Practical 1

Aim: Perform Descriptive statistic on given data set.

Overview of selected dataset:

- In this practical I'll use the Iris dataset, which is a classic dataset in statistics and machine learning.
- It consists of 150 samples of iris flowers, with four features measured for each sample: sepal length, sepal width, petal length, and petal width.
- The dataset also includes the species of each flower.

Using Python code in Google Colab to perform Descriptive statistics:

```python
#DSV prac1 - Dipam Shah

import pandas as pd
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()

# Create a DataFrame with the dataset
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target

# Map the target numbers to actual species names
iris_df['species'] = iris_df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

# Display the first few rows of the dataset
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Figure(i)

```
# Perform descriptive statistics
descriptive_stats = iris_df.describe()

# Add descriptive statistics for the species column
species_counts = iris_df['species'].value_counts()
species_counts.name = 'count'

# Display descriptive statistics
descriptive_stats, species_counts
```

```
(       sepal length (cm)  sepal width (cm)  petal length (cm)  \
 count         150.000000        150.000000         150.000000
 mean            5.843333          3.057333           3.758000
 std             0.828066          0.435866           1.765298
 min             4.300000          2.000000           1.000000
 25%             5.100000          2.800000           1.600000
 50%             5.800000          3.000000           4.350000
 75%             6.400000          3.300000           5.100000
 max             7.900000          4.400000           6.900000

        petal width (cm)
 count        150.000000
 mean           1.199333
 std            0.762238
 min            0.100000
 25%            0.300000
 50%            1.300000
 75%            1.800000
 max            2.500000  ,
 species
 setosa         50
 versicolor     50
 virginica      50
 Name: count, dtype: int64)
```

Figure(ii)

Practical 2

Aim: Consider dataset with student name, gender, Enrolment no, 4-semester result with marks of each subject, his mobile number, city. Implement the following in Python. Perform descriptive analysis and identify the data type and implement a method to find out variation in data. Show the difference between the highest and lowest marks in each subject semester-wise.

Code:

```
#DSV prac2 - Dipam Shah

import pandas as pd

# Sample dataset creation

data = {

    'Student Name': ['Arya', 'Dipam', 'Karmil', 'Dev'],

    'Gender': ['F', 'M', 'M', 'M'],

    'Enrolment No': [1001, 1002, 1003, 1004],

    'Math': [ [100, 90, 100, 90], [90, 90, 95, 90], [92, 88, 85, 84], [70,
90, 87, 91]],

    'Science': [[100, 90, 100, 90], [90, 90, 95, 90], [85, 90, 88, 87],
[95, 89, 92, 90]],

    'Mobile Number': [1234567890, 1234567891, 1234567892, 1234567893],

    'City': ['Jacksonville', 'Los Angeles', 'Chicago', 'Houston']

}


df = pd.DataFrame(data)


# Add semester information as columns

df[['Sem1_Math', 'Sem2_Math', 'Sem3_Math', 'Sem4_Math']] =
pd.DataFrame(df.Math.tolist(), index= df.index)

df[['Sem1_Science', 'Sem2_Science', 'Sem3_Science', 'Sem4_Science']] =
pd.DataFrame(df.Science.tolist(), index= df.index)
```

```
df = df.drop(['Math', 'Science'], axis=1) # Remove original 'Math' and
'Science' columns


print(df)


# Descriptive statistics
desc_stats = df.describe(include='all')
print(desc_stats)


# Function to calculate variation in marks (modified to accommodate new
structure)
def calculate_variation(df, subject, semesters):
    variations = {}
    for semester in semesters:
        column_name = f"Sem{semester}_{subject}"
        max_mark = df[column_name].max()
        min_mark = df[column_name].min()
        variations[semester] = max_mark - min_mark
    return variations


# Calculate variations for Math and Science across semesters
math_variations = calculate_variation(df, 'Math', [1, 2, 3, 4])
science_variations = calculate_variation(df, 'Science', [1, 2, 3, 4])


print("Variation in Math across semesters:", math_variations)
print("Variation in Science across semesters:", science_variations)
```

```
  Student Name Gender  Enrolment No  Mobile Number          City  Sem1_Math  \
0         Arya      F          1001     1234567890  Jacksonville        100
1        Dipam      M          1002     1234567891   Los Angeles         90
2       Karmil      M          1003     1234567892       Chicago         92
3          Dev      M          1004     1234567893       Houston         70

   Sem2_Math  Sem3_Math  Sem4_Math  Sem1_Science  Sem2_Science  Sem3_Science  \
0         90        100         90           100            90           100
1         90         95         90            90            90            95
2         88         85         84            85            90            88
3         90         87         91            95            89            92

   Sem4_Science
0            90
1            90
2            87
3            90
       Student Name Gender  Enrolment No  Mobile Number          City  \
count             4      4      4.000000   4.000000e+00             4
unique            4      2           NaN            NaN             4
top            Arya      M           NaN            NaN  Jacksonville
freq              1      3           NaN            NaN             1
mean            NaN    NaN   1002.500000   1.234568e+09           NaN
std             NaN    NaN      1.290994   1.290994e+00           NaN
min             NaN    NaN   1001.000000   1.234568e+09           NaN
25%             NaN    NaN   1001.750000   1.234568e+09           NaN
50%             NaN    NaN   1002.500000   1.234568e+09           NaN
75%             NaN    NaN   1003.250000   1.234568e+09           NaN
max             NaN    NaN   1004.000000   1.234568e+09           NaN

         Sem1_Math  Sem2_Math   Sem3_Math  Sem4_Math  Sem1_Science  \
count     4.000000        4.0    4.000000   4.000000      4.000000
unique         NaN        NaN         NaN        NaN           NaN
top            NaN        NaN         NaN        NaN           NaN
freq           NaN        NaN         NaN        NaN           NaN
mean     88.000000       89.5   91.750000  88.750000     92.500000
std      12.754084        1.0    6.994045   3.201562      6.454972
min      70.000000       88.0   85.000000  84.000000     85.000000
25%      85.000000       89.5   86.500000  88.500000     88.750000
50%      91.000000       90.0   91.000000  90.000000     92.500000
75%      94.000000       90.0   96.250000  90.250000     96.250000
max     100.000000       90.0  100.000000  91.000000    100.000000
```

Figure(i)

	Sem2_Science	Sem3_Science	Sem4_Science
count	4.00	4.000000	4.00
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	89.75	93.750000	89.25
std	0.50	5.057997	1.50
min	89.00	88.000000	87.00
25%	89.75	91.000000	89.25
50%	90.00	93.500000	90.00
75%	90.00	96.250000	90.00
max	90.00	100.000000	90.00

```
Variation in Math across semesters: {1: 30, 2: 2, 3: 15, 4: 7}
Variation in Science across semesters: {1: 15, 2: 1, 3: 12, 4: 3}
```

Figure(ii)

Practical 3

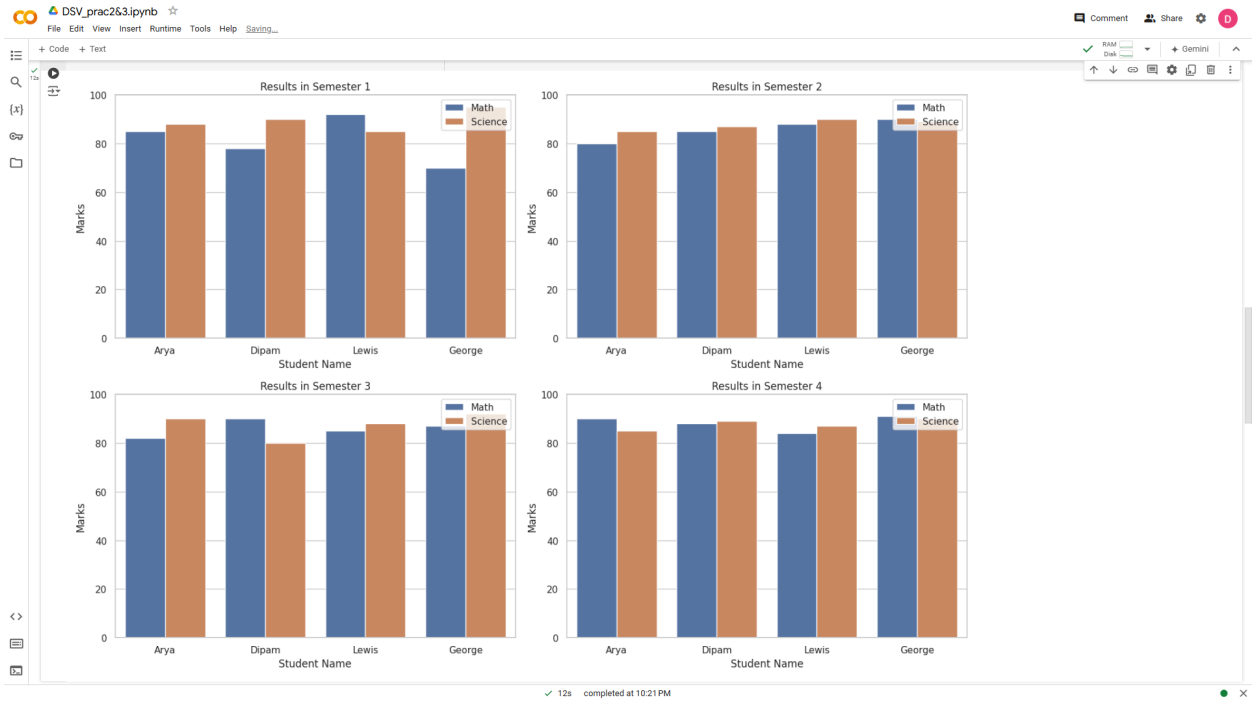
Aim: Plot the graph showing the results of students in each semester. (Using Practical 2 dataset).

Code:

```
#DSV prac3 - Dipam Shah


import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


data = {

    'Student Name': ['Arya', 'Dipam', 'Lewis', 'George'],

    'Gender': ['F', 'M', 'M', 'M'],

    'Enrolment No': [1001, 1002, 1003, 1004],

    'Semester 1 Math': [85, 78, 92, 70],

    'Semester 1 Science': [88, 90, 85, 95],

    'Semester 2 Math': [80, 85, 88, 90],

    'Semester 2 Science': [85, 87, 90, 89],

    'Semester 3 Math': [82, 90, 85, 87],

    'Semester 3 Science': [90, 80, 88, 92],

    'Semester 4 Math': [90, 88, 84, 91],

    'Semester 4 Science': [85, 89, 87, 90],

    'Mobile Number': [1234567890, 1234567891, 1234567892, 1234567893],

    'City': ['Jacksonville', 'Los Angeles', 'UK', 'London']

}


df = pd.DataFrame(data)
```

```
df_long = pd.melt(df, id_vars=['Student Name'],
                  value_vars=[col for col in df.columns if 'Semester' in
col],
                  var_name='Subject', value_name='Marks')

df_long['Semester'] = df_long['Subject'].apply(lambda x: x.split()[1])
df_long['Subject'] = df_long['Subject'].apply(lambda x: x.split()[2])

plt.figure(figsize=(15, 10))
sns.set(style="whitegrid")

for semester in ['1', '2', '3', '4']:
    subset = df_long[df_long['Semester'] == semester]
    plt.subplot(2, 2, int(semester))
    sns.barplot(x='Student Name', y='Marks', hue='Subject', data=subset)
    plt.title(f'Results in Semester {semester}')
    plt.ylim(0, 100)
    plt.legend(loc='upper right')

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

Result:



Figure(i)

Practical 4

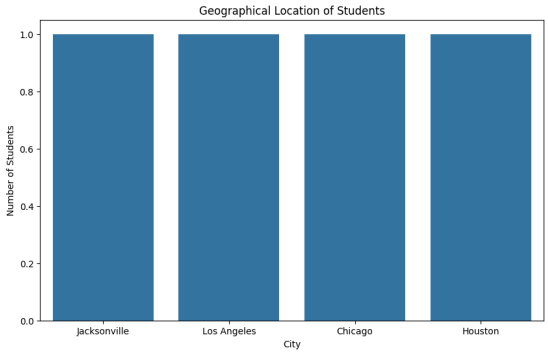
Aim: Plot the graph showing the geographical location of students, also plot the graph showing number of male and female students and implement a method to treat missing values for gender and missing value for marks. (Using Practical 2 dataset)

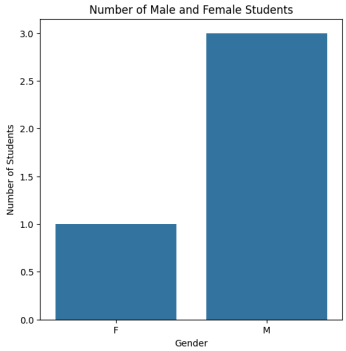
Code:

```
#DSV prac4 - Dipam Shah

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


data = {

    'Student Name': ['Arya', 'Dipam', 'Lewis', 'David'],

    'Gender': ['F', 'M', 'M', None],

    'Enrolment No': [1001, 1002, 1003, 1004],

    'Semester 1 Math': [85, 78, None, 70],

    'Semester 1 Science': [88, 90, 85, 95],

    'Semester 2 Math': [80, 85, 88, 90],

    'Semester 2 Science': [85, 87, 90, 89],

    'Semester 3 Math': [82, 90, 85, 87],

    'Semester 3 Science': [90, 80, 88, 92],

    'Semester 4 Math': [90, 88, 84, 91],

    'Semester 4 Science': [85, 89, 87, 90],

    'Mobile Number': [1234567890, 1234567891, 1234567892, 1234567893],

    'City': ['Jacksonville', 'Los Angeles', 'Chicago', 'Houston']

}


df = pd.DataFrame(data)
```

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


for col in df.columns:
    if 'Math' in col or 'Science' in col:
        df[col].fillna(df[col].mean(), inplace=True)


plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='City', order=df['City'].value_counts().index)
plt.title('Geographical Location of Students')
plt.xlabel('City')
plt.ylabel('Number of Students')
plt.show()


plt.figure(figsize=(6, 6))
sns.countplot(data=df, x='Gender')
plt.title('Number of Male and Female Students')
plt.xlabel('Gender')
plt.ylabel('Number of Students')
plt.show()


print(df)
```

Result:





	Student Name	Gender	Enrolment No	Semester 1 Math	Semester 1 Science	\
0	Arya	F	1001	85.000000	88	
1	Dipam	M	1002	78.000000	90	
2	Lewis	M	1003	77.666667	85	
3	David	M	1004	70.000000	95	

	Semester 2 Math	Semester 2 Science	Semester 3 Math	Semester 3 Science	\
0	80	85	82	90	
1	85	87	90	80	
2	88	90	85	88	
3	90	89	87	92	

	Semester 4 Math	Semester 4 Science	Mobile Number	City
0	90	85	1234567890	Jacksonville
1	88	89	1234567891	Los Angeles
2	84	87	1234567892	Chicago
3	91	90	1234567893	Houston

Figure(i)

Practical 5

Aim: Study the various graphs using visualization library.

Code:

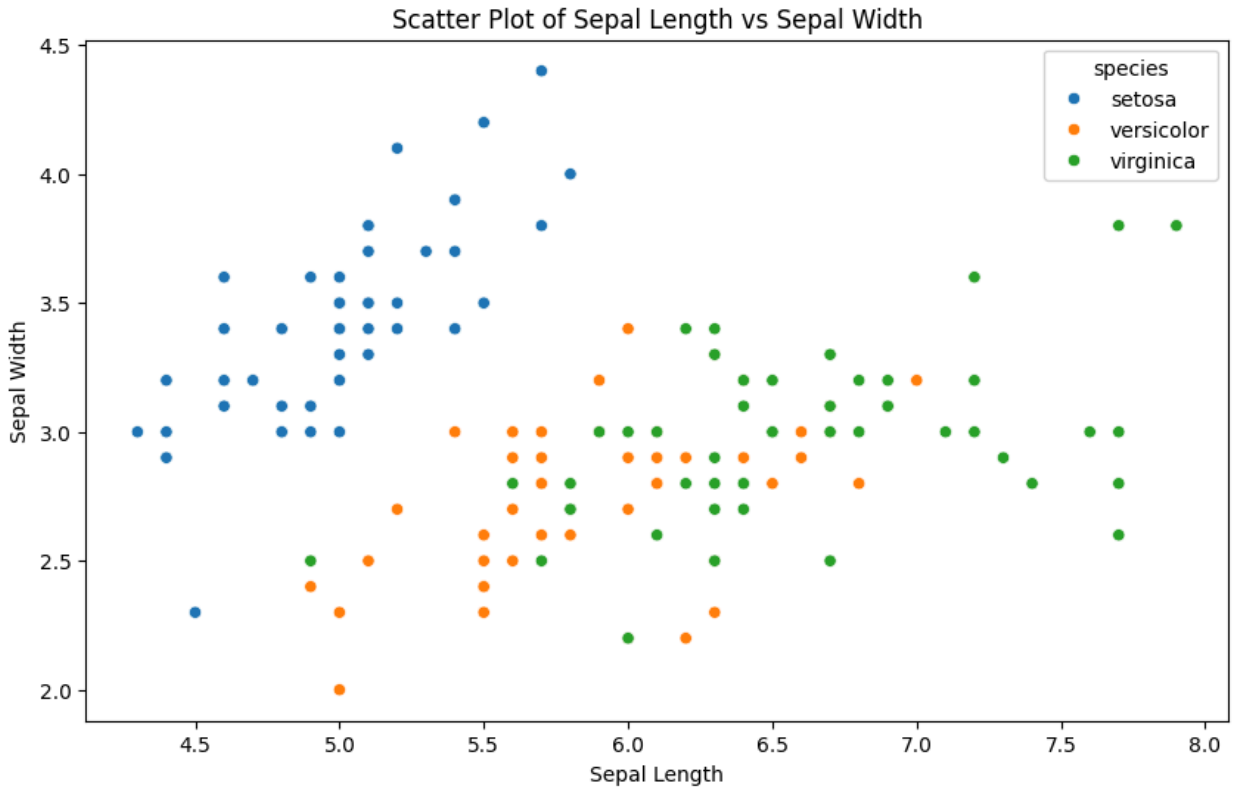
```
#Prac5 DSV by Dipam Shah

!pip install matplotlib seaborn plotly


import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px


df = sns.load_dataset('iris')

print("Dataset Head:")

print(df.head())


plt.figure(figsize=(10, 6))

plt.plot(df['species'], df['sepal_length'], marker='o', linestyle='--')

plt.title('Line Plot of Sepal Length by Species')

plt.xlabel('Species')

plt.ylabel('Sepal Length')

plt.grid(True)

plt.show()


plt.figure(figsize=(10, 6))

sns.scatterplot(data=df, x='sepal_length', y='sepal_width', hue='species')

plt.title('Scatter Plot of Sepal Length vs Sepal Width')

plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width')
```

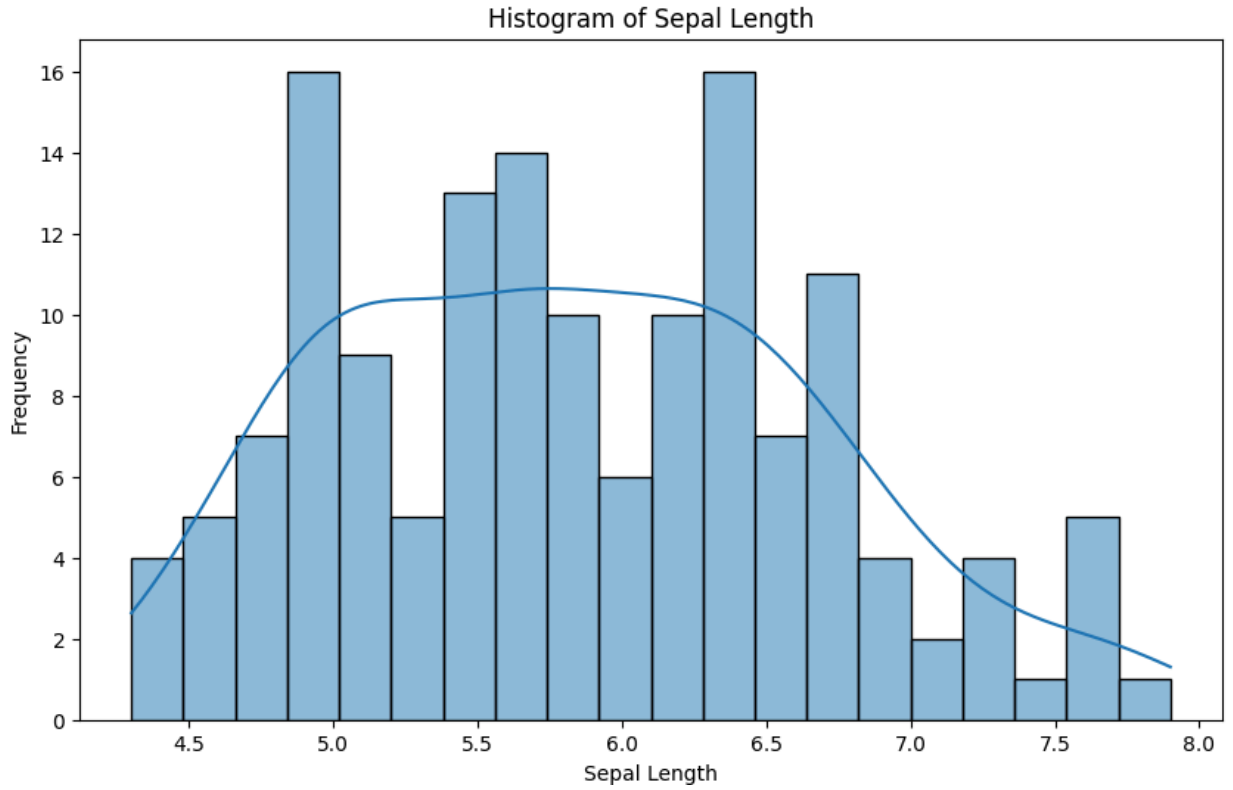
```
plt.show()


plt.figure(figsize=(10, 6))
sns.histplot(df['sepal_length'], bins=20, kde=True)
plt.title('Histogram of Sepal Length')
plt.xlabel('Sepal Length')
plt.ylabel('Frequency')
plt.show()


plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='species', y='sepal_length')
plt.title('Box Plot of Sepal Length by Species')
plt.xlabel('Species')
plt.ylabel('Sepal Length')
plt.show()


fig = px.scatter(df, x='sepal_length', y='sepal_width', color='species',
title='Interactive Scatter Plot of Sepal Length vs Sepal Width')
fig.show()
```

Result:

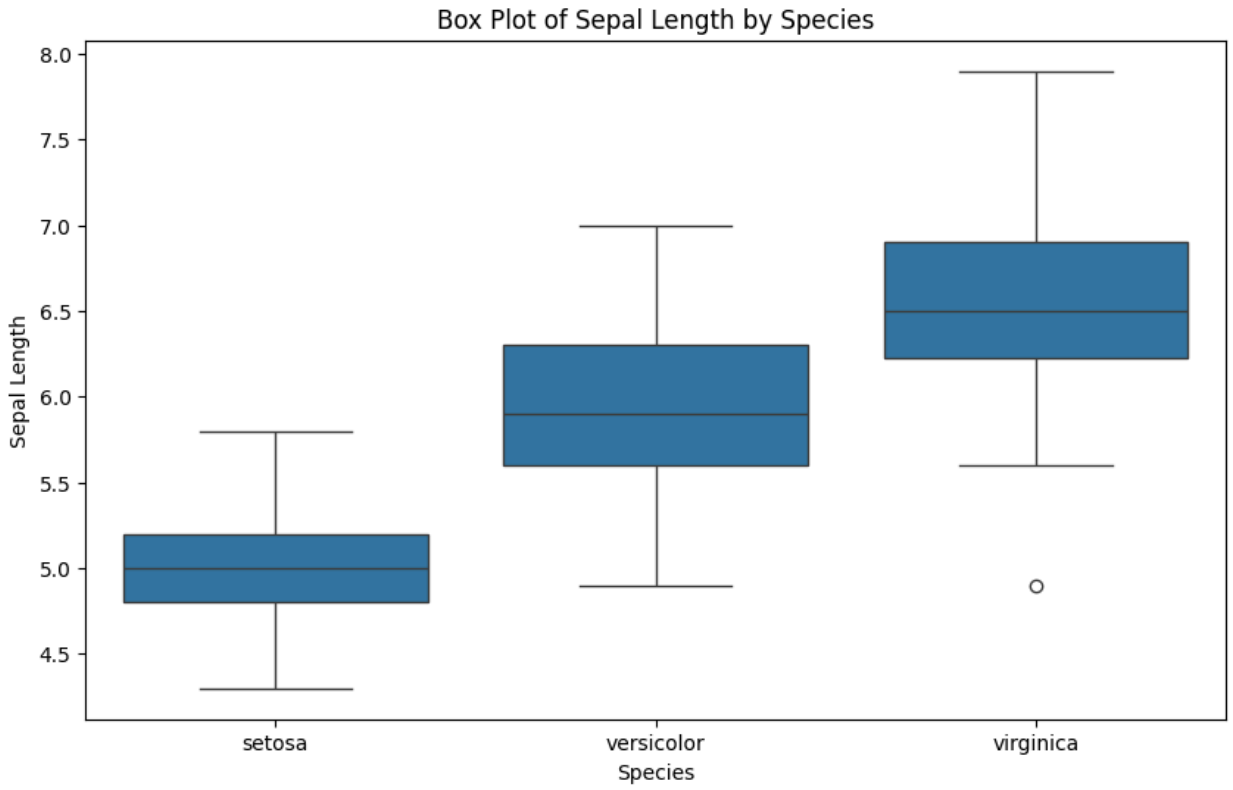
```
Dataset Head:
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

Figure(i)

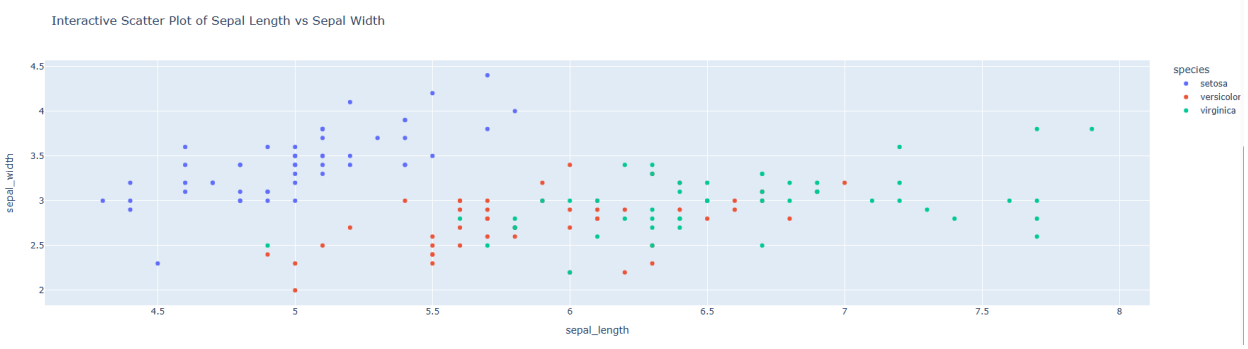




Figure(ii and iii)



Figure(iv)



Figure(v)

Practical 6

Aim: Perform encoding of categorical variable in given data set.

[52]:
```
import pandas as pd
import numpy as np
import sklearn
```

[53]:
```
df= pd.read_csv(r"E:\A sem 6\DSV\data.csv")
df
```

[53]:

	Country	Age	Salary	Purchased
0	India	40.0	72000.0	No
1	China	32.0	NaN	Yes
2	Japan	33.0	41000.0	No
3	Spain	44.0	30000.0	No
4	Japan	28.0	NaN	Yes
5	India	41.0	12000.0	No
6	China	NaN	60000.0	Yes
7	India	42.0	90000.0	Yes
8	Japan	50.0	83000.0	No
9	India	NaN	69000.0	No

[54]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        8 non-null      float64
 2   Salary     8 non-null      float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

[55]:
```
x= df.iloc[:,:-1].values
x
```

[55]:
```
array([['India', 40.0, 72000.0],
       ['China', 32.0, nan],
       ['Japan', 33.0, 41000.0],
       ['Spain', 44.0, 30000.0],
       ['Japan', 28.0, nan],
       ['India', 41.0, 12000.0],
       ['China', nan, 60000.0],
       ['India', 42.0, 90000.0],
       ['Japan', 50.0, 83000.0],
       ['India', nan, 69000.0]], dtype=object)
```

[56]:
```
y=df.iloc[:,3].values
y
```

[56]:
```
array(['No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No'],
      dtype=object)
```

[57]:
```
df['Age']= df['Age'].fillna(value=df['Age'].mean())
df['Salary']= df['Salary'].fillna(value=df['Salary'].median())
df
```

[57]:

	Country	Age	Salary	Purchased
0	India	40.00	72000.0	No
1	China	32.00	64500.0	Yes
2	Japan	33.00	41000.0	No
3	Spain	44.00	30000.0	No
4	Japan	28.00	64500.0	Yes
5	India	41.00	12000.0	No
6	China	38.75	60000.0	Yes
7	India	42.00	90000.0	Yes
8	Japan	50.00	83000.0	No
9	India	38.75	69000.0	No

Handling missing values using sklearn

[58]:
```
from sklearn.impute import SimpleImputer
im= SimpleImputer(missing_values= np.nan, strategy='mean')
im.fit(x[:,1:3])
x[:,1:3]= im.transform(x[:,1:3])
x
```

[58]:
```
array([['India', 40.0, 72000.0],
       ['China', 32.0, 57125.0],
       ['Japan', 33.0, 41000.0],
       ['Spain', 44.0, 30000.0],
       ['Japan', 28.0, 57125.0],
       ['India', 41.0, 12000.0],
       ['China', 38.75, 60000.0],
       ['India', 42.0, 90000.0],
       ['Japan', 50.0, 83000.0],
       ['India', 38.75, 69000.0]], dtype=object)
```

Encoding Categorical Data

Independent variable - Country - 3 values

[59]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct= ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0])], remainder='passthrough')
x= np.array(ct.fit_transform(x))
print(x)
```

```
[[0.0 1.0 0.0 0.0 40.0 72000.0]
 [1.0 0.0 0.0 0.0 32.0 57125.0]
 [0.0 0.0 1.0 0.0 33.0 41000.0]
 [0.0 0.0 0.0 1.0 44.0 30000.0]
 [0.0 0.0 1.0 0.0 28.0 57125.0]
 [0.0 1.0 0.0 0.0 41.0 12000.0]
 [1.0 0.0 0.0 0.0 38.75 60000.0]
 [0.0 1.0 0.0 0.0 42.0 90000.0]
 [0.0 0.0 1.0 0.0 50.0 83000.0]
 [0.0 1.0 0.0 0.0 38.75 69000.0]]
```

Purchased variable- 2 values - LabelEncoder

[60]:
```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
y= le.fit_transform(y)
print(y)
```

```
[0 1 0 0 1 0 1 1 0 0]
```

[ ]:

splitting Dataset

[61]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x,y, test_size= 0.3, random_state=42)
x_train
```

[61]:
```
array([[0.0, 1.0, 0.0, 0.0, 40.0, 72000.0],
       [0.0, 1.0, 0.0, 0.0, 42.0, 90000.0],
       [0.0, 0.0, 1.0, 0.0, 33.0, 41000.0],
       [0.0, 1.0, 0.0, 0.0, 38.75, 69000.0],
       [0.0, 0.0, 1.0, 0.0, 28.0, 57125.0],
       [0.0, 0.0, 0.0, 1.0, 44.0, 30000.0],
       [1.0, 0.0, 0.0, 0.0, 38.75, 60000.0]], dtype=object)
```

```
[62]: x_test
```

```
[62]: array([[0.0, 0.0, 1.0, 0.0, 50.0, 83000.0],
             [1.0, 0.0, 0.0, 0.0, 32.0, 57125.0],
             [0.0, 1.0, 0.0, 0.0, 41.0, 12000.0]], dtype=object)
```

```
[63]: y_train
```

```
[63]: array([0, 1, 0, 0, 1, 0, 1])
```

```
[64]: y_test
```

```
[64]: array([0, 1, 0])
```

Feature Scalling- Standardization

```
[65]: from sklearn.preprocessing import StandardScaler
      sc= StandardScaler()
      x_train[:,3:]= sc.fit_transform(x_train[:,3:])
      x_test[:,3:]= sc.transform(x_test[:,3:])
      x_train
```

```
[65]: array([[0.0, 1.0, 0.0, -0.40824829046386296, 0.43449168830662765,
              0.6556235153527858],
             [0.0, 1.0, 0.0, -0.40824829046386296, 0.826935793873904,
              1.6289202804125917],
             [0.0, 0.0, 1.0, -0.40824829046386296, -0.9390626811788397,
              -1.020609802250213],
             [0.0, 1.0, 0.0, -0.40824829046386296, 0.1892141223270799,
              0.49340738784281823],
             [0.0, 0.0, 1.0, -0.40824829046386296, -1.9201729450970308,
              -0.148698116884137],
             [0.0, 0.0, 0.0, 2.4494897427831783, 1.2193798994411804,
              -1.615402269786761],
             [1.0, 0.0, 0.0, -0.40824829046386296, 0.1892141223270799,
              0.006759005312915318]], dtype=object)
```

Practical 7

Aim:- Study and Introduction to data visualization setup tools.

What is Data Visualization?

Graphical representation of any information or data is known as Data Visualization. This helps in segregating the data in an efficient manner by using various types of visuals such as graphs, maps, charts, maps, and visualization tools. In addition to this, with the help of a data visualization tool, the data can be presented in a very unique and understandable manner so that people who are not from a technical background can understand everything easily.

What are Data Visualization Tools?

Data Visualization Tools are software platforms that provide information in a visual format such as a graph, chart, etc to make it easily understandable and usable. Data Visualization tools are so popular as they allow analysts and statisticians to create visual data models easily according to their specifications by conveniently providing an interface, database connections, and Machine Learning tools all in one place! Visualization Tools:

1. Tableau

Tableau is a data visualization tool that can be used by data analysts, scientists, statisticians, etc. to visualize the data and get a clear opinion based on the data analysis. Tableau is very famous as it can take in data and produce the required data visualization output in a very short time. And it can do this while providing the highest level of security with a guarantee to handle security issues as soon as they arise or are found by users.

Tableau also allows its users to prepare, clean, and format their data and then create data visualizations to obtain actionable insights that can be shared with other users. Tableau is available for individual data analysts or at scale for business teams and organizations. It provides a 14-day free trial followed by the paid version.

2. Looker

Looker is a data visualization tool that can go in-depth into the data and analyze it to obtain useful insights. It provides real-time dashboards of the data for more in-depth analysis so that businesses can make instant decisions based on the data visualizations obtained. Looker also provides connections with Redshift, Snowflake, and BigQuery, as well as more than 50 SQL-supported dialects so you can connect to multiple databases without any issues.

Looker data visualizations can be shared with anyone using any particular tool. Also, you can export these files in any format immediately. It also provides customer support wherein you can ask any question and it shall be answered. A price quote can be obtained by submitting a form.

3. Zoho Analytics

Zoho Analytics is a Business Intelligence and Data Analytics software that can help you create wonderful-looking data visualizations based on your data in a few minutes. You can obtain data

from multiple sources and mesh it together to create multidimensional data visualizations that allow you to view your business data across departments. In case you have any questions, you can use Zia which is a smart assistant created using artificial intelligence, machine learning, and natural language processing.

Zoho Analytics allows you to share or publish your reports with your colleagues and add comments or engage in conversations as required. You can export Zoho Analytics files in any format such as Spreadsheet, MS Word, Excel, PPT, PDF, etc. The pricing options available for this software include a basic plan with approx. A$34.1/month billed yearly.

4. Sisense

Sisense is a business intelligence-based data visualization system and it provides various tools that allow data analysts to simplify complex data and obtain insights for their organization and outsiders. Sisense believes that eventually, every company will be a data-driven company and every product will be related to data in some way. Therefore it tries its best to provide various data analytics tools to business teams and data analytics so that they can help make their companies the data-driven companies of the future.

It is very easy to set up and learn Sisense. It can be easily installed within a minute and data analysts can get their work done and obtain results instantly. Sisense also allows its users to export their files in multiple formats such as PPT, Excel, MS Word, PDF, etc. Sisense also provides full-time customer support services whenever users face any issues. A price quote can be obtained by submitting a form.

5. IBM Cognos Analytics

IBM Cognos Analytics is an Artificial Intelligence-based business intelligence platform that supports data analytics among other things. You can visualize as well as analyze your data and share actionable insights with anyone in your organization. Even if you have limited or no knowledge about data analytics, you can use IBM Cognos Analytics easily as it interprets the data for you and presents you with actionable insights in plain language.

You can also share your data with multiple users if you want on the cloud and share visuals over email or Slack. You can also import data from various sources like spreadsheets, cloud, CSV files, or on-premises databases and combine related data sources into a single data module. IBM Cognos Analytics provides a free trial for 30 days followed by a plan Starting at approx. A$20.87 per month.

6. Qlik Sense

Qlik Sense is a data visualization platform that helps companies to become data-driven enterprises by providing an associative data analytics engine, sophisticated Artificial Intelligence system, and scalable multi-cloud architecture that allows you to deploy any combination of SaaS, on-premises, or a private cloud.

You can easily combine, load, visualize, and explore your data on Qlik Sense, no matter its size. All the data charts, tables, and other visualizations are interactive and instantly update

themselves according to the current data context. The Qlik Sense AI can even provide you with data insights and help you create analytics using just drag and drop. You can try Qlik Sense Business for free for 30 days and then move on to a paid version.

7. Domo

Domo is a business intelligence model that contains multiple data visualization tools that provide a consolidated platform where you can perform data analysis and then create interactive data visualizations that allow other people to easily understand your data conclusions. You can combine cards, text, and images in the Domo dashboard so that you can guide other people through the data while telling a data story as they go.

In case of any doubts, you can use their pre-built dashboards to obtain quick insights from the data. Domo has a free trial option so you can use it to get a sense of this platform before committing to it fully. In case of any customer service inquiries, Domo is always available from 7 AM to 6 PM from Monday to Friday and you can try it for free followed by the paid version.

8. Microsoft Power BI

Microsoft Power BI is a Data Visualization platform focused on creating a data-driven business intelligence culture in all companies today. To fulfill this, it offers self-service analytics tools that can be used to analyze, aggregate, and share data in a meaningful fashion.

Microsoft Power BI offers hundreds of data visualizations to its customers along with built-in Artificial Intelligence capabilities and Excel integration facilities. And all this is very pocket friendly at a $9.99 monthly price per user for the Microsoft Power BI Pro. It also provides you with multiple support systems such as FAQs, forums, and also live chat support with the staff.

9. Klipfolio

Klipfolio is a Canadian business intelligence company that provides one of the best data visualization tools. You can access your data from hundreds of different data sources like spreadsheets, databases, files, and web services applications by using connectors. Klipfolio also allows you to create custom drag-and-drop data visualizations wherein you can choose from different options like charts, graphs, scatter plots, etc.

Klipfolio also has tools you can use to execute complex formulas that can solve challenging data problems. You can obtain a free trial of 14 days followed by $49 per month for the basic business plan. In the case of customer inquiries, you can get help from the community forum or the knowledge forum.

10. SAP Analytics Cloud

SAP Analytics Cloud uses business intelligence and data analytics capabilities to help you evaluate your data and create visualizations in order to predict business outcomes. It also provides you with the latest modeling tools that help you by alerting you of possible errors in the data and categorizing different data measures and dimensions. SAP Analytics Cloud also suggests Smart Transformations to the data that lead to enhanced visualizations.

In case you have any doubts or business questions related to data visualization, SAP Analytics Cloud provides you with complete customer satisfaction by handling your queries using conversational artificial intelligence and natural language technology. You can try this platform for free for 30 days and after that pay $22 per month for the Business Intelligence package.

11. Yellowfin

Yellowfin is a worldwide famous analytics and business software vendor that has a well-suited automation product that is specially created for people who have to take decisions within a short period of time. This is an easy-to-use data visualization tool that allows people to understand things and act according to them in the form of collaboration, data storytelling, and stunning action-based dashboards.

Yellowfin provides complete customer satisfaction with its five core products which have been integrated properly in order to manage analytics properly across the whole enterprise. You can try this platform for free for 30 days and after that pay $250 per month for the paid package.

12. Whatagraph

Whatagraph is a seamless integration that provides marketing agencies with an easy and useful way of sharing or sending marketing campaign data with clients. With this platform, you can create the data in a way that the result is easy to understand and comprehend. This Data visualization tool has numerous customization options which can be picked virtually and help in creating reporting widgets or creating your own methods of presenting data.

Whatagraph also helps in comparing data of different marketing platforms and their performance in one single report. You can try this platform for free for 30 days and after that pay $199 per month for the paid package.

Practical 8

Aim:- Develop the different basic Graphical Shapes using HTML5 CANVAS.

The HTML5 canvas element can be used to draw graphics on the webpage via JavaScript. The canvas was originally introduced by Apple for the Mac OS dashboard widgets and to power graphics in the Safari web browser. Later it was adopted by the Firefox, Google Chrome and Opera. Now the canvas is a part of the new HTML5 specification for next generation web technologies.

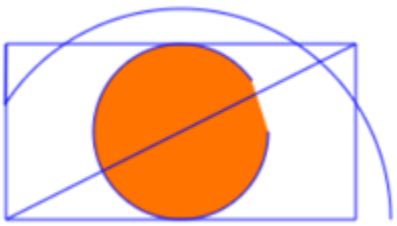
By default the <canvas> element has 300px of width and 150px of height without any border and content. However, custom width and height can be defined using the CSS height and width property whereas the border can be applied using the CSS border property.

Implementation:-

```
<html>
        <head>
        <script>
       window.onload=function(){
        var canvas= document.getElementById("myCanvas");
        var context= canvas.getContext("2d");
          context.arc(150,100,50, 2*Math.PI, 1.8*Math.PI, false);
          context.moveTo(50,150);
          context.lineTo(250,50);
          context.linewidth= 5;
          context.lineCap= "round";
          context.fillStyle= "#ff7300";
```

```
        context.fill();
        context.strokeStyle= "Blue";
        context.rect(50, 50, 200, 100);
      var grd= context.createRadialGradient(150, 100, 10, 160, 110, 100);
        grd.addColorStop(0,'#8ED6FF');
        grd.addColorStop(1,'#004CB3');
        context.fillStyle= grd;
        context.arc(150, 150, 120, 10, 2*Math.PI, false);
        context.font="bold 32px Arial";
        context.fillText("SAHAL", 100, 200);
        context.stroke();}
    </script>
      </head>
      <body>
      <div style="text-align: center; margin-top: 120px;" >
      <canvas id="myCanvas" width="300" height="200"></canvas>
      </div>
      </body> </html>
```

Result:



DIPAM

Practical 9

Aim:- Develop the different basic Graphical Shapes using SVG TAG.

SVG is a language for describing two-dimensional graphics. As a standalone format or when mixed with other XML, it uses the XML syntax. SVG code used inside HTML documents uses the HTML syntax. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects.

SVG drawings can be interactive and dynamic. Animations can be defined and triggered either declaratively (i.e., by embedding SVG animation elements in SVG content) or via scripting.

SVG contains the following set of basic shape elements:

●rectangles (including optional rounded corners), created with the ‘rect’ element,

● circles, created with the ‘circle’ element,

● ellipses, created with the ‘ellipse’ element,

● straight lines, created with the ‘line’ element,

● polylines, created with the ‘polyline’ element, and

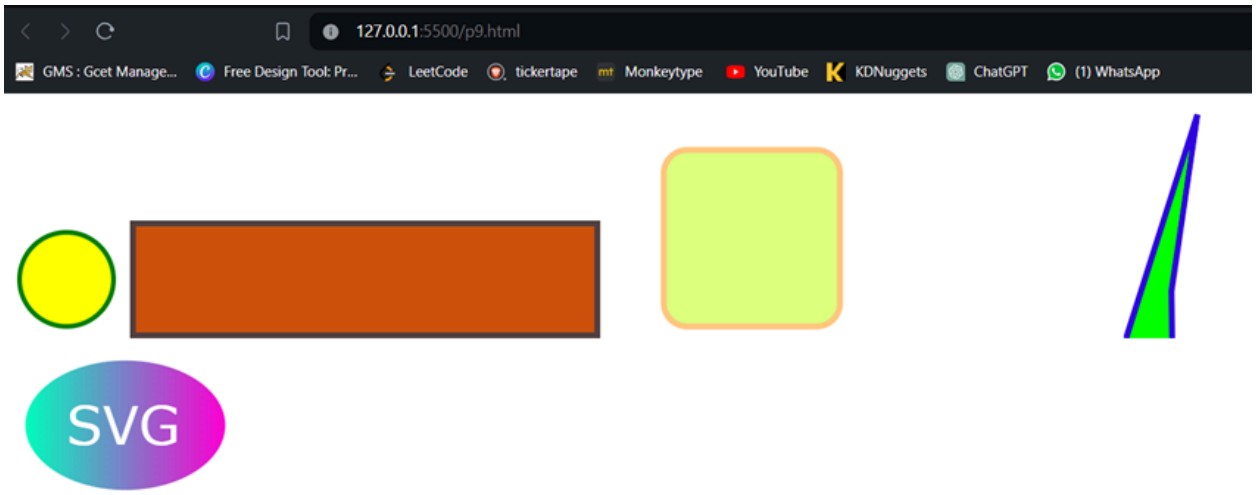
● polygons, created with the ‘polygon’ element

CODE:-

```
<html>
    <body>
```

```
<svg width="100" height="100">

<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

</svg>

<svg width="400" height="100">

<rect width="400" height="100" style="fill:rgb(207, 84, 13); stroke-width:10; stroke:rgb(82, 65, 65)" />

</svg>

<svg width="400" height="180">

<rect x="50" y="20" rx="20" ry="20" width="150" height="150" style="fill:rgb(187, 255, 0); stroke-width:5; stroke:rgb(255, 145, 0);opacity:0.5" />

</svg>

<svg width="300" height="200">

<polygon points="100,10 40,198 190,7810,78 160,198" style="fill:lime; stroke-width:5; stroke:rgb(49, 0, 228); fill-rule:evenodd;" />

</svg>

<svg width="500" height="130">

<defs>

<linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">

        <stop offset="0%" style="stop-color:rgb(0, 255, 191); stop-opacity:1 "/>

        <stop offset="100%" style="stop-color:rgb(255, 0, 212); stop-opacity:1 "/>

</linearGradient>

</defs>


<ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
```

```
<text fill="#ffffff" font-size="45" font-family="Verdana" x="50" y="86"> SVG </text>

Sorry, your browser does not support inline SVG.

</svg>

</body>

</html>
```



Practical 10

Aim:- Develop the simple bar chart using HTML5 CANVAS.

What Is a Bar Chart?

Bar charts are very common tools used to represent numerical data. From financial reports to PowerPoint presentations to infographics, bar charts are used very often since they offer a view of numerical data that is very easy to understand.

Bar charts represent numerical data using bars, which are rectangles with either their widths or heights proportional to the numerical data that they represent.

There are many types of bar charts, for example:

● horizontal bar charts and vertical bar charts depending on the chart orientation

● stacked bar charts or classic bar charts for representing multiple series of data

● 2D or 3D bar charts

What Are the Components of a Bar Chart?

● The chart data: these are sets of numbers and associated categories which are represented by the chart.

● Name of the data series (1).

● The chart grid (2): gives a reference system so that the visual representation can be easily understood.

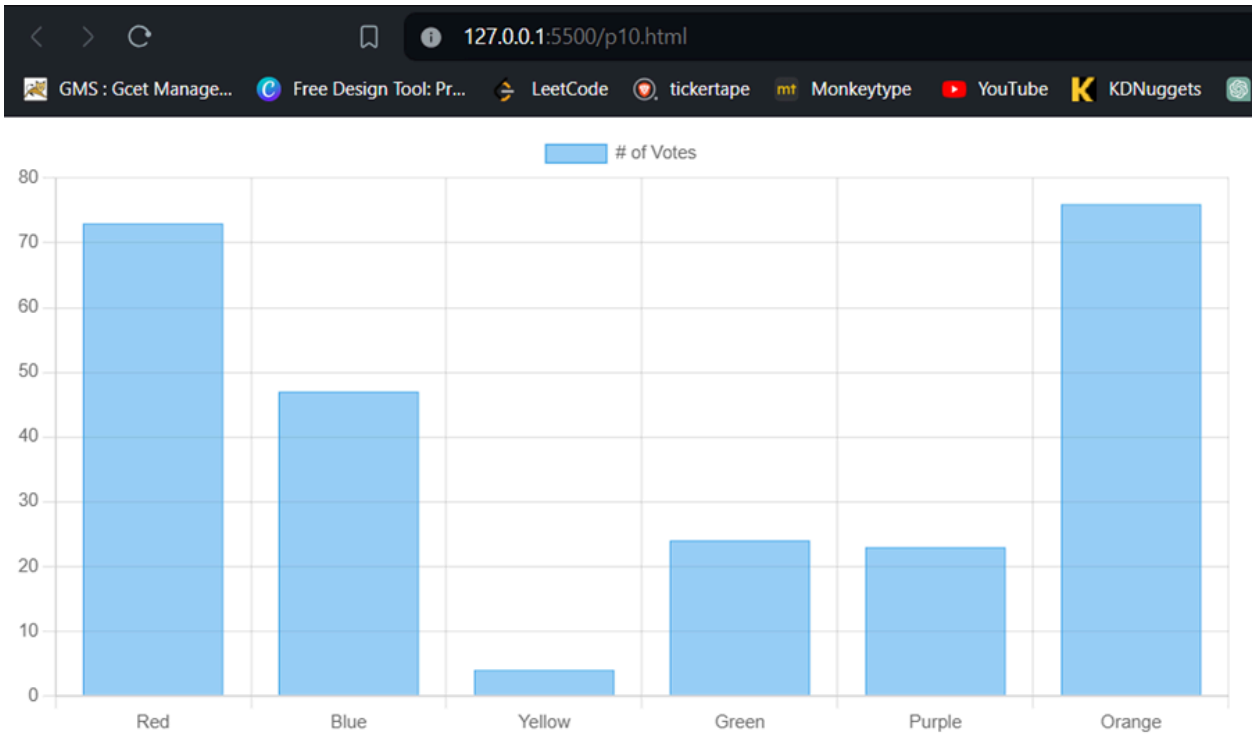
● The bars (3): color-filled rectangles with dimensions proportional to the data represented.

● Chart legend (4): shows the correspondence between the colors used and the data they represent.

IMPLEMENTATION:-

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Bar Chart Example</title>
<style>
        #chart-container {
        width: 800px; }
</style>
</head>
<body>
<div id="chart-container">
        <canvas id="myChart"></canvas>
</div>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
   document.addEventListener("DOMContentLoaded", function() {
        const ctx = document.getElementById('myChart').getContext('2d');
        new Chart(ctx, {
        type: 'bar',
        data: {
        labels: ['Red', 'Blue', 'Yellow', 'Green', 'Purple', 'Orange'],
        datasets: [{
```

```
                label: '# of Votes',
                data: [73, 47, 4, 24, 23, 76],
                borderWidth: 1
        }]},
        options: { scales: {
                y: {
                beginAtZero: true
                }}}});
        });
</script>
</body>
</html>
```



Practical 11

Aim:- Case study on Agriculture.

Introduction:-

Agriculture is a vital sector for global food security and economic development. In recent years, the integration of data science techniques has transformed traditional farming practices, leading to improved crop yields, resource efficiency, and sustainability. This case study explores how data science is revolutionizing agriculture through data-driven decision-making and innovative technologies.

Background:-

The agricultural sector generates vast amounts of data, including soil composition, weather patterns, crop yields, and market trends. Leveraging this data effectively can optimize farming operations, mitigate risks, and enhance productivity. In this case study, we focus on a fictional farm located in the Midwest region of the United States, specializing in corn and soybean production.

Objective:-

The primary objective of this case study is to demonstrate how data science techniques can be applied to address key challenges faced by farmers, such as optimizing crop production, managing resources efficiently, and adapting to changing environmental conditions.

Methodology:-

1. Data Collection: The first step involves collecting diverse datasets relevant to agricultural operations, including soil data, weather forecasts, satellite imagery, historical crop yields, and market prices.

2. Data Preprocessing: The raw data collected from various sources may require cleaning, normalization, and integration to ensure consistency and accuracy.

3. Predictive Modeling: Using machine learning algorithms, we develop predictive models to forecast key variables such as crop yields, pest infestations, and optimal planting times. These models take into account factors such as soil quality, weather patterns, and historical data.

4. Precision Agriculture: Utilizing IoT sensors, drones, and satellite imagery, we implement precision agriculture techniques to monitor crop health, detect anomalies, and optimize resource allocation. This enables targeted interventions such as irrigation scheduling, fertilization, and pest control.

5. Decision Support Systems: We develop decision support systems (DSS) that provide farmers with real-time insights and recommendations based on the analysis of large-scale agricultural data. These systems empower farmers to make informed decisions regarding planting strategies, crop rotations, and market timing.

Results:-

By integrating data science techniques into their operations, the fictional farm experiences significant improvements in productivity, profitability, and sustainability:

- Increased Crop Yields: Predictive models accurately forecast optimal planting times and crop management strategies, resulting in higher yields and improved quality.

- Resource Efficiency: Precision agriculture techniques enable targeted application of water, fertilizers, and pesticides, reducing waste and minimizing environmental impact.

- Risk Mitigation: Early detection of pest outbreaks and adverse weather conditions allows farmers to take proactive measures to protect their crops and minimize losses.

- Market Insights: Data-driven market analysis helps farmers identify lucrative opportunities, optimize pricing strategies, and diversify their product offerings.

Conclusion:-

In conclusion, data science holds immense potential to transform the agricultural sector by enabling data-driven decision-making, precision farming, and sustainable practices. By harnessing the power of big data, advanced analytics, and emerging technologies, farmers can overcome challenges, maximize productivity, and contribute to global food security in a rapidly changing world.

References:-

- Precision Agriculture: Principles and Applications - Pierre Robert and Robert L. Rust

- Data-Driven Agriculture: Technology and Applications - Qin Zhang and Liping Jiang

- FAO Agricultural Data: http://www.fao.org/agriculture/agn/en/