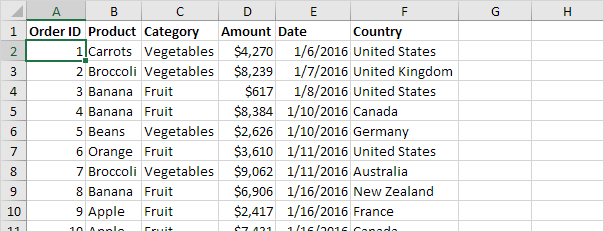
Roll no. 09 practical no.1 Date: 09/01/24

AIM: Introduction to Excel

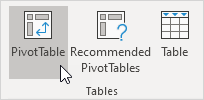
* Perform Conditional Formatting on a dataset using various criteria

Our data set consists of 213 records and 6 fields. Order ID, Product, Category, Amount, Date and Country.

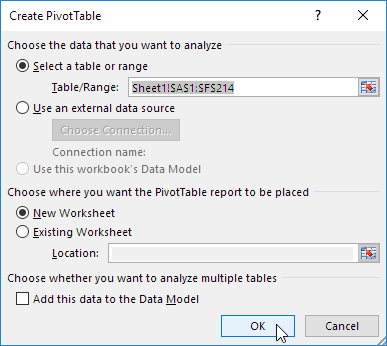
# Insert a Pivot Table

To insert a pivot table, execute the following steps.

1. Click any single cell inside the data set.
2. On the Insert tab, in the Tables group, click PivotTable.

The following dialog box appears. Excel automatically selects the data for you. The default location for a new pivot table is New Worksheet.

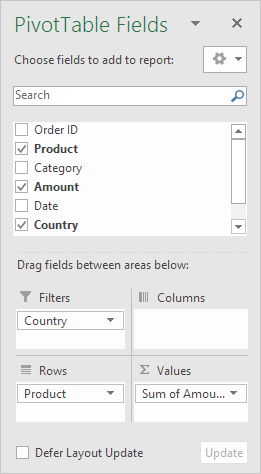
1. Click OK.

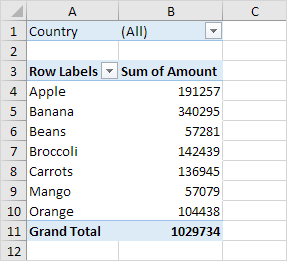


# Drag fields

The PivotTable Fields pane appears. To get the total amount exported of each product, drag the following fields to the different areas.

1. Product field to the Rows area.
2. Amount field to the Values area.
3. Country field to the Filters area.

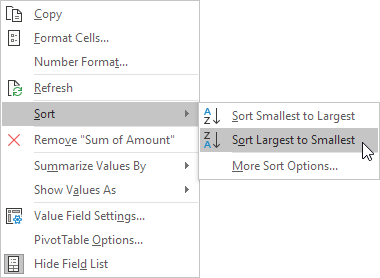


Below you can find the pivot table. Bananas are our main export product. That's how easy pivot tables can be!

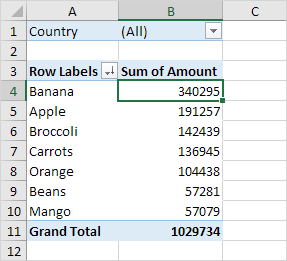
# Sort a Pivot Table

To get Banana at the top of the list, sort the pivot table.

1. Click any cell inside the Sum of Amount column.
2. click and click on Sort, Sort Largest to Smallest.



Result.

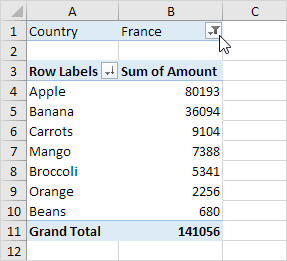


# Filter

Because we added the Country field to the Filters area, we can filter this pivot table by Country. For example, which products do we export the most to France?

1. Click the filter drop-down and select France.

Result. Apples are our main export product to France.

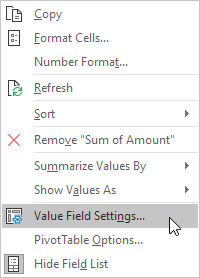


Note: you can use the standard filter (triangle next to Row Labels) to only show the amounts of specific products.

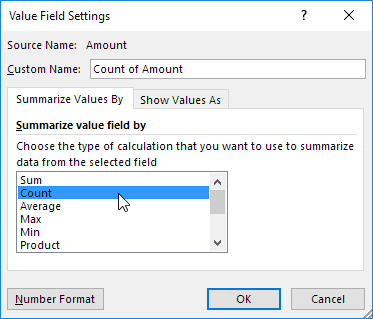
# Change Summary Calculation

By default, Excel summarizes your data by either summing or counting the items. To change the type of calculation that you want to use, execute the following steps.

1. Click any cell inside the Sum of Amount column.
2. Right click and click on Value Field Settings.

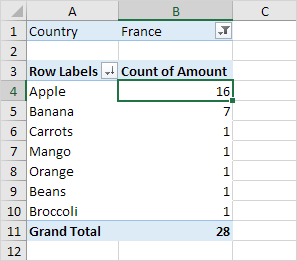


1. Choose the type of calculation you want to use. For example, click Count.



1. Click OK.

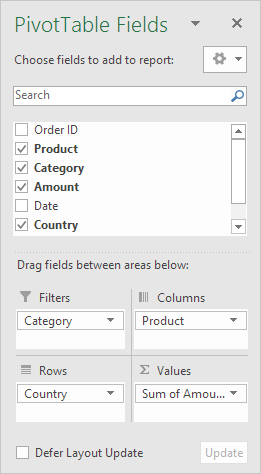
Result. 16 out of the 28 orders to France were 'Apple' orders.



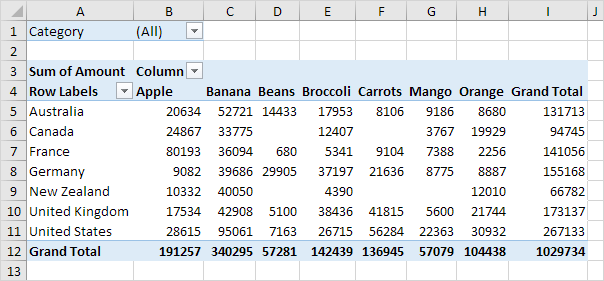
# Two-dimensional Pivot Table

If you drag a field to the Rows area and Columns area, you can create a two-dimensional pivot table. First, [insert a pivot table](https://www.excel-easy.com/data-analysis/pivot-tables.html). Next, to get the total amount exported to each country, of each product, drag the following fields to the different areas.

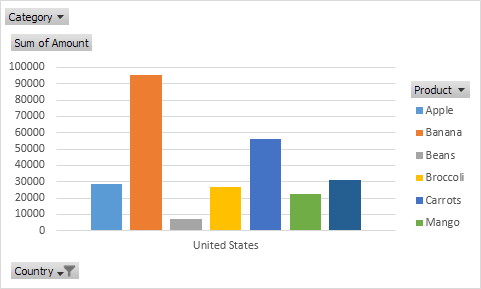
1. Country field to the Rows area.
2. Product field to the Columns area.
3. Amount field to the Values area.
4. Category field to the Filters area.



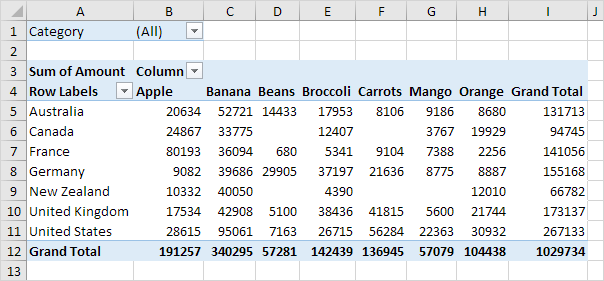
Below you can find the two-dimensional pivot table.



To easily compare these numbers, create a [pivot chart](https://www.excel-easy.com/examples/pivot-chart.html) and apply a filter. Maybe this is one step too far for you at this stage, but it shows you one of the many other powerful pivot table features Excel has to offer.



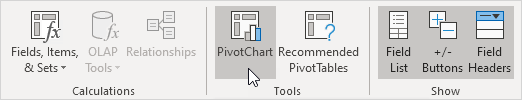
A pivot chart is the visual representation of a pivot table in Excel. Pivot charts a



# **Create a Pivot table to analyze and summarize data**

To insert a pivot chart, execute the following steps.

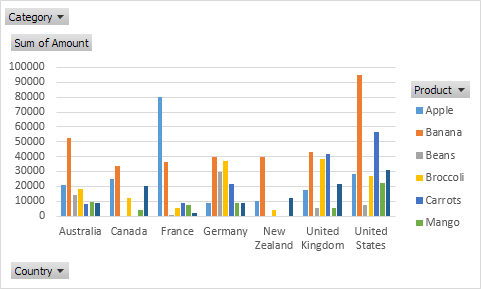
1. Click any cell inside the pivot table.
2. On the PivotTable Analyze tab, in the Tools group, click PivotChart.



The Insert Chart dialog box appears.

1. Click OK.

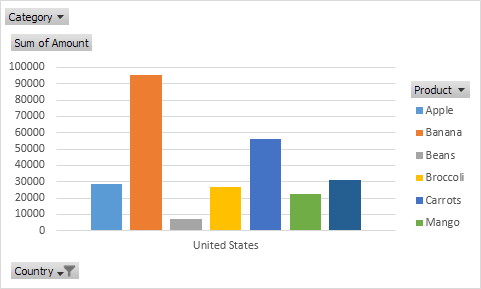
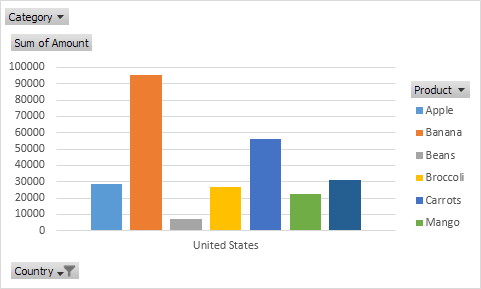
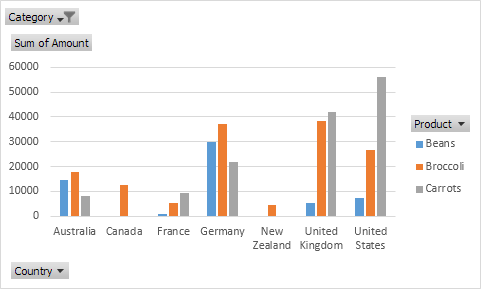
Below you can find the pivot chart. This pivot chart will amaze and impress your boss.



Note: any changes you make to the pivot chart are immediately reflected in the pivot table and vice versa.

# Filter Pivot Chart

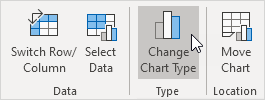
To filter this pivot chart, execute the following steps.

1. Use the standard filters (triangles next to Product and Country). For example, use the Country filter to only show the total amount of each product exported to the United States.
2. Remove the Country filter.
3. Because we added the Category field to the Filters area, we can filter this pivot chart (and pivot table) by Category. For example, use the Category filter to only show the vegetables exported to each country.

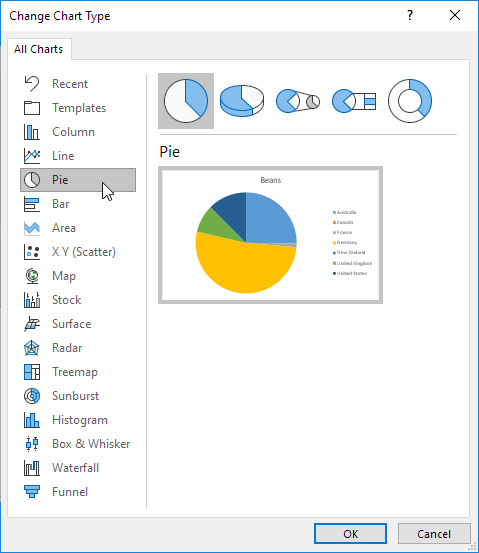
# Change Pivot Chart Type

You can change to a different type of pivot chart at any time.

1. Select the chart.
2. On the Design tab, in the Type group, click Change Chart Type.

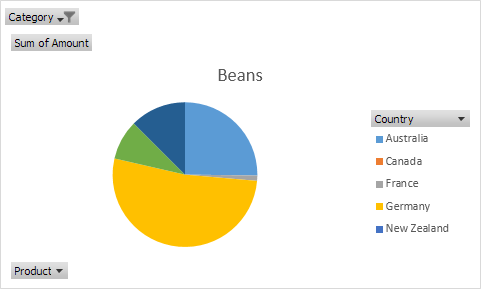


1. Choose Pie.

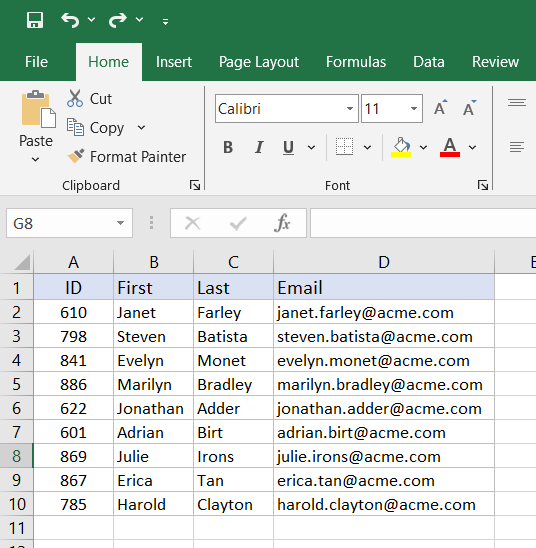


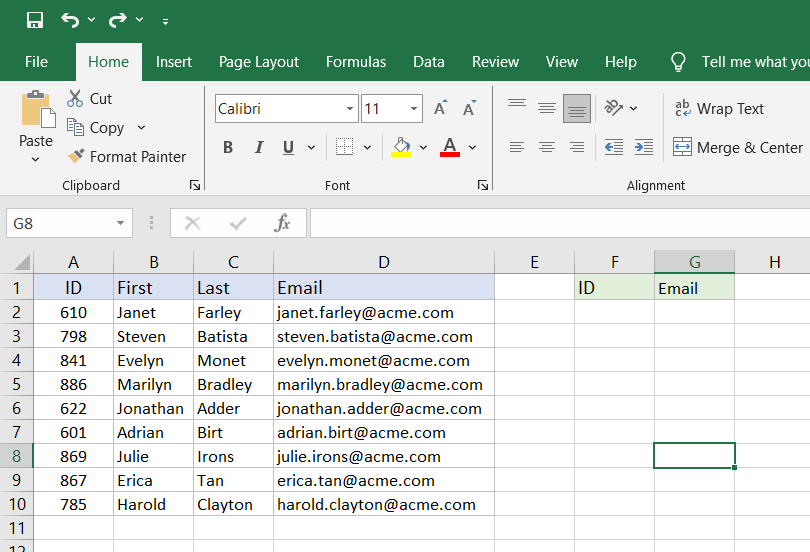
1. Click OK.

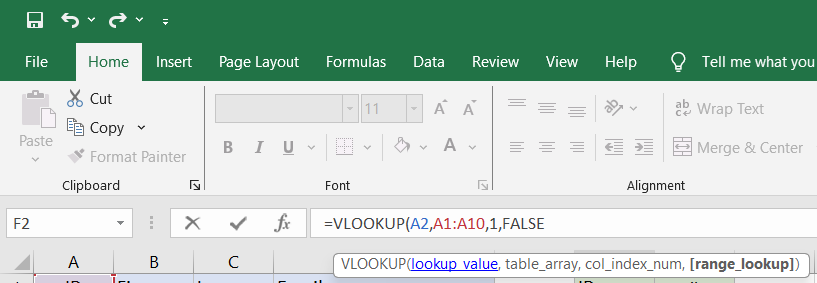
Result:

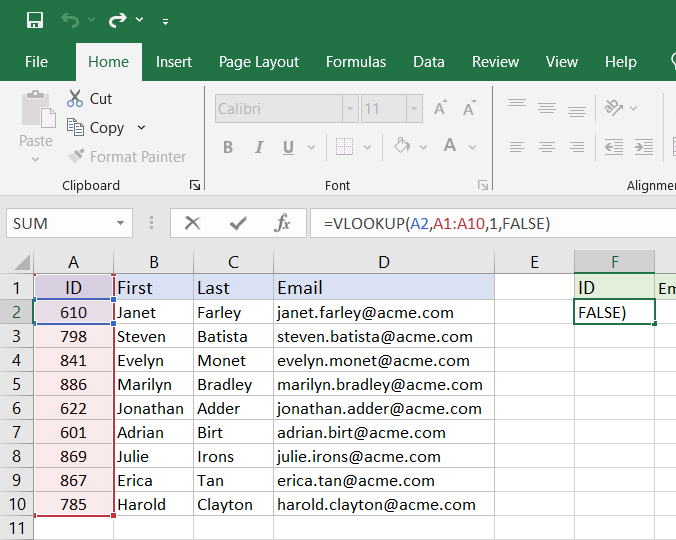


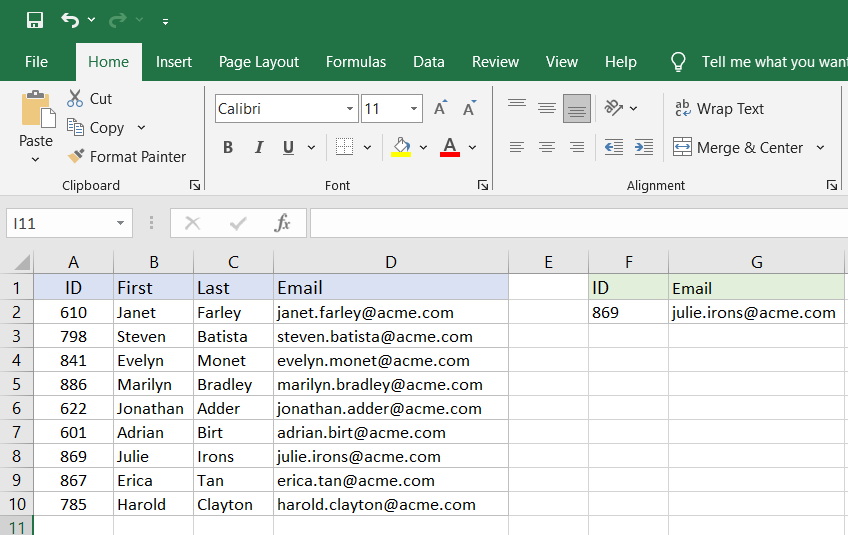
* **Use VLOOKUP function to retrieve information from a different worksheet or table**



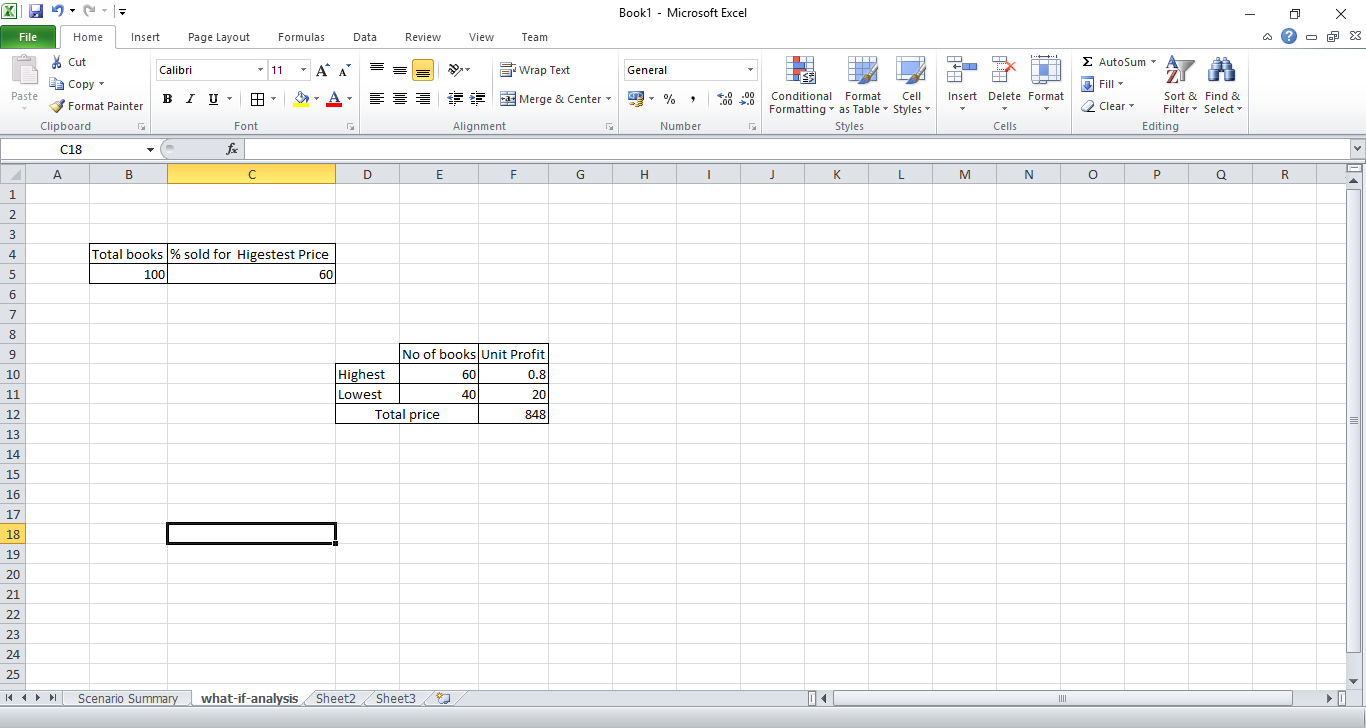


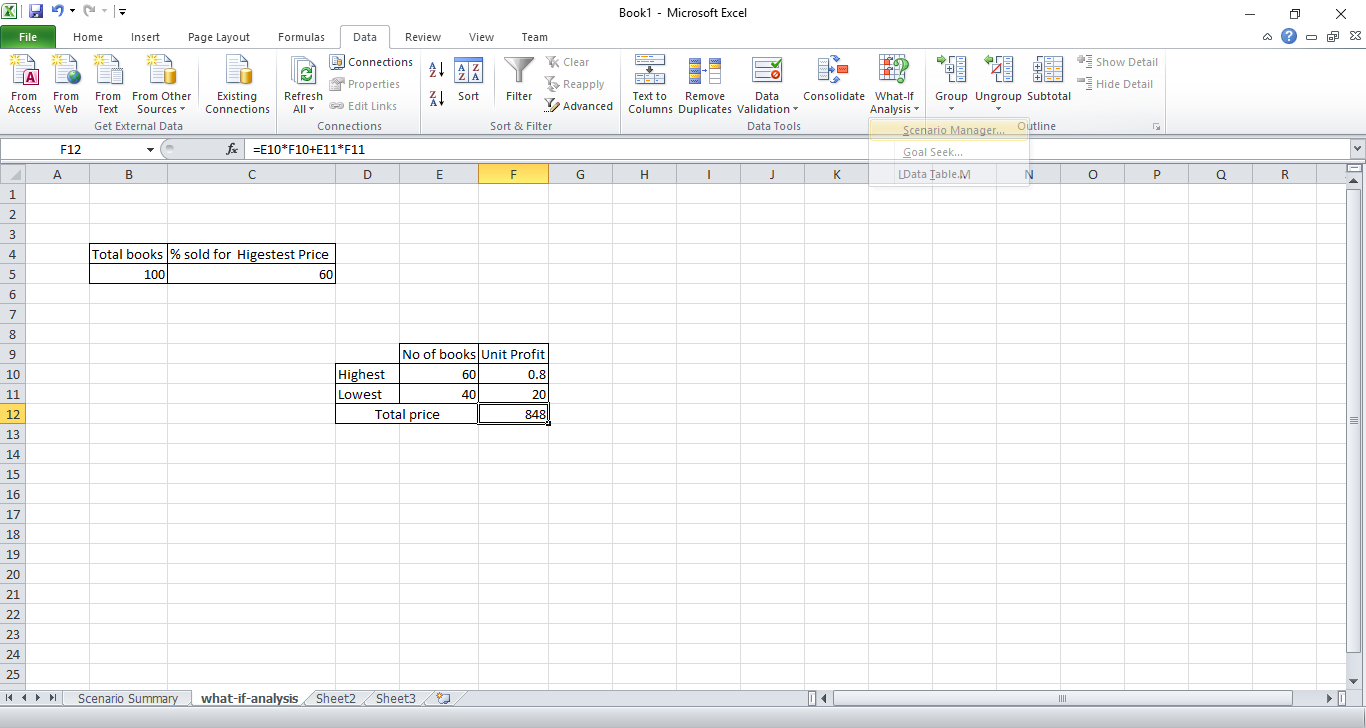




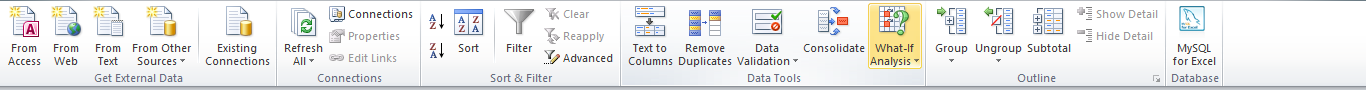


* **What If Analysis using Goal Seek to determine input values for desired output**

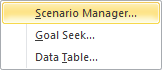




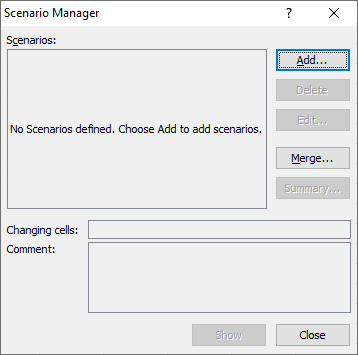
**Step 1:** On the Data tab, in the Forecast group, click What-If Analysis.



**Step 2:** Click Scenario Manager.

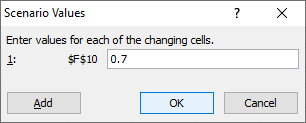
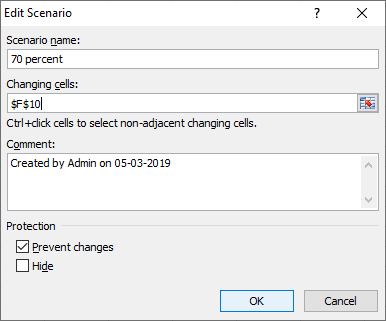


**Step 3:** Add a scenario by clicking on Add.

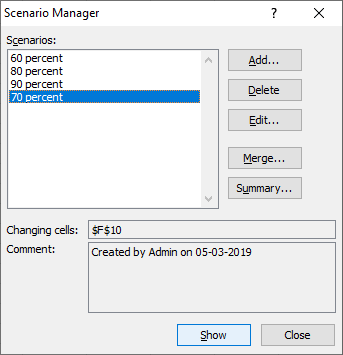


**Step 4:** Type a name (70% highest), select cell C4 (% sold for the highest price) for the Changing cells and click on OK.

Enter the corresponding value 0.7 and click on OK again.



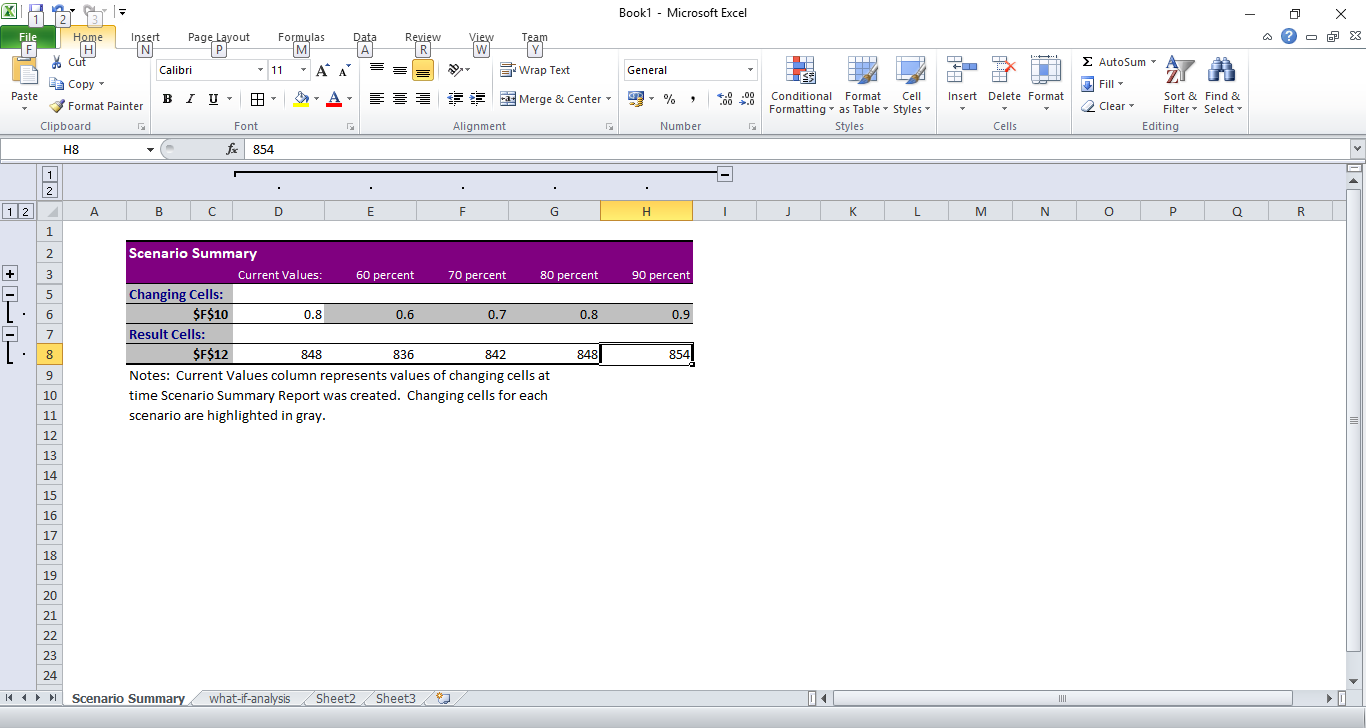
**Step 5:** Next, add 4 other scenarios (60%, 80%, 90% and 100%).



**Step 6:** Finally, your Scenario Manager should be consistent with the picture below:

**Note: to see the result of a scenario, select the scenario and click on the Show button. Excel will change the value of cell C4 accordingly for you to see the corresponding result on the sheet.**

1. **Scenario Summary**
2. **To easily compare the results of these scenarios, execute the following steps.**
3. **Click the Summary button in the Scenario Manager.**
4. **Next, select cell D10 (total profit) for the result cell and click on OK.**



**Roll no. 09 Practical No: 02 Date: 17/01/24**

**Aim: Implement K-means clustering**

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import sklearn

from sklearn.datasets import load\_iris

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

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)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=42)

kmeans.fit(X\_scaled)

wcss.append(kmeans.inertia\_)

plt.figure(figsize=(8, 6))

plt.plot(range(1, 11), wcss, marker='o', linestyle='--')

plt.title('Elbow Method')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS') # Within cluster sum of squares

plt.show()

kmeans = KMeans(n\_clusters=3, init='k-means++', random\_state=42)

kmeans.fit(X\_scaled)

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

centroids = kmeans.cluster\_centers\_

labels = kmeans.labels\_

plt.figure(figsize=(8, 6))

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=labels, cmap='viridis', s=50, alpha=0.5)

plt.scatter(centroids[:, 0], centroids[:, 1], marker='o', c='red', s=200, edgecolor='k')

plt.title('K-Means Clustering')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.colorbar()

plt.show()

cluster\_df = iris\_df.copy()

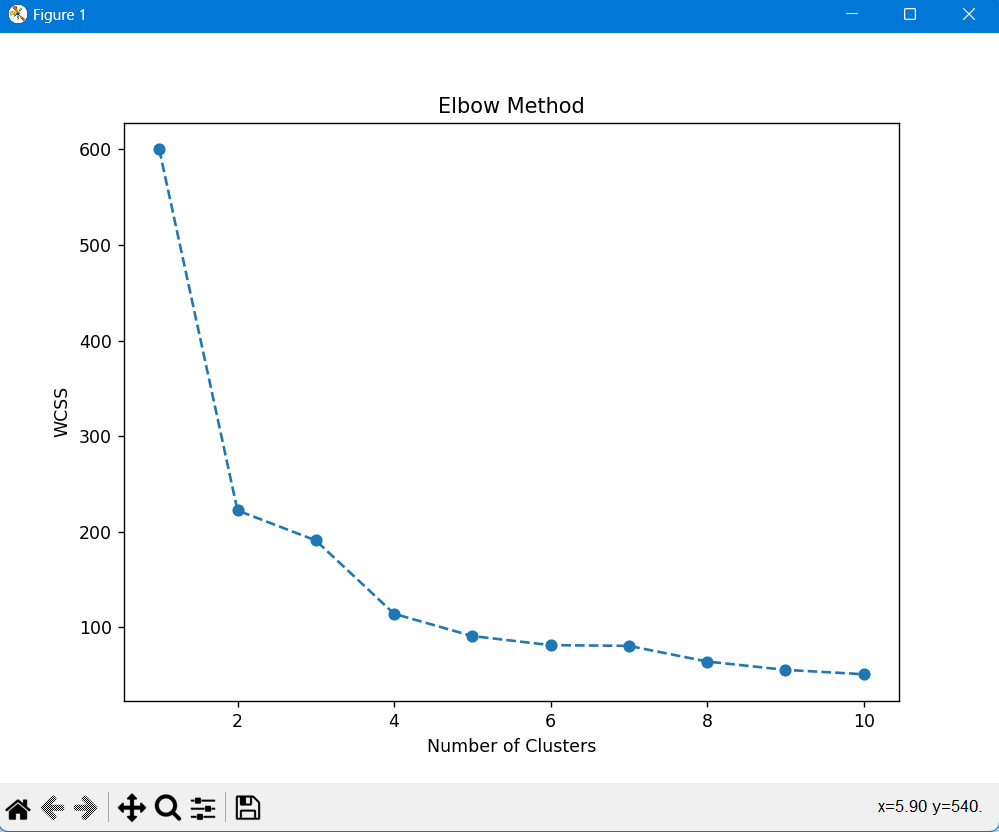
cluster\_df['cluster'] = labels

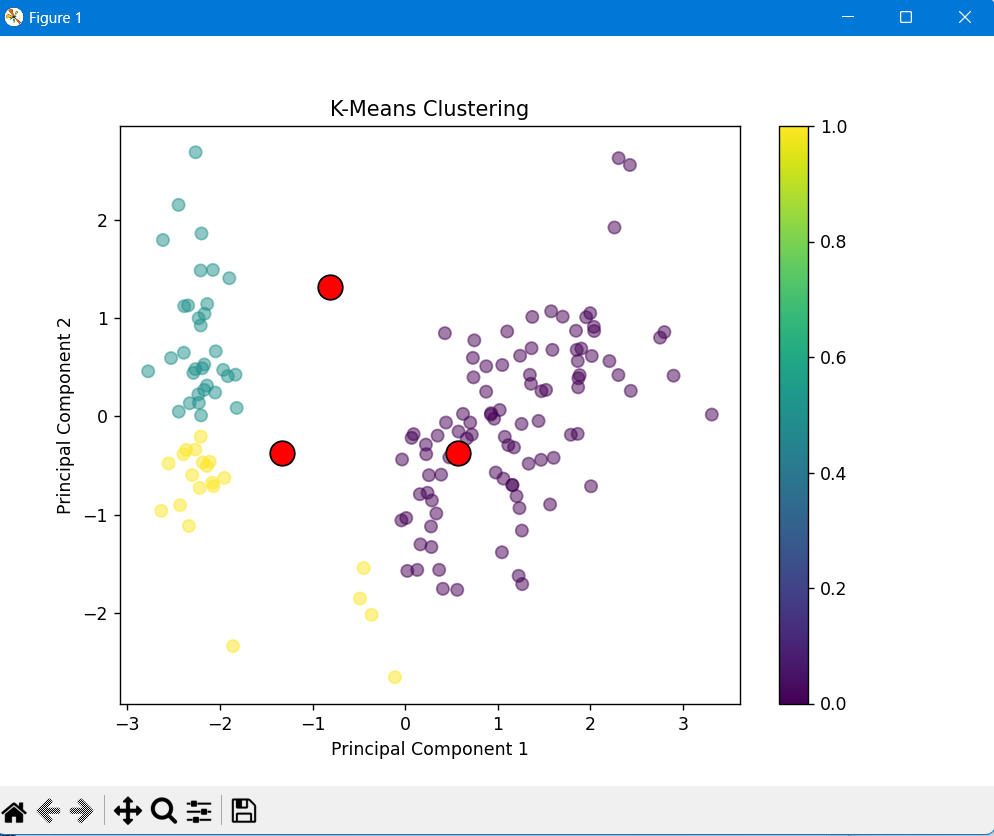
for cluster in sorted(cluster\_df['cluster'].unique()):

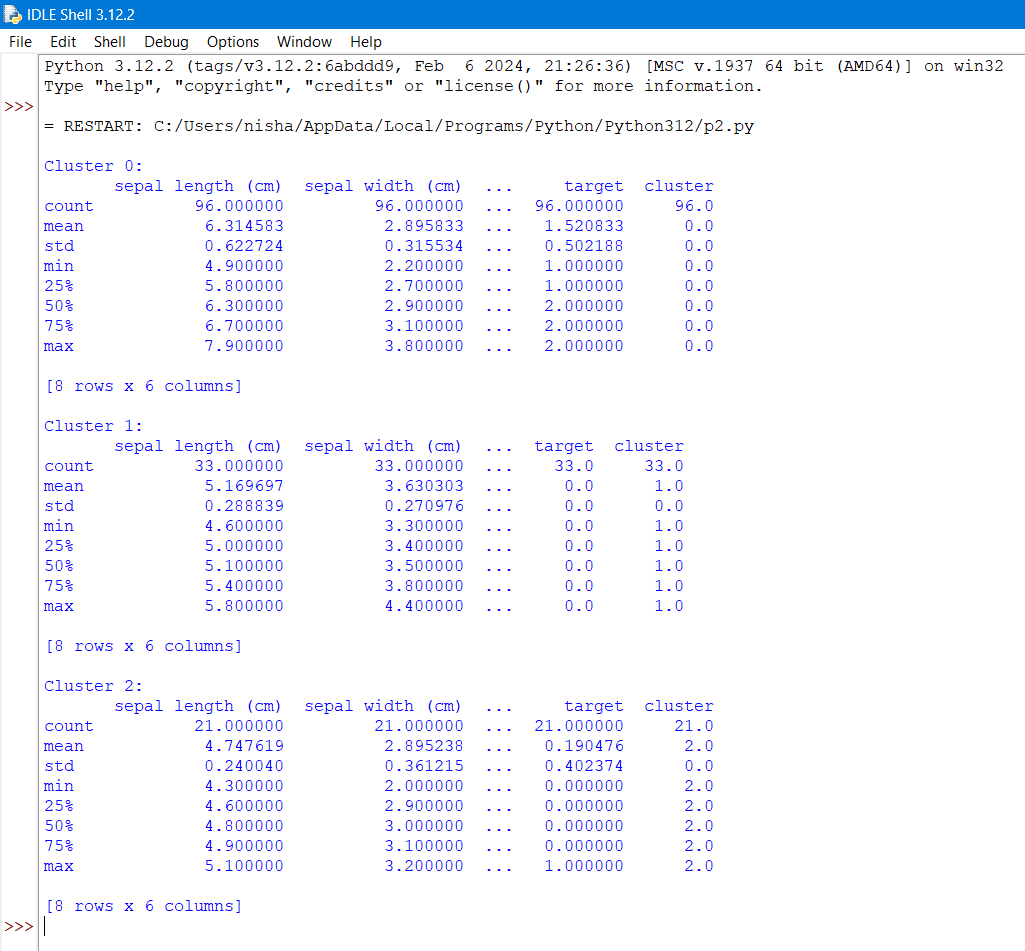
print(f"\nCluster {cluster}:")

print(cluster\_df[cluster\_df['cluster'] == cluster].describe())

**Output:**







**Roll no. 09 Practical No: 03 Date:24/01/24**

**Aim: Implement Logistic regression and Decision tree**

**Code:**

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

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']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

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))

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))

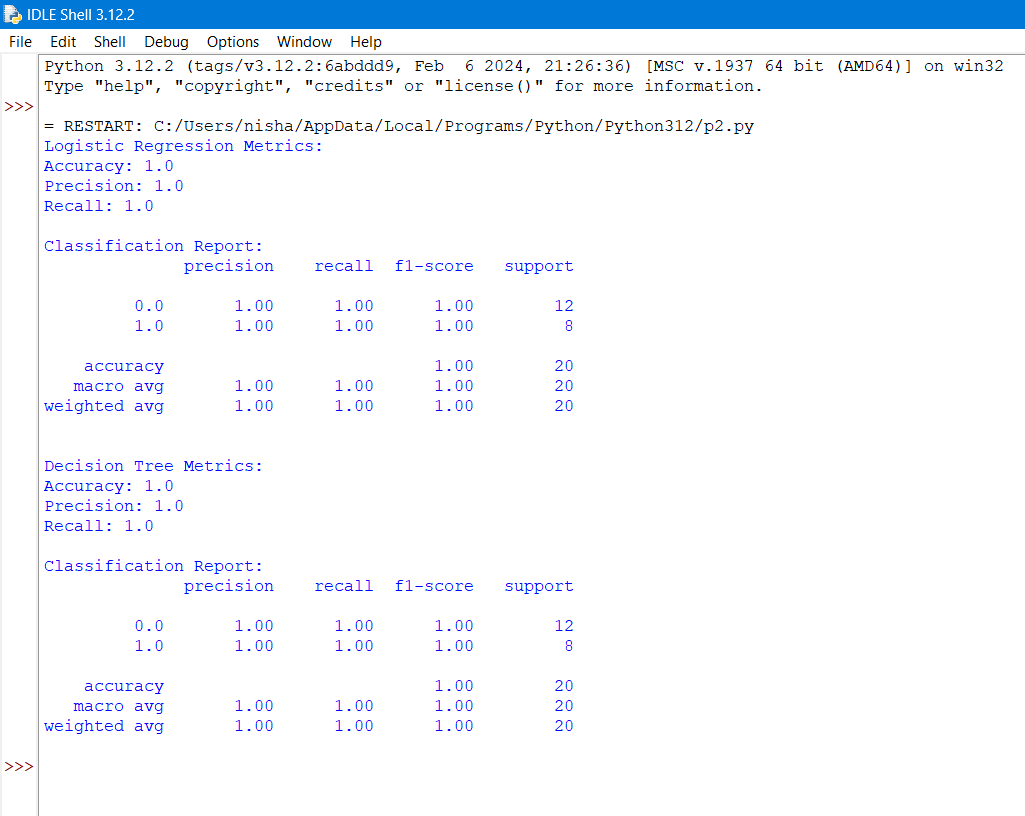
print("Precision:", precision\_score(y\_test, y\_pred\_tree))

print("Recall:", recall\_score(y\_test, y\_pred\_tree))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred\_tree))

**Output:**

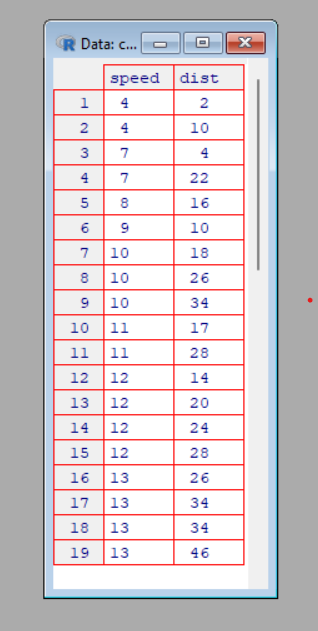


Roll no. 09 Practical no.4 Date: 27/01/24

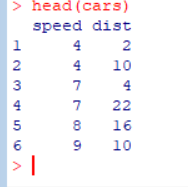
AIM: Linear regression (Weight and Height)

CODE:

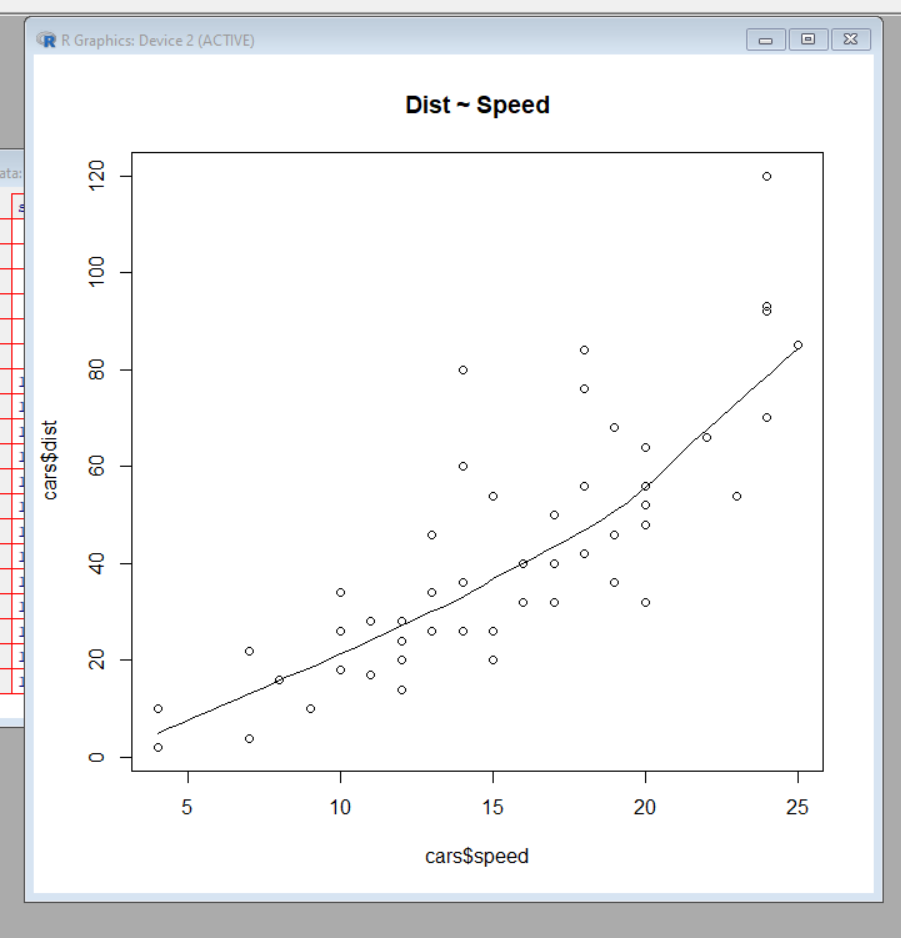
> View(cars)



> head(cars)

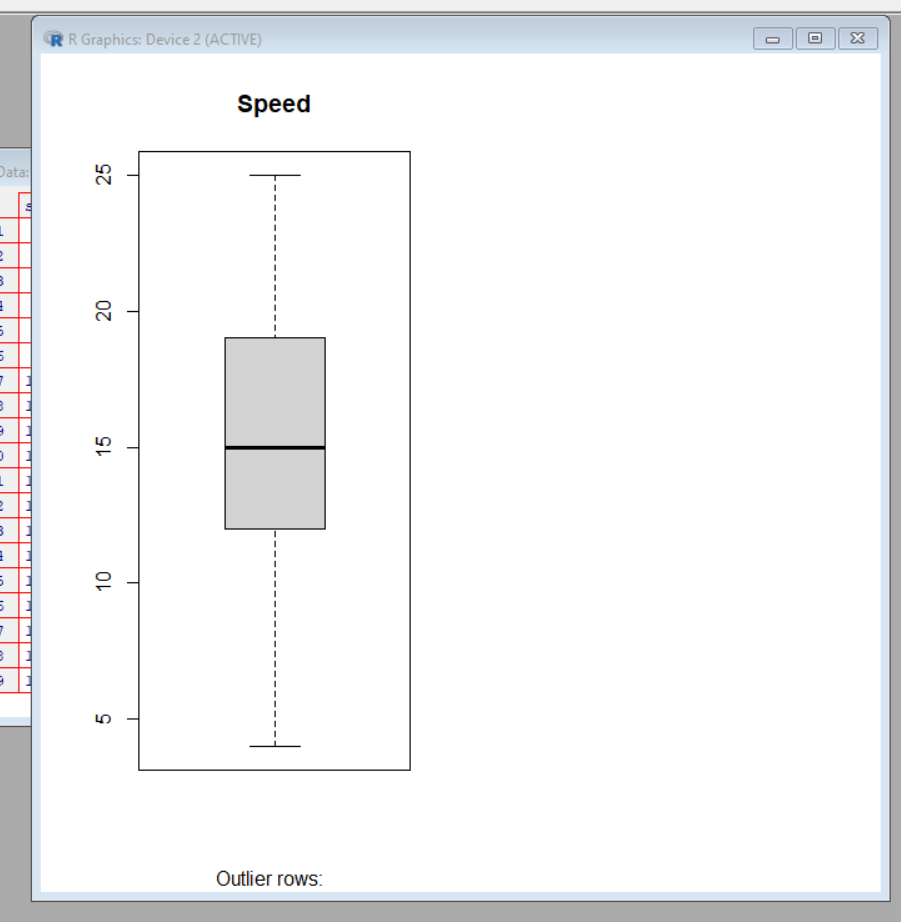


> scatter.smooth(x=cars$speed, y=cars$dist, main="Dist ~ Speed")

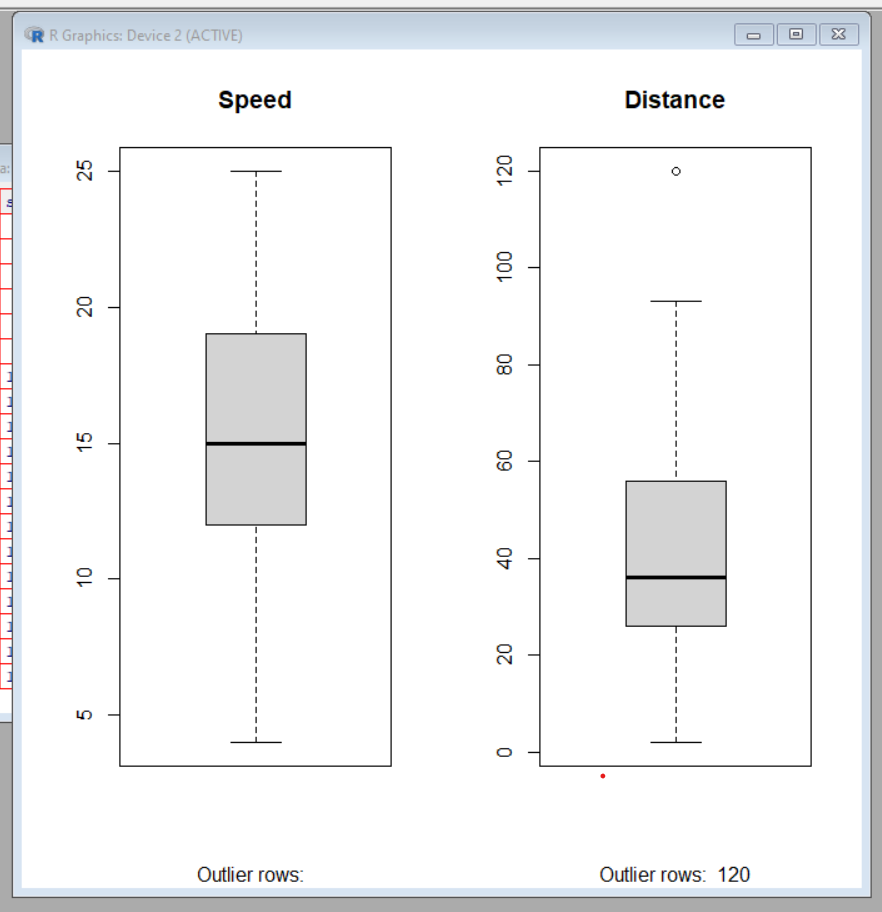


> par(mfrow=c(1, 2))

> boxplot(cars$speed, main="Speed", sub=paste("Outlier rows: ", boxplot.stats(cars$speed)$out))

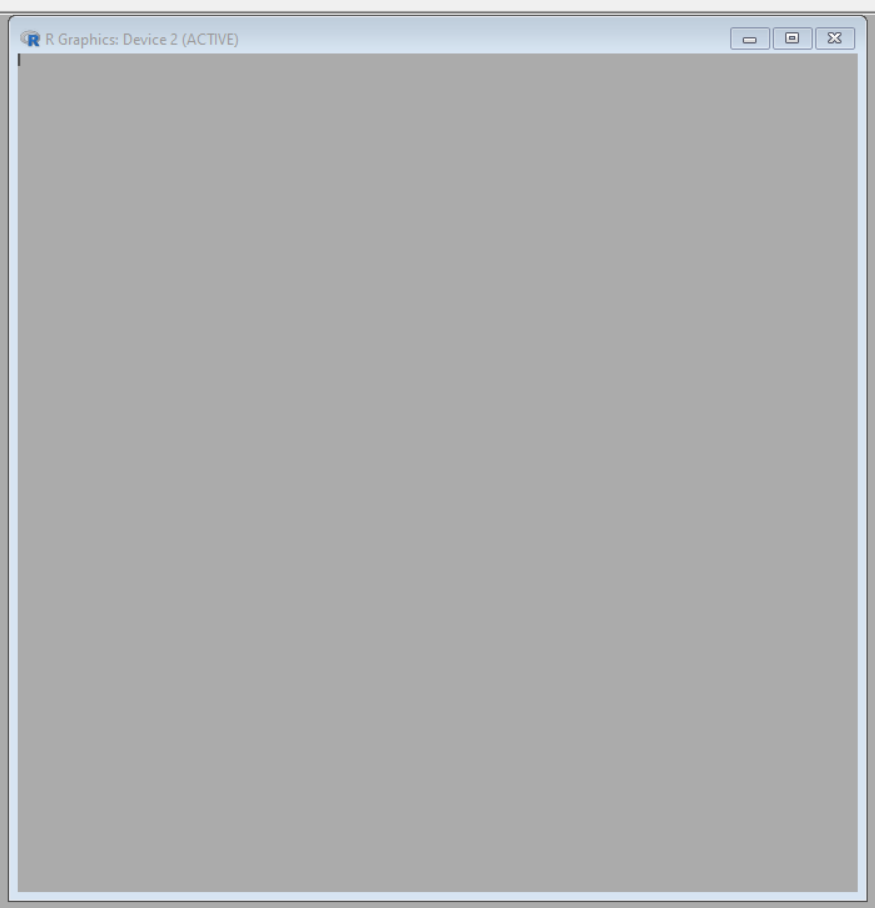


> boxplot(cars$dist, main="Distance", sub=paste("Outlier rows: ", boxplot.stats(cars$dist)$out))



> library(e1071)

> par(mfrow=c(1, 2))

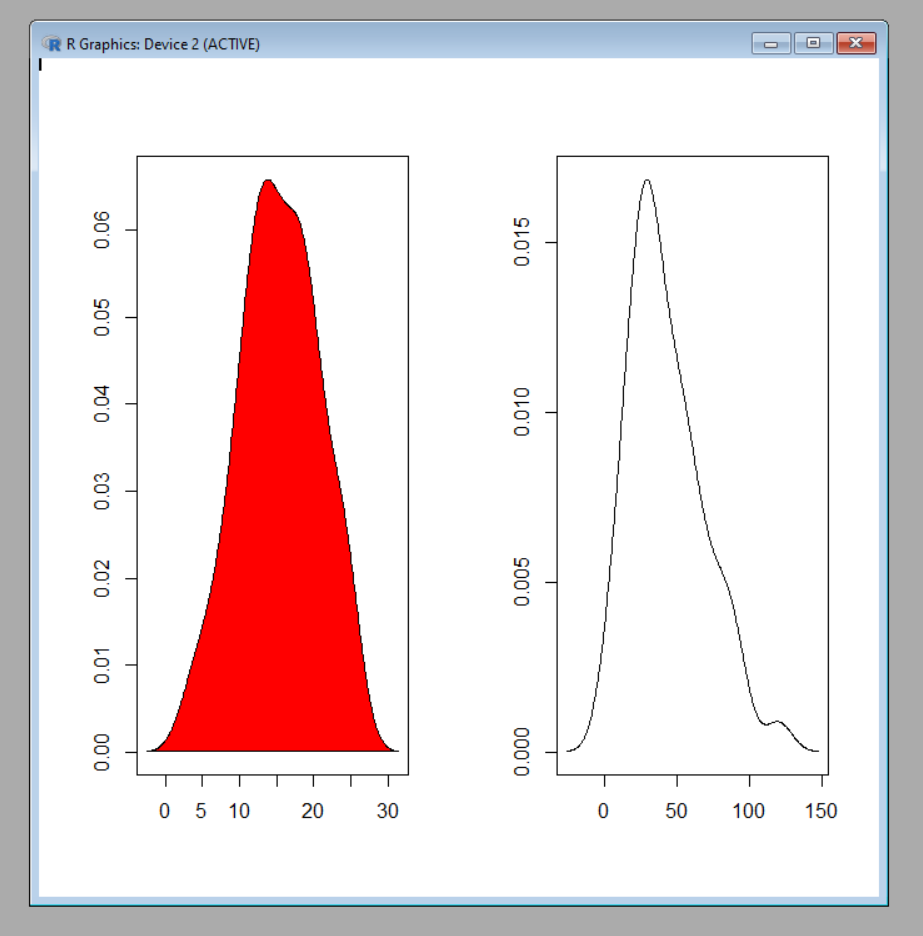


> plot(density(cars$speed), main="Density Plot: Speed", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(cars$speed), 2)))

> polygon(density(cars$speed), col="red")

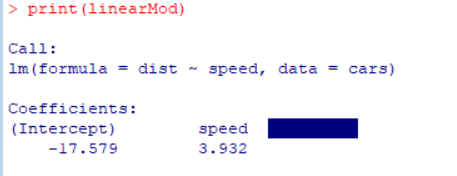
> plot(density(cars$dist), main="Density Plot: Distance", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(cars$dist), 2)))

> polygon(density(cars$dist), col="red") cor(cars$speed, cars$dist)

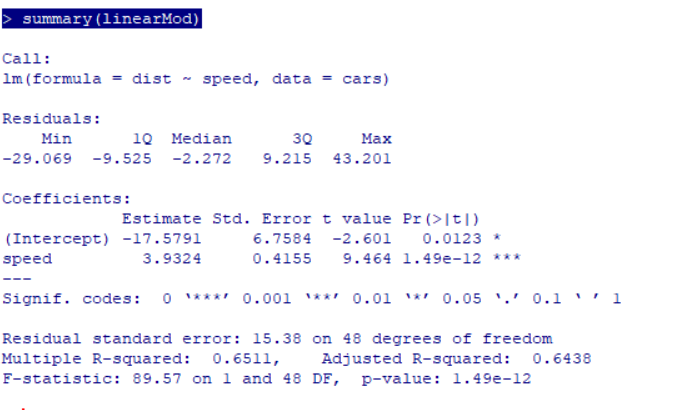


> linearMod <- lm(dist ~ speed, data=cars)

> print(linearMod)



> summary(linearMod)



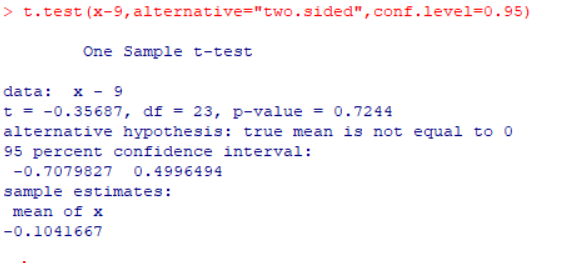
Roll no. 09 Practical no.5 Date:24/02/24

AIM: Hypothesis Testing.

CODE:

> x=c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6, 8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4, 10.8,11.3,11.9)

> t.test(x-9,alternative="two.sided",conf.level=0.95)

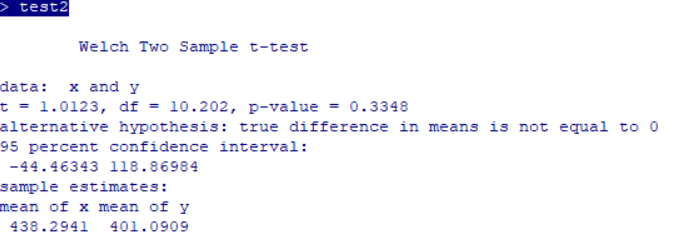


> x=c(418,421,421,422,425,427,431,434,437,439,446,447,448,453,454,463,465)

> y=c(429,430,430,431,36,437,440,441,445,446,447)

> test2<-t.test(x,y,alternative="two.sided",mu=0, var.equal=F,conf.level=0.95)

> test2



PYTHON CODE:

import numpy as np

from scipy import stats

import matplotlib.pyplot as plt

np.random.seed(42)

sample1 = np.random.normal(loc=10, scale=2, size=30)

sample2 = np.random.normal(loc=12, scale=2, size=30)

t\_statistic, p\_value = stats.ttest\_ind(sample1, sample2)

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}")

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()

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')

plt.show()

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.")

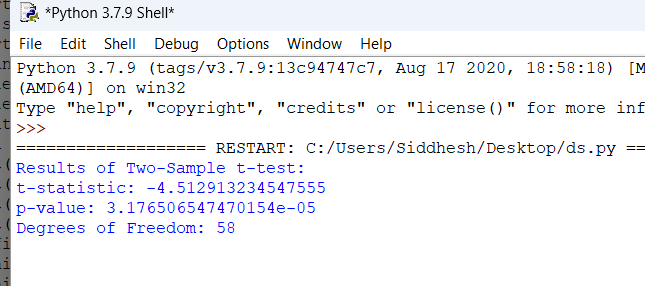
print("Interpretation: The mean of Sample 2 is significantly higher than that of Sample 1.")

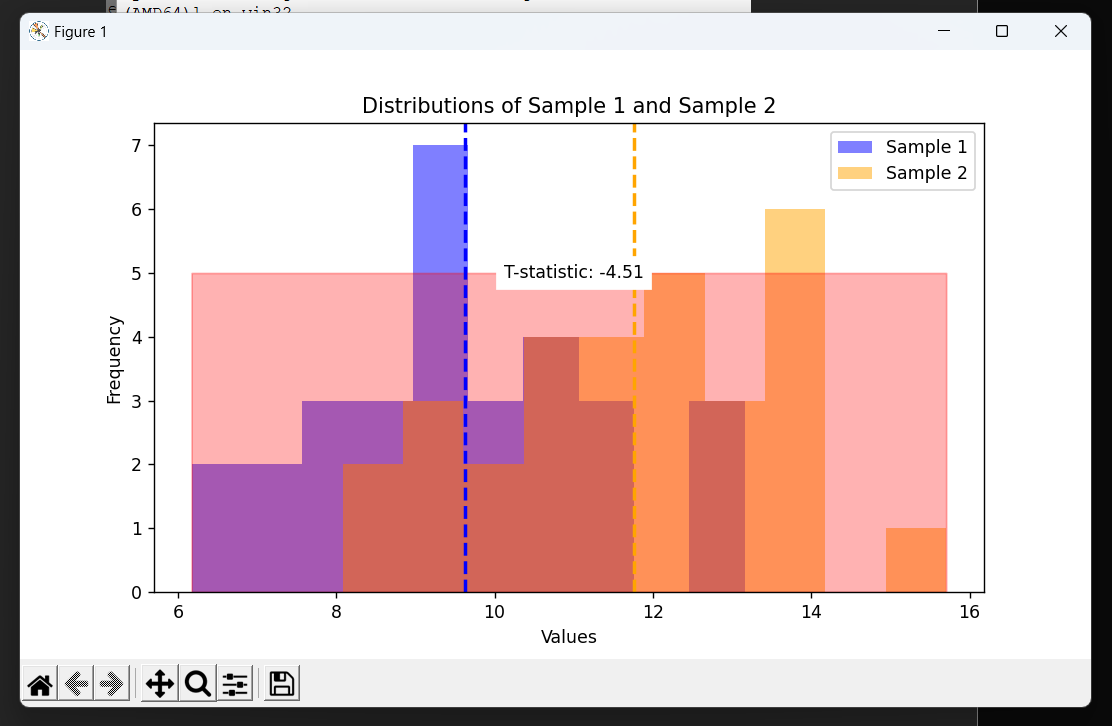
else:

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

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

OUTPUT:





Roll no:08 Practical no.6 Date:04/03/24

Aim:

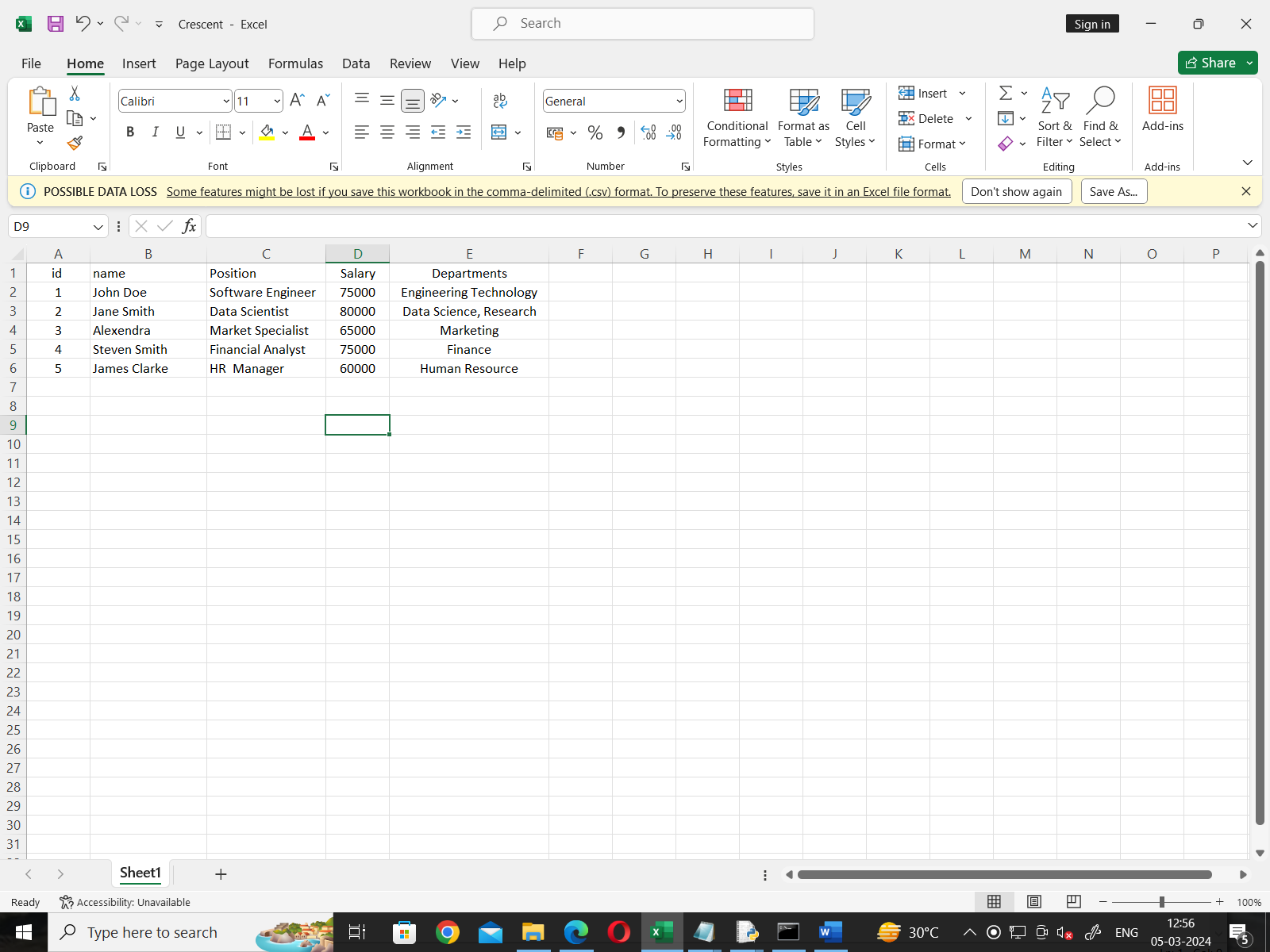
1. To read data from CSV and JSON files into a data frame.

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

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

Code:

Crescent.csv:



import pandas as pd

csv\_file\_path = 'Crescent.csv'

def read\_csv\_file(file\_path):

try:

data\_frame = pd.read\_csv(file\_path)

return data\_frame

except FileNotFoundError:

print(f"Error: File not found at path '{file\_path}'")

csv\_data = read\_csv\_file(csv\_file\_path)

if csv\_data is not None:

print("CSV Data:")

print(csv\_data)

Output:



( File Name: hkkk.py )

import json

json\_file\_path = 'Crescent.json'

def read\_json\_file(file\_path):

with open(file\_path, 'r') as file:

data = json.load(file)

return data

try:

json\_data = read\_json\_file(json\_file\_path)

print("JSON Data:")

print(json.dumps(json\_data, indent=2))

except FileNotFoundError:

print(f"Error: File not found at path '{json\_file\_path}'")

except json.JSONDecodeError:

print(f"Error: Invalid JSON format in file at path '{json\_file\_path}'")

( File Name: Crescent.json )

{

"employees": [

{

"id": 1,

"name": "John Doe",

"position": "Software Engineer",

"salary": 75000,

"departments": ["Engineering", "Technology"]

},

{

"id": 2,

"name": "Jane Smith",

"position": "Data Scientist",

"salary": 80000,

"departments": ["Data Science", "Research"]

},

{

"id": 3,

"name": "Alexandra",

"position": "Marketing Specialist",

"salary": 65000,

"departments": ["Marketing"]

},

{

"id": 4,

"name": "Steven Smith",

"position": "Financial Analyst",

"salary": 75000,

"departments": ["Finance"]

},

{

"id": 5,

"name": "James Clarke",

"position": "HR Manager",

"salary": 60000,

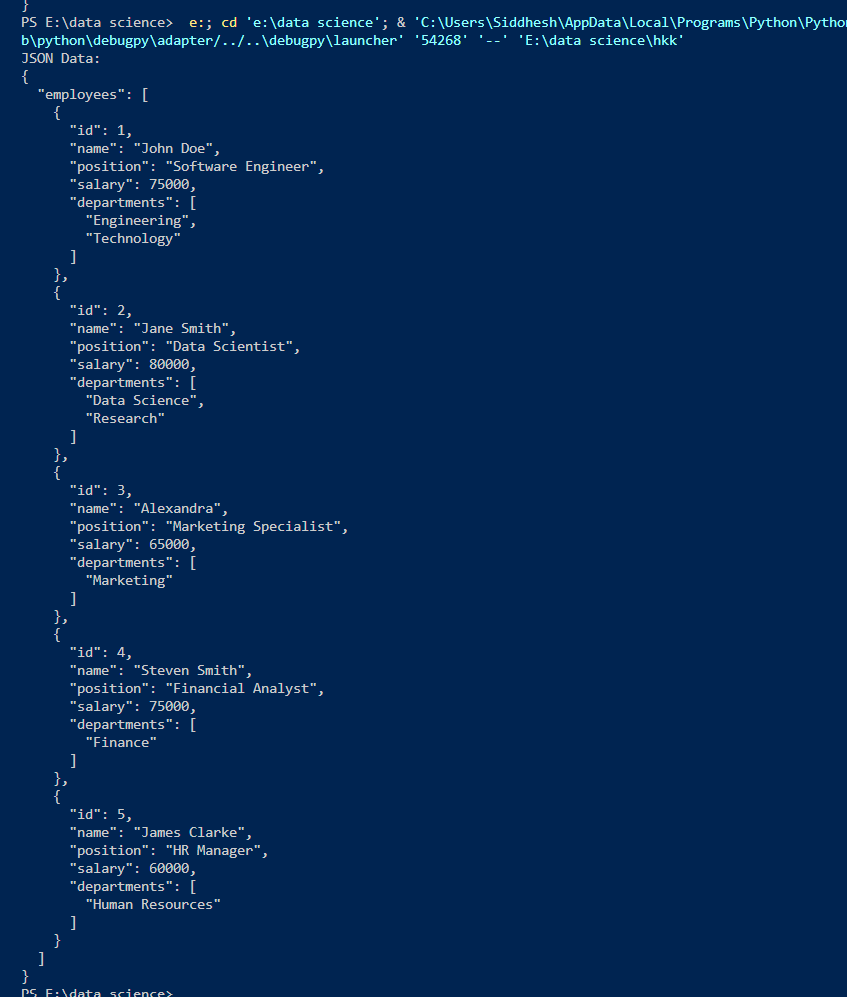
"departments": ["Human Resources"]

}

]

}

Output:



Roll no. 09 practical no.7 Date: 15/03/24

AIM: Feature Scaling and Dummification

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

SOURCE CODE:

import 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)

(wine.csv):

1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065

1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050

1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185

1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480

1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735

1,14.2,1.76,2.45,15.2,112,3.27,3.39,.34,1.97,6.75,1.05,2.85,1450

1,14.39,1.87,2.45,14.6,96,2.5,2.52,.3,1.98,5.25,1.02,3.58,1290

1,14.06,2.15,2.61,17.6,121,2.6,2.51,.31,1.25,5.05,1.06,3.58,1295

1,14.83,1.64,2.17,14,97,2.8,2.98,.29,1.98,5.2,1.08,2.85,1045

1,13.86,1.35,2.27,16,98,2.98,3.15,.22,1.85,7.22,1.01,3.55,1045

1,14.1,2.16,2.3,18,105,2.95,3.32,.22,2.38,5.75,1.25,3.17,1510

1,14.12,1.48,2.32,16.8,95,2.2,2.43,.26,1.57,5,1.17,2.82,1280

1,13.75,1.73,2.41,16,89,2.6,2.76,.29,1.81,5.6,1.15,2.9,1320

1,14.75,1.73,2.39,11.4,91,3.1,3.69,.43,2.81,5.4,1.25,2.73,1150

1,14.38,1.87,2.38,12,102,3.3,3.64,.29,2.96,7.5,1.2,3,1547

1,13.63,1.81,2.7,17.2,112,2.85,2.91,.3,1.46,7.3,1.28,2.88,1310

1,14.3,1.92,2.72,20,120,2.8,3.14,.33,1.97,6.2,1.07,2.65,1280

1,13.83,1.57,2.62,20,115,2.95,3.4,.4,1.72,6.6,1.13,2.57,1130

1,14.19,1.59,2.48,16.5,108,3.3,3.93,.32,1.86,8.7,1.23,2.82,1680

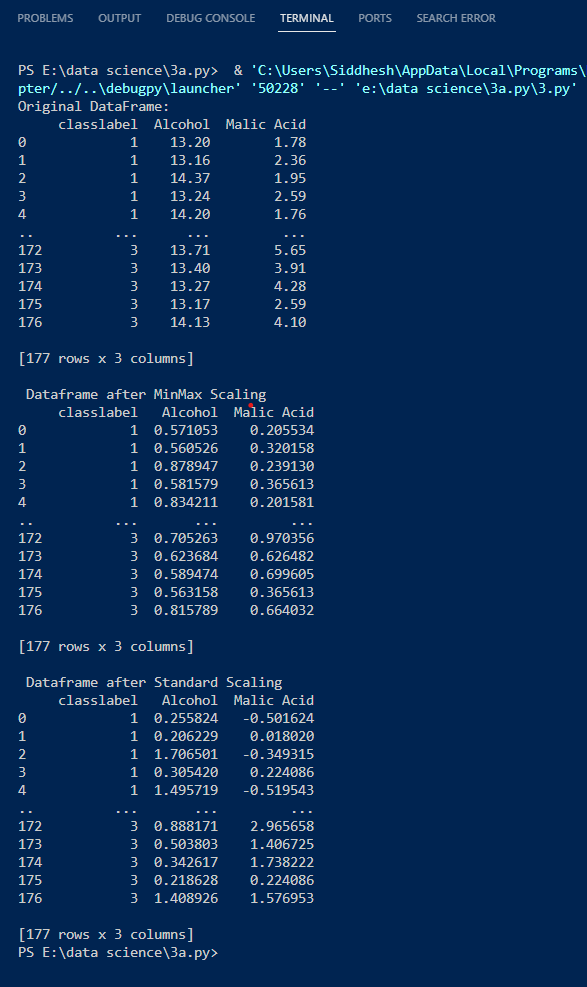
1,13.64,3.1,2.56,15.2,116,2.7,3.03,.17,1.66,5.1,.96,3.36,845

1,14.06,1.63,2.28,16,126,3,3.17,.24,2.1,5.65,1.09,3.71,780

1,12.93,3.8,2.65,18.6,102,2.41,2.41,.25,1.98,4.5,1.03,3.52,770

, ……..

OUTPUT:



1. Perform feature dummification to convert categorical variables into numerical representations.

SOURCE CODE:

import pandas as pd

iris = pd.read\_csv("Iris.csv")

print(iris)

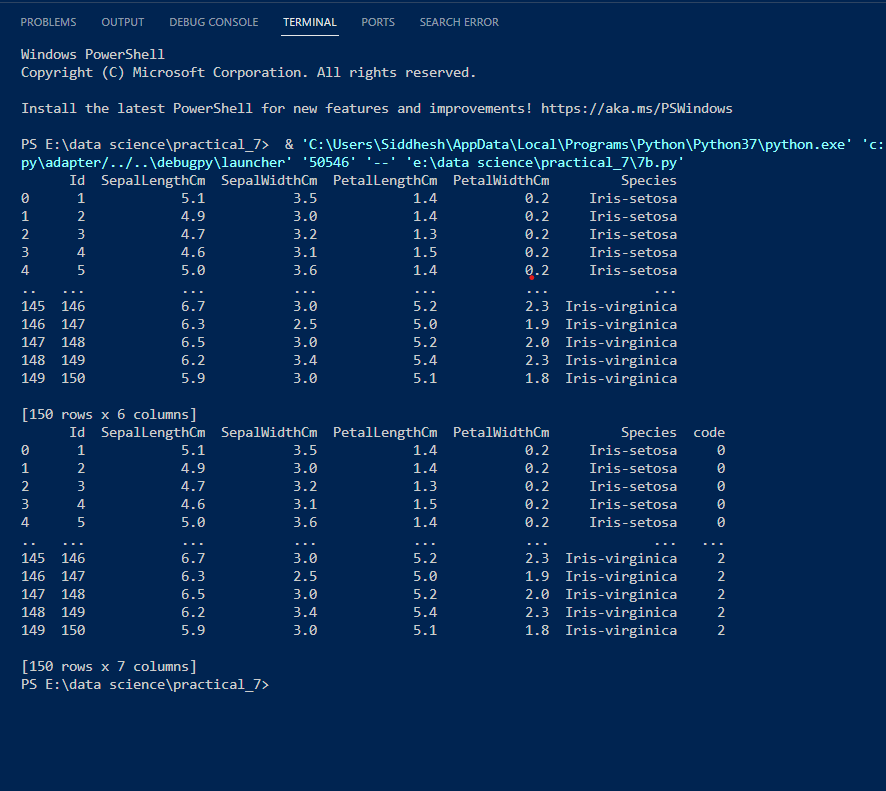
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

iris['code'] = le.fit\_transform(iris.Species)

print(iris)

OUTPUT:



Roll no. 09 PRACTICAL NO.9 DATE: 22/03/24

AIM: Implement Principal Component Analysis Code

SOURCE CODE:

# Iris Data data <- read.csv(file.choose(), header=T) str(data) data$NSP <- factor(data$NSP)

# Partition Data set.seed(111) ind <- sample(2, nrow(data), replace = TRUE, prob = c(.8, .2)) training <- data[ind==1,] testing <- data[ind==2,]

# Scatter Plots & Correlations library(psych) pairs.panels(training[,-22], gap=0, bg=c("red","yellow","blue")[training$NSP], pch=21)

# PCA

pc <- prcomp(training[,-22], center = TRUE, scale. = TRUE) attributes(pc) print(pc) summary(pc)

plot(pc, type = "lines")

# Orthogonality of PCs pairs.panels(pc$x, gap=0, bg=c("red","yellow","blue")[training$NSP], pch=21) # Bi-Plot library(devtools) install\_github("ggbiplot", "vqv") library(ggbiplot) g <- ggbiplot(pc,

obs.scale = 1, var.scale = 1, groups = training$NSP, ellipse = TRUE, circle = TRUE, ellipse.prob = 0.68) g <- g + scale\_color\_discrete(name = '') g <- g + theme(legend.direction = 'horizontal', legend.position = 'top') print(g)

# Prediction with Principal Components trg <- predict(pc, training) trg <- data.frame(trg, training[22]) tst <- predict(pc, testing) tst <- data.frame(tst, testing[22])

# Multinomial Logistic regression with 1st two PCs library(nnet) trg$NSP<-relevel(trg$NSP, ref="1") mymodel <- multinom(NSP~PC1+PC2+PC3+PC4+PC5, data=trg) summary(mymodel)

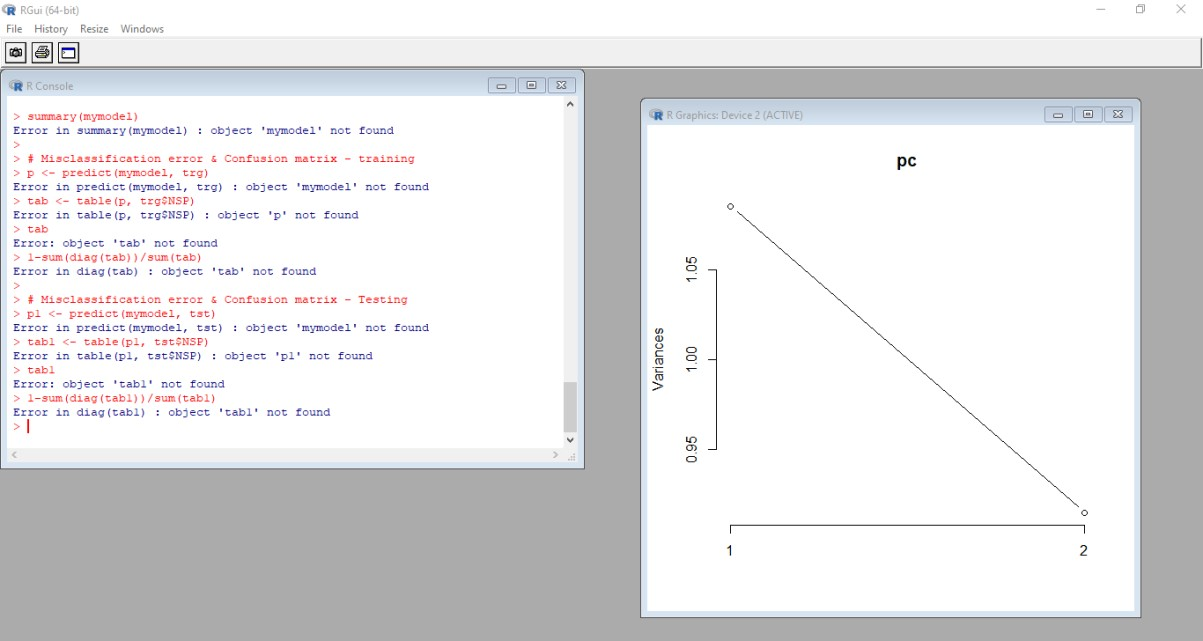
# Misclassification error & Confusion matrix - training p <- predict(mymodel, trg) tab <- table(p, trg$NSP) tab

1-sum(diag(tab))/sum(tab)

# Misclassification error & Confusion matrix - Testing p1 <- predict(mymodel, tst) tab1 <- table(p1, tst$NSP) tab1

1-sum(diag(tab1))/sum(tab1)

OUTPUT:



Roll no. 09 PRACTICAL NO.8 DATE: 22/03/24

AIM: ONE WAY ANOVA

SOURCE CODE:

import scipy.stats as stats

from statsmodels.stats.multicomp import pairwise\_tukeyhsd

group1 = [23, 25, 29, 34, 30]

group2 = [19, 20, 22, 25, 24]

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)

# Perform one-way ANOVA

f\_statistic, p\_value = stats.f\_oneway(group1, group2, group3, group4)

# Print ANOVA results

print("One-way ANOVA:")

print("F-statistic:", f\_statistic)

print("P-value:", p\_value)

# Perform Tukey-Kramer post-hoc test

tukey\_results = pairwise\_tukeyhsd(all\_data, group\_labels)

# Print Tukey-Kramer results

print("\nTukey-Kramer post-hoc test:")

print(tukey\_results)

OUTPUT:

