target variable. A lower MSE value indicates that the model's predictions are closer to the actual values on average, suggesting better performance.

R^2 tells us how well the independent variables explain the variability of the dependent variable. It ranges from 0 to 1, where: 0 indicates that the model does not explain any of the variability of the dependent variable around its mean. 1 indicates that the model explains all the variability of the dependent variable around its mean.

The intercept represents the point where the regression line intersects the y-axis on a graph. It provides information about the baseline value of the dependent variable when all predictors are zero.

Coefficients represent the impact of changes in the independent variables on the dependent variable in a linear regression model.

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Practical 7**Logistic Regression and Decision Tree**

```

import numpy as
npimport pandas
as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import
train_test_split from sklearn.linear_model
import LogisticRegressionfrom 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 problemiris = 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)

```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

http://www.youtube.com/@omega_teched

```
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
performancedecision_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))
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Output:-

Logistic Regression Metrics

Accuracy: 1.0

Precision: 1.0

Recall: 1.0

Classification Report

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	12
1.0	1.00	1.00	1.00	8
accuracy			1.00	20
macro avg	1.00	1.00	1.00	20
weighted avg	1.00	1.00	1.00	20

Decision Tree Metrics

Accuracy: 1.0

Precision: 1.0

Recall: 1.0

Classification Report

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	12
1.0	1.00	1.00	1.00	8
accuracy			1.00	20
macro avg	1.00	1.00	1.00	20
weighted avg	1.00	1.00	1.00	20

Conclusion:

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Precision: Precision measures the ratio of correctly predicted positive observations to the total predicted positives. It indicates the accuracy of positive predictions made by the model. A higher precision indicates fewer false positives. Ideally, precision should be as close to 1.0 as possible. A precision of 1.0 indicates that all positive predictions made by the model are correct, with no false positives.

Recall (also called Sensitivity or True Positive Rate): Recall measures the ratio of correctly predicted positive observations to all actual positives in the dataset. It indicates the model's ability to identify all positive instances correctly. A higher recall indicates fewer false negatives. An ideal recall score would also be 1.0. A recall of 1.0 indicates that the model correctly identifies all positive instances in the dataset, with no false negatives.

F1-score: The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall. F1-score reaches its best value at 1 and worst at 0. It provides a balance between precision and recall. A higher F1-score indicates better overall performance, especially in scenarios where we want to balance false positives and false negatives.

Support: Support is the number of actual occurrences of the class in the specified dataset. It represents the number of samples in each class. The support value for each class ideally reflects a well-distributed dataset, with enough samples for each class.

OMega TechEd

Practical 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("C:\\Users\\Reape\\Downloads\\wholesale\\wholesale.c
sv")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)
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

```
sum_of_squared_distances
```

```
= []K = range(1, 15)
```

```
for k in K:
```

```
    km =
```

```
        KMeans(n_clusters=k)
```

```
    km =
```

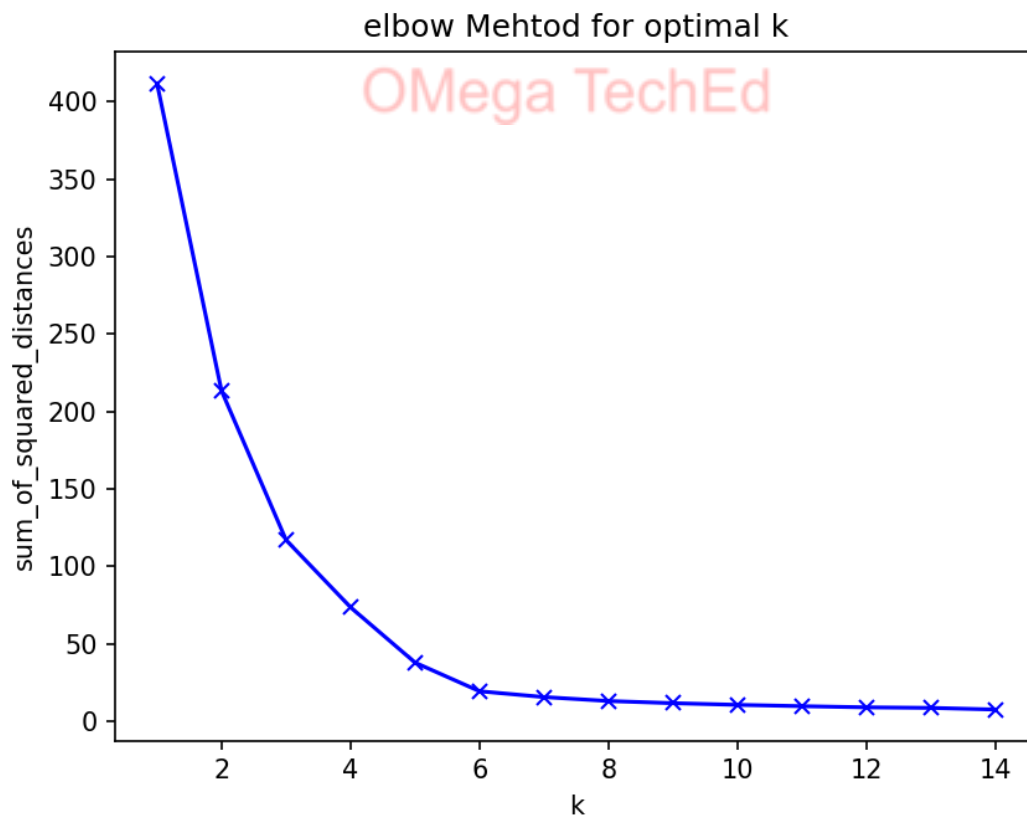
```
        km.fit(data_transformed)
```

OMega TechEd

```
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()
```

Output:



For video demonstration of the practical click on the below link:
[Data Science Practical Playlist](#)

Conclusion:

- The elbow method helps in determining the optimal number of clusters for the dataset. The point where the rate of decrease in the sum of squared distances significantly slows down suggests a suitable number of clusters.
- We conclude that the optimal number of clusters for the data is 5.
- The optimal number of clusters identified using this method can be used for further analysis or segmentation of customers based on their purchasing behavior.

OMega TechEd

Practical 9

Principal Component Analysis (PCA)

```
import pandas as
pdimport numpy
as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import
StandardScalerfrom
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_
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

```
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}")
```

OMega TechEd

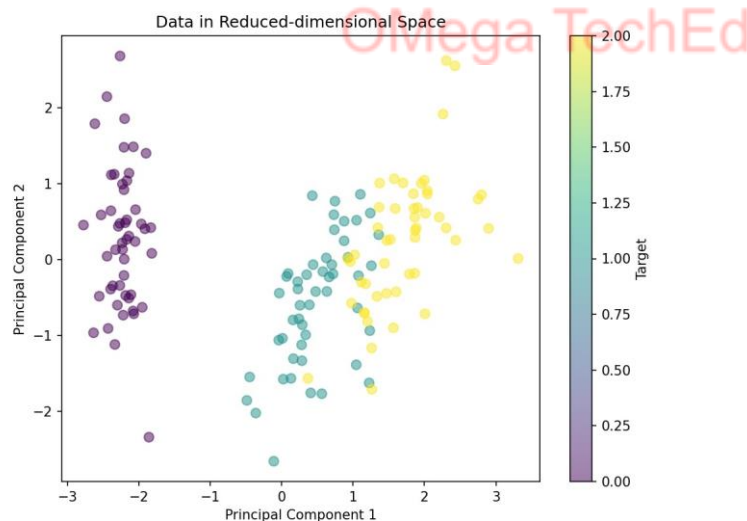
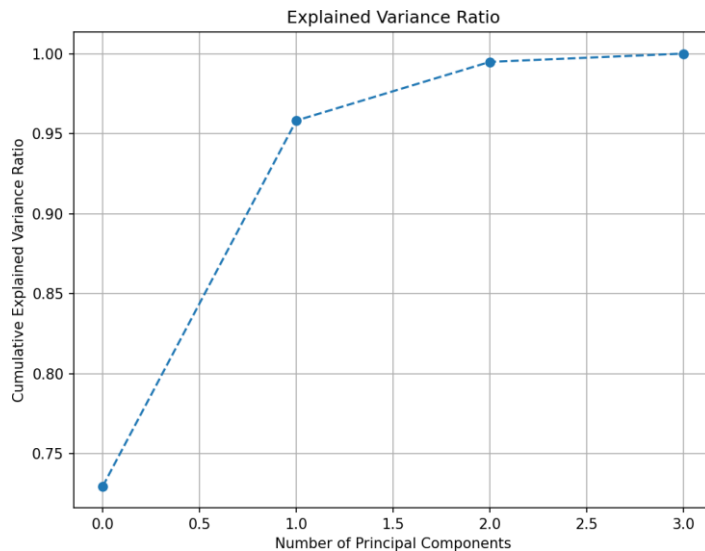
```
pca =
PCA(n_components=n_components)
X_reduced =
pca.fit_transform(X_scaled)
```

```
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()
```

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Output:



----- RESTART. E./all notes/DS/PL
Number of principal components to explain 95% variance: 2

Conclusion:

In conclusion, the code demonstrates the effectiveness of PCA in reducing the dimensionality of high-dimensional datasets while preserving essential information. It provides a systematic approach to exploring and visualizing complex datasets, thereby aiding in data analysis and interpretation.

For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

Practical 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")
```

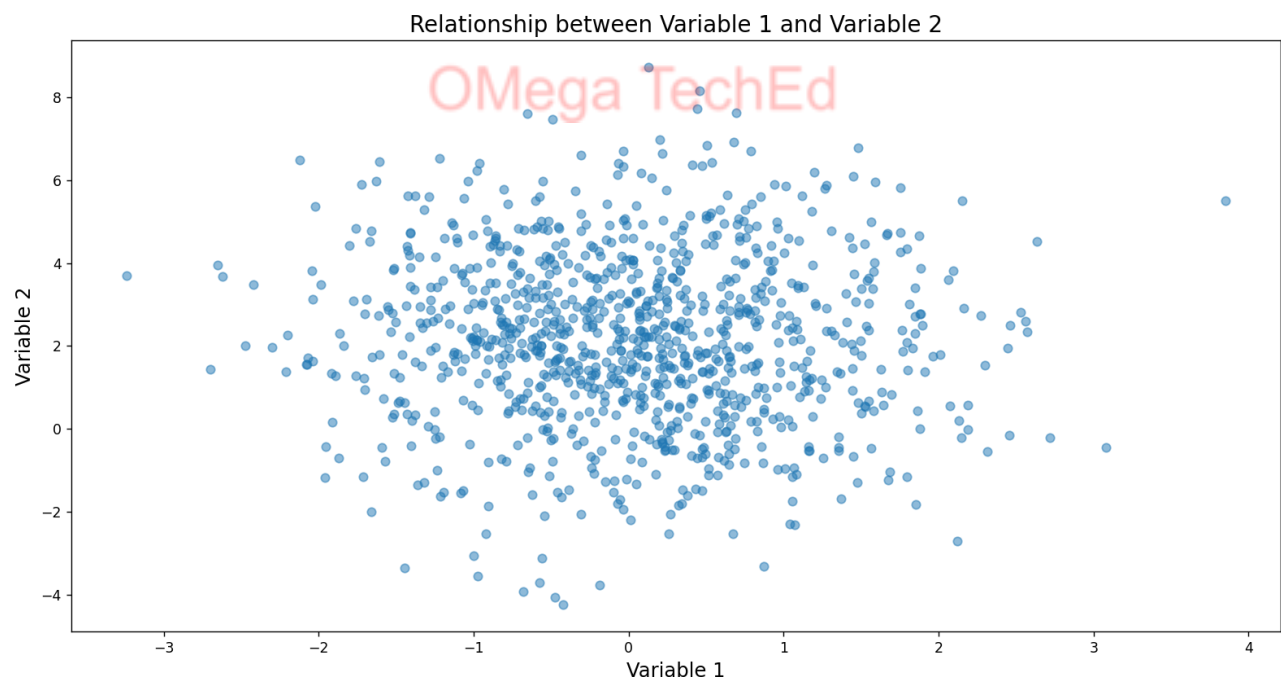
For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

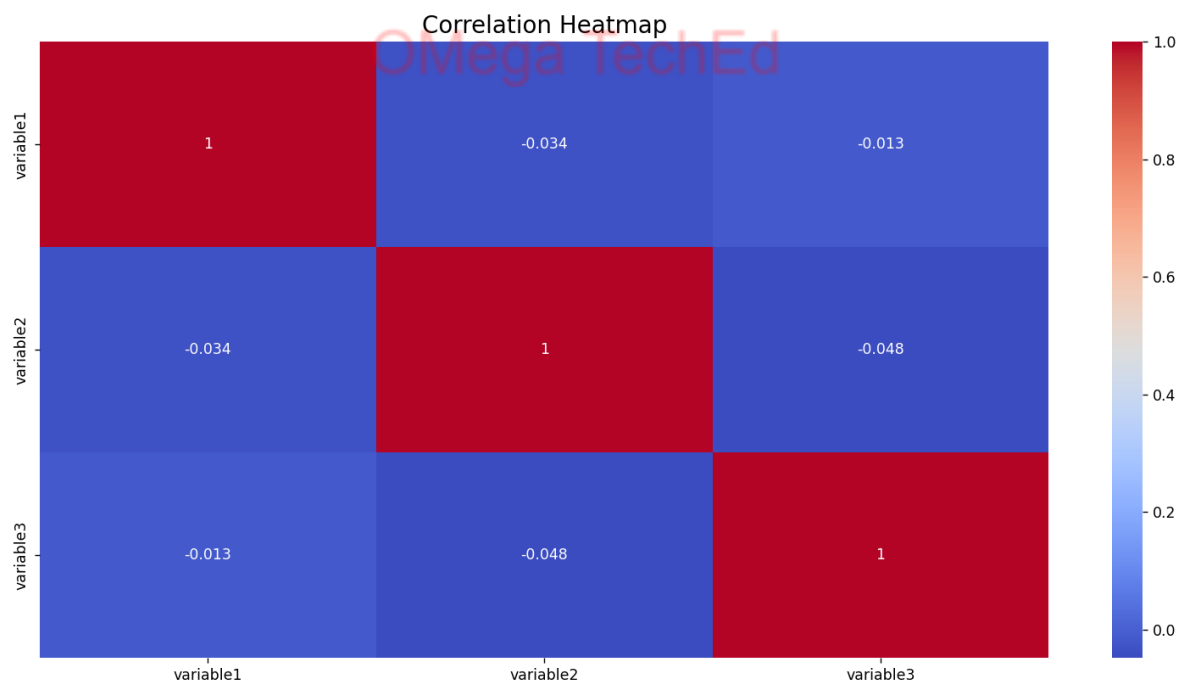
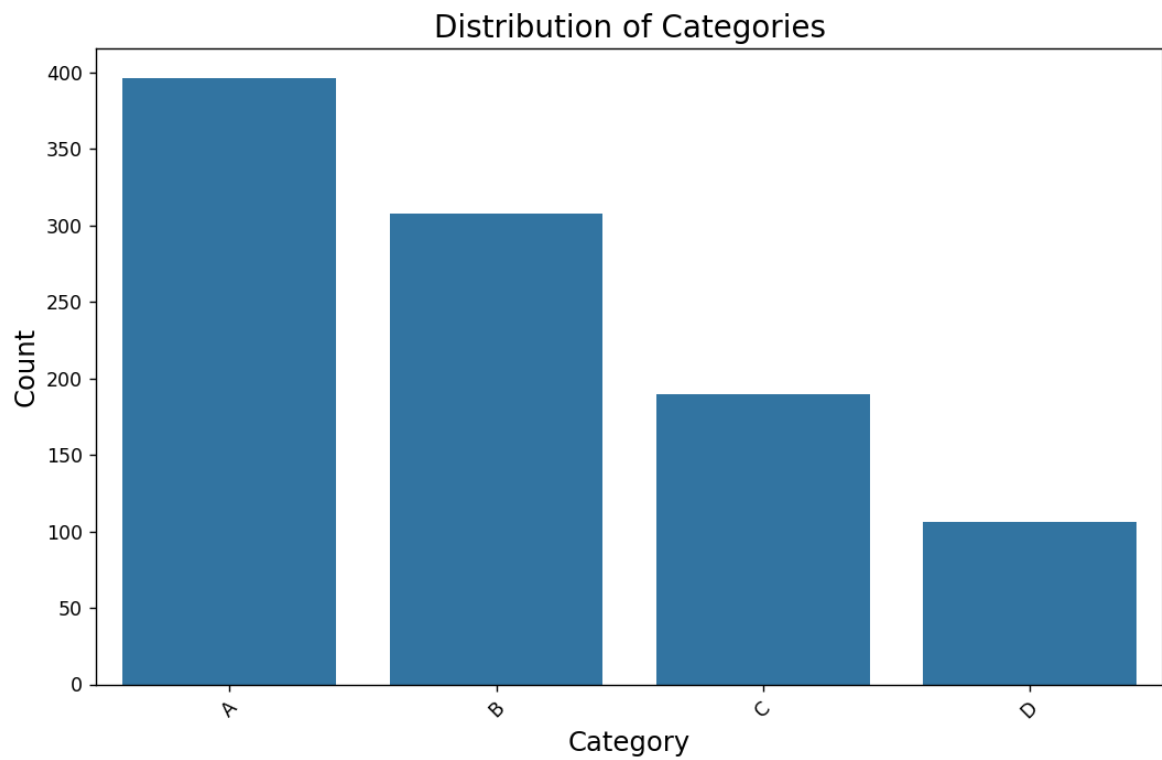
http://www.youtube.com/@omega_teched

```
print("\nThe scatter plot (Figure 1) shows the relationship between Variable 1 and  
Variable 2. ")  
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 ")  
print("\nBar Chart")  
print("Figure 2: Distribution of Categories")  
print("\nAdditionally, we explored the correlation between numerical variables using  
a heatmap ")  
print("\nHeatmap")  
print("Figure 3: Correlation Heatmap")  
  
print("\nIn summary, the visualizations and analysis provide insights into the  
relationships ")
```

Output:



For video demonstration of the practical click on the below link:
[Data Science Practical Playlist](#)



For video demonstration of the practical click on the below link:

[Data Science Practical Playlist](#)

http://www.youtube.com/@omega_teched

= RESTART: D:/pract 10.py

Title: Exploring the Relationship between Variable 1 and Variable 2

The scatter plot (Figure 1) shows the relationship between Variable 1 and Variable 2.

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: Distribution of Categories

Additionally, we explored the correlation between numerical variables using a heatmap

Heatmap

Figure 3: Correlation Heatmap

In summary, the visualizations and analysis provide insights into the relationships

OMega TechEd