EX.NO:1	DEMONSTRATE AGGREGATION
DATE:	

AIM:

To write a program to understand aggregate functions in python.

APPROACH:

Step 1: Open Jupiter notebook, create a new file ex01.
Step 2: Import Pandas library from python.
Step 3: Create a data with dictionary datatype in a variable and convert it into dataframe.
Step 4: Select two or more attributes and perform aggregation using sum, average, min, max and mean.
Step 5: Execute the program and display the output.

PROGRAM:

```
import pandas as pd
technologies = {
   'Courses':["Spark","PySpark","Hadoop","Python","PySpark","Spark"],
   'Fee' :[20000,25000,26000,22000,24000,3000],
   'Duration':['30day','40days','35days','40days','60days','60days'],
   'Discount':[1000,2300,1200,2500,2000,2000]}
df = pd.DataFrame(technologies)
print("DataFrame:")
print(df)
print()

# Using Aggregate Functions Sum
print("Aggregate Functions sum on Fee and Discount: ")
result = df[['Fee','Discount']].aggregate('sum')
print(result)
print()

# Using Aggregate Function Average
print("Aggregate Functions average on Fee and Discount: ")
result = df[['Fee','Discount']].aggregate('average')
print(result)
print()

# Using Aggregate Function MinMax
result = df.groupby('Courses')['Fee'].aggregate(['min','max'])
print(result)
print()

# Using Aggregate Function Mean
print("Aggregate Functions mean value on Fee and Discount: ")
result = df[["Fee","Discount"]].mean()
print(result)
```

OUTPUT:

```
DataFrame:
   Courses    Fee Duration  Discount
0    Spark  20000    30day      1000
1  PySpark  25000   40days      2300
2   Hadoop  26000   35days      1200
3   Python  22000   40days      2500
4  PySpark  24000   60days      2000
5    Spark   3000   60days      2000

Aggregate Functions sum on Fee and Discount:
Fee         120000
Discount     11000
dtype: int64

Aggregate Functions average on Fee and Discount:
10916.666666666666

           min    max
Courses
Hadoop   26000  26000
PySpark  24000  25000
Python   22000  22000
Spark     3000  20000

Aggregate Functions mean value on Fee and Discount:
Fee         20000.000000
Discount     1833.333333
dtype: float64
```

RESULT:

EX.NO:1(a)	DISPLAY AGGREGATES AND VALUES
DATE:	

AIM:

To write a Python program that calculates and displays the mean, median, and sum of a given list of numbers.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:2	DEMONSTRATE INDEXING AND SORTING
DATE:	

AIM:

To write a program to understand indexing and sorting in dataframes using python.

APPROACH:

Step 1: Open Jupiter notebook, create a new file ex02.
Step 2: Import Pandas library from python.
Step 3: Create a data with dictionary datatype in a variable and convert it into dataframe.
Step 4: Using sort_values() the dataframe is sorted.
Step 5: Using reset_index() the sorted dataframe is indexed.
Step 6: Execute the program and display the output.

PROGRAM:

```
import pandas as pd
technologies = {
   'Courses':["Spark","PySpark","Hadoop","Python","PySpark","Spark"],
   'Fee' :[20000,25000,26000,22000,24000,3000],
   'Duration':['30day','40days','35days','40days','60days','60days'],
   'Discount':[1000,2300,1200,2500,2000,2000]
           }
#before sorting and indexing
df = pd.DataFrame(technologies)
print(df)
print()

# indexing
print("Indexed dataframe:")
df.index.name = 'Index'
print(df)
print()
# after sorting
print("Sorted dataframe: ")
sorted_df = df.sort_values(by = ["Courses"])
print(sorted_df)
print()
```

OUTPUT:

```
   Courses    Fee Duration  Discount
0    Spark  20000    30day      1000
1  PySpark  25000   40days      2300
2   Hadoop  26000   35days      1200
3   Python  22000   40days      2500
4  PySpark  24000   60days      2000
5    Spark   3000   60days      2000

Indexed dataframe:
       Courses    Fee Duration  Discount
Index
0        Spark  20000    30day      1000
1      PySpark  25000   40days      2300
2       Hadoop  26000   35days      1200
3       Python  22000   40days      2500
4      PySpark  24000   60days      2000
5        Spark   3000   60days      2000

Sorted dataframe:
       Courses    Fee Duration  Discount
Index
2       Hadoop  26000   35days      1200
1      PySpark  25000   40days      2300
4      PySpark  24000   60days      2000
3       Python  22000   40days      2500
0        Spark  20000    30day      1000
5        Spark   3000   60days      2000
```

RESULT:

EX.NO:2(A)	DATAFRAME OPERATIONS
DATE:	

AIM:

To write a Python program to create a DataFrame from a dictionary, then index it by a specific column and sort the DataFrame based on that column.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:3	DEMONSTRATE HANDLING OF MISSING DATA
DATE:	

AIM:

To write a program to handle missing data in a CSV file using python.

APPROACH:

1. Load the Dataset.
2. Read the dataset from a CSV file into a Pandas DataFrame.
3. Identify Missing Data.
4. Handle Missing Data.
5. Analyze the Cleaned Data.
6. Output and Result.

PROGRAM:

```
import pandas as pd

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

missing_data = df.isnull()
data_info = df.info()

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

print("Cleaned Dataset:")
print(df)

print("Missing Data Information:")
print(missing_data)
print("Dataset Information:")
print(data_info)
```

OUTPUT:

Output of Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):

#	Column	Non-Null	Count	Datatype
0	1	Yes	4	Float64
1	2	Yes	3	Int32
2	3	yes	4	Int16

dtypes: float64(3)
memory usage: 224.0 bytes

RESULT:

EX.NO:3(a)	DATAFRAME REFINING
DATE:	

AIM:

To Write a Python program that reads a CSV file into a DataFrame and displays the count of missing values for each column in the DataFrame.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:4	DEMONSTRATE HIERARCHICAL INDEXING
DATE:	

AIM:

To write a program to understand hierarchical indexing in csv file using python.

APPROACH:

Step 1: Open Jupiter notebook, create a new file ex04.
Step 2: Import Pandas library from python.
Step 3: Upload the csv file in the current directory.
Step 4: Read the content of csv file using read_csv().
Step 5: Set multiple levels of indexing using function set_index().
Step 6: Execute the program and display the output.

PROGRAM:

```
import pandas as pd

df = pd.read_csv('homelessness.csv')
print(df.head()) # first 10 rows from dataframe
print()

# using the pandas set_index() function.
indexed_df = df.set_index(['region', 'state', 'individuals'])

print(indexed_df.head())
```

OUTPUT:

```
                region       state  individuals  family_members  state_pop
0  East South Central     Alabama         2570             864    4887681
1             Pacific      Alaska         1434             582     735139
2            Mountain     Arizona         7259            2606    7158024
3  West South Central    Arkansas         2280             432    3009733
4             Pacific  California       109008           20964   39461588

                                        family_members  state_pop
region             state      individuals
East South Central Alabama    2570                 864    4887681
Pacific            Alaska     1434                 582     735139
Mountain           Arizona    7259                2606    7158024
West South Central Arkansas   2280                 432    3009733
Pacific            California 109008              20964   39461588
```

RESULT:

EX.NO:4(a)	DATAFRAME INDEXING AND DISPLAYING
DATE:	

AIM:

To Write a Python program to read a CSV file with hierarchical index columns and display the data using hierarchical indexing.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:5	DEMONSTRATE USAGE OF PIVOT TABLE
DATE:	

AIM:

To write a program to create a pivot table of dataframe using python.

APPROACH:

Step 1: Open Jupiter notebook, create a new file ex05.
Step 2: Import Pandas library from python.
Step 3: Import Numpy library from python.
Step 4: Create a data with dictionary datatype in a variable and convert it into dataframe.
Step 5: A pivot table is created using pivot_table() function.
Step 6: Basic aggregation operations are performed on the pivot table.
Step 7: Execute the program and display the output.

PROGRAM:

```
import pandas as pd
import numpy as np

#Pandas pivot tables provide a powerful tool to perform these analysis techniques with
python.
#Creating own dataframe or load a dataset
df = pd.DataFrame({"A": ["burger", "burger", "burger", "burger", "burger","fries", "fries",
"fries", "fries"],
                   "B": ["one", "one", "one", "two", "two","one", "one", "two", "two"],
                   "C": ["small", "large", "large", "small","small", "large", "small", "small","large"],
                   "D": [1, 2, 2, 3, 3, 4, 5, 6, 7],
                   "E": [2, 4, 5, 5, 6, 6, 8, 9, 9]})
print("Dataframe:")
print(df)
print()
#CREATING BASIC PIVOT TABLE
print("BASIC PIVOT TABLE")
table = pd.pivot_table(df, index=['A', 'B'])
print(table)
#SOME BASIC AGGREGATE OPERATION USAGE USING PIVOT
print("SOME BASIC AGGREGATE  OPERATIONS")
table = pd.pivot_table(df, index=['A', 'B'], aggfunc=np.sum)
print(table)
print()

#mean
table = pd.pivot_table(df, values=['D', 'E'], index=['A', 'C'],
                       aggfunc={'D': np.mean,
                                'E': np.mean})
print(table)
print(0)
```

```
#multiple aggregations with pivot
table = pd.pivot_table(df, values=['D', 'E'], index=['A', 'C'],
                       aggfunc={'D': np.mean,
                                'E': [min, max, np.mean]})
print(table)
```

OUTPUT:

```
Dataframe:
         A    B      C  D  E
0  burger  one  small  1  2
1  burger  one  large  2  4
2  burger  one  large  2  5
3  burger  two  small  3  5
4  burger  two  small  3  6
5   fries  one  large  4  6
6   fries  one  small  5  8
7   fries  two  small  6  9
8   fries  two  large  7  9

BASIC PIVOT TABLE
                   D         E
A      B
burger one  1.666667  3.666667
       two  3.000000  5.500000
fries  one  4.500000  7.000000
       two  6.500000  9.000000
SOME BASIC AGGREGATE  OPERATIONS
             D   E
A      B
burger one   5  11
       two   6  11
fries  one   9  14
       two  13  18

                     D         E
A      C
burger large  2.000000  4.500000
       small  2.333333  4.333333
fries  large  5.500000  7.500000
       small  5.500000  8.500000
0
                     D    E
                  mean  max      mean  min
A      C
burger large  2.000000  5.0  4.500000  4.0
       small  2.333333  6.0  4.333333  2.0
fries  large  5.500000  9.0  7.500000  6.0
       small  5.500000  9.0  8.500000  8.0
```

RESULT:

EX.NO:5(a)	PIVOT TABLE CREATION
DATE:	

AIM:

To Write a Python program to create a pivot table from a given DataFrame, aggregating values based on specified rows and columns.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:6a	DEMONSTRATE USE OF EVAL()
DATE:	

AIM:

To write a program to perform eval operation using python.

APPROACH:

Step 1: Open Jupiter notebook, create a new file ex06.
Step 2: Import Pandas library from python.
Step 3: Import NumPy library from python.
Step 4: Get the expression as input from user.
Step 5: Get the values of the variables used in the expression from the user.
Step 6: Then the expression is evaluated using eval() function.
Step 7: Execute the program and display the output.

PROGRAM:

```
import numpy as np
import pandas as pd

# expression to be evaluated
expr = input("Enter the function(in terms of a and b):")

# variable used in expression
a = int(input("Enter the value of a:"))
b = int(input("Enter the value of b:"))

# evaluating expression
res= eval(expr)

# printing evaluated result
print("Result=", res)
```

OUTPUT:

```
Enter the function(in terms of a and b):(a*a)+(b*b)+2*a*b
Enter the value of a:2
Enter the value of b:4
Result= 36
```

RESULT:

EX.NO:6b	DEMONSTRATE USE OF QUERY()
DATE:	

AIM:

To write a program to perform query operation using python.

APPROACH:

Step 1: Open Jupiter notebook, create a new file ex06.
Step 2: Import Pandas library from python.
Step 3: Import NumPy library from python
Step 4: Create a data with dictionary datatype in a variable and convert it into dataframe.
Step 5: Write a query according to the given statement and process using query() function. Store the result set in a separate variable.
Step 6: Execute the program and display the output.

PROGRAM:

```
import numpy as np
import pandas as pd

dataFrame = pd.DataFrame({'Name': ['RACHEL', 'MONICA', 'PHOEBE','ROSS',
'CHANDLER', 'JOEY'],
                    'Age': [30, 35, 37, 23, 24, 30],
                    'Salary': [100000, 93000, 88000, 120000, 94000, 95000],
                    'JOB': ['DESIGNER', 'CHEF', 'MASUS', 'PALENTOLOGY', 'IT',
'ARTIST']})
print('Dataframe:')
print(dataFrame)
print()

#Basic  query
print('To print age below 30')
res=dataFrame.query('Age<30')
print(res)
print()

#multiple query
print('To print salary <=100000 and age<40 and job name starts with c')
res=dataFrame.query('Salary  <= 100000 & Age < 40 & JOB.str.startswith("C").values')
print(res)
```

OUTPUT:

```
Dataframe:
       Name  Age  Salary          JOB
0    RACHEL   30  100000     DESIGNER
1    MONICA   35   93000         CHEF
2    PHOEBE   37   88000        MASUS
3      ROSS   23  120000  PALENTOLOGY
4  CHANDLER   24   94000           IT
5      JOEY   30   95000       ARTIST

To print age below 30
       Name  Age  Salary          JOB
3      ROSS   23  120000  PALENTOLOGY
4  CHANDLER   24   94000           IT

To print salary <=100000 and age<40 and job name starts with c
     Name  Age  Salary   JOB
1  MONICA   35   93000  CHEF
```

RESULT:

EX.NO:6(c)	EVALUATE USER INPUTS
DATE:	

AIM:

To Write a Python program that uses the eval() function to perform a mathematical operation on user-input numbers.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:7a	DEMONSTRATE SCATTER PLOT
DATE:	

AIM:

To write a program to display the dataset in a scatter plot using python.

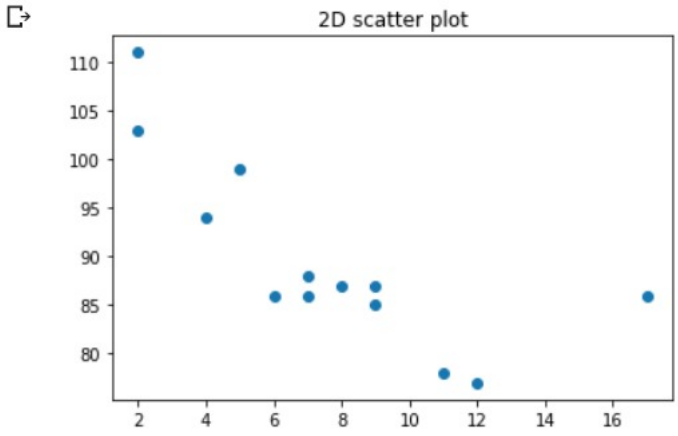
APPROACH:

Step 1: Open Jupiter notebook, create a new file ex07.
Step 2: Import pyplot from matplotlib.
Step 3: Create the coordinates values for x and y axis.
Step 4: Using scatter() function set x and y as parameters.
Step 5: Using title() to set the title and show() to display the graph.
Step 6: Execute the program and display the output.

PROGRAM:

```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np
#A scatter plot is a diagram where each value in the data set is represented by a dot.
#The Matplotlib module has a method for drawing scatter plots
#BASIC 2D X,Y PLOT
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.title("2D scatter plot")
plt.show()
```

OUTPUT:



RESULT:

EX.NO:7b	DEMONSTRATE 3D PLOTTING
DATE:	

AIM:

To write a program to display the dataset in a 3D plot using python.

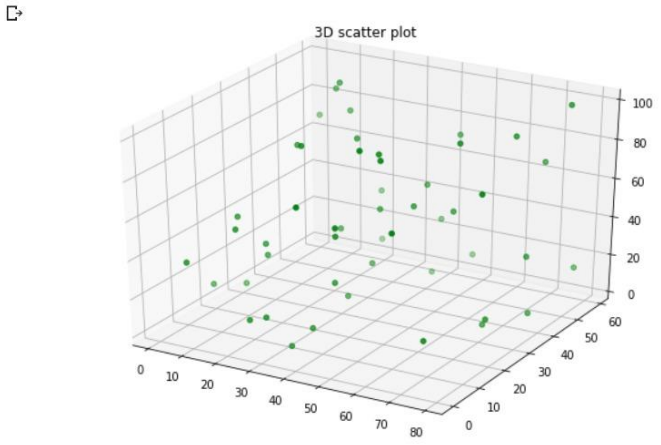
APPROACH:

Step 1: Open Jupiter notebook, create a new file ex07.
Step 2: Import pyplot from matplotlib.
Step 3: Create 3 coordinates x, y, z and set the start value and size of the points which should be plotted.
Step 4: Create figure of the graph and axes projection using the function figure() and axes().
Step 5: Using scatter3D() function set x, y and z as parameters.
Step 6: Using title() to set the title and show() to display the graph.
Step 7: Execute the program and display the output.

PROGRAM:

```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np


#3D scatter plot is used for 3d representation and is created by using ax.scatter3D()
# Creating dataset

z = np.random.randint(100, size =(50))
x = np.random.randint(80, size =(50))
y = np.random.randint(60, size =(50))

# Creating figure
fig = plt.figure(figsize = (10, 7))
ax = plt.axes(projection ="3d")

# Creating plot
ax.scatter3D(x, y, z, color = "green")
plt.title("3D scatter plot")

plt.show()
```

OUTPUT:



RESULT:

EX.NO:7(c)	EVALUATE USER QUERIES
DATE:	

AIM:

To Write a Python program that queries a given DataFrame based on user-defined conditions and displays the filtered results.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:8	IMPLEMENT AN ANALYTIC APPLICATION FOR INSTAGRAM TO DEMONSTRATE THE NUMBER OF LIKES, EMOTIONS.
DATE:	

AIM:

To write a program to implement an analytic application for instagram to demonstrate the number of likes, emotions using python.

APPROACH:

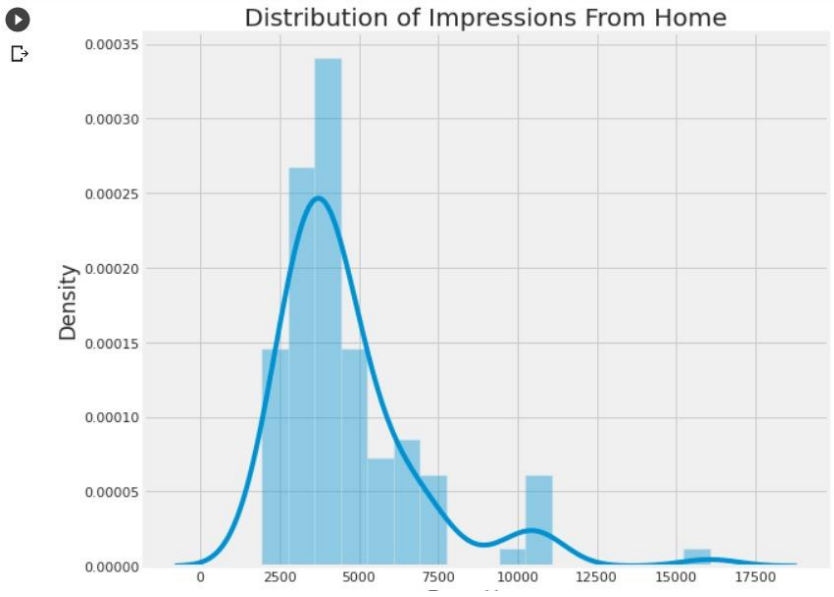
Step 1: Open Jupiter notebook, create a new file ex08.
Step 2: Import all necessary libraries to perform data analysis and visualization.
Step 3: Load the dataset into the current directory and read using read_csv().
Step 4: Group the associated attributes of the dataset and create a chart to get the pictorial representation of the data.
Step 5: Analyze the dataset using linear model and get the distribution of values over the attributes.
Step 6: Display a graphical representation of the distribution acquired from the analysis.
Step 7: Execute the program and display the output.

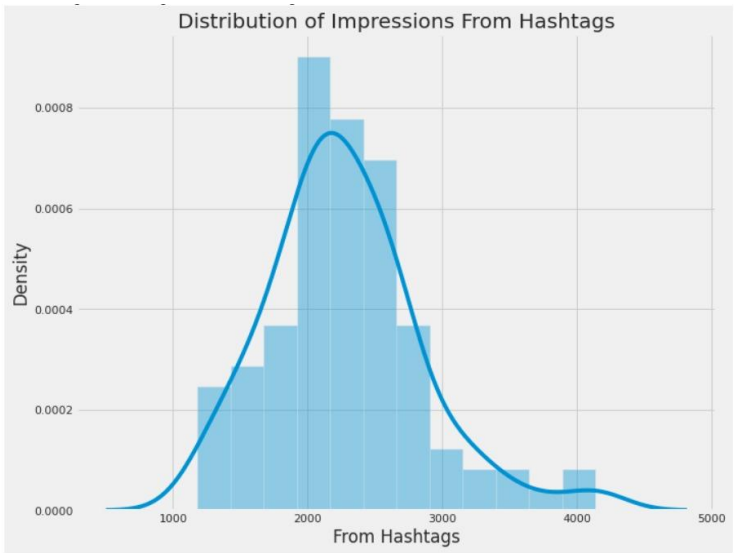
PROGRAM:

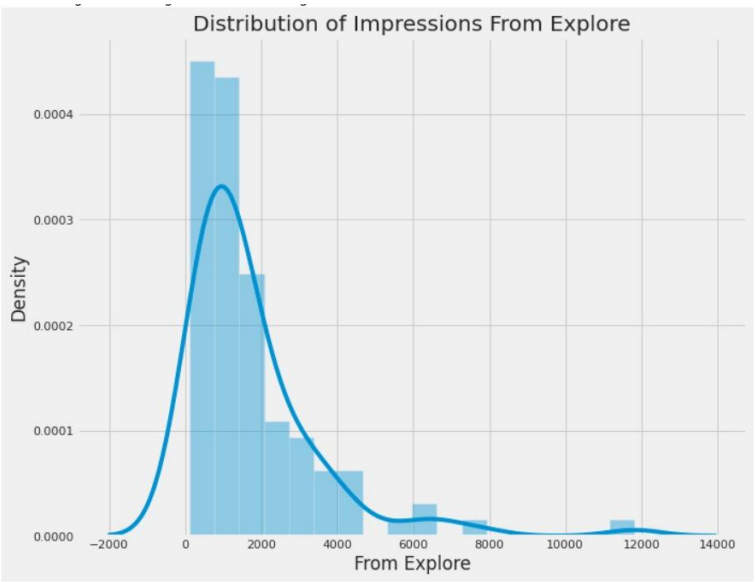
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from sklearn.model_selection import train_test_split
from sklearn.linear_model import PassiveAggressiveRegressor

data = pd.read_csv(r"Instagram.csv", encoding = 'latin1')
print(data.head())

plt.figure(figsize=(10, 8))
plt.style.use('fivethirtyeight')
plt.title("Distribution of Impressions From Home")
sns.distplot(data['From Home'])
plt.show()
plt.figure(figsize=(10, 8))
plt.title("Distribution of Impressions From Hashtags")
sns.distplot(data['From Hashtags'])
plt.show()
plt.figure(figsize=(10, 8))
plt.title("Distribution of Impressions From Explore")
sns.distplot(data['From Explore'])
plt.show()
home = data["From Home"].sum()
hashtags = data["From Hashtags"].sum()
explore = data["From Explore"].sum()
```

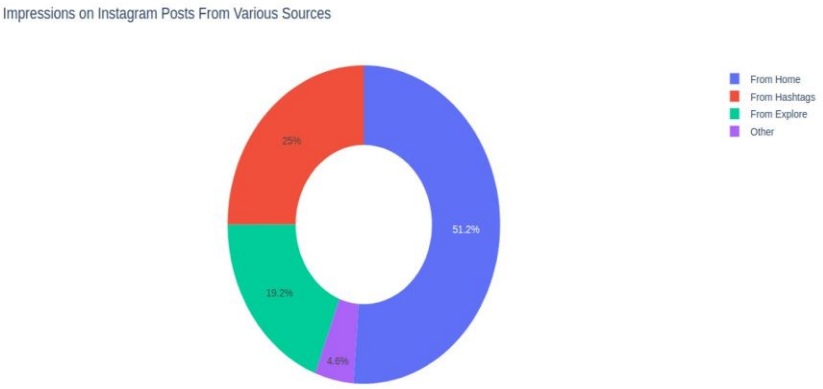
```
other = data["From Other"].sum()

labels = ['From Home','From Hashtags','From Explore','Other']
values = [home, hashtags, explore, other]

fig = px.pie(data, values=values, names=labels,
             title='Impressions on Instagram Posts From Various Sources', hole=0.5)
fig.show()
```

OUTPUT:




Distribution of Impressions From Hashtags
Density
0.0008
0.0006
0.0004
0.0002
0.0000
1000
2000
3000
4000
5000
From Hashtags


Distribution of Impressions From Explore
Density
0.0004
0.0003
0.0002
0.0001
0.0000
−2000
0
2000
4000
6000
8000
10000
12000
14000
From Explore


Impressions on Instagram Posts From Various Sources
From Home
From Hashtags
From Explore
Other
25%
51.2%
19.2%
4.6%

RESULT:

EX.NO:8(a)	DATAFRAME EXTRACTION
DATE:	

AIM:

To Write a Python program to read a dataset, extract two numeric columns, and create a scatter plot to visualize their relationship.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:9	IMPLEMENT AN ANALYTIC APPLICATION FOR TWITTER TO DEMONSTRATE SENTIMENT ANALYSIS AND ENTITY RECOGNITION.
DATE:	

AIM:

To write a program to implement an analytic application for twitter to demonstrate Sentiment Analysis and Entity Recognition using python.

APPROACH:

Step 1: Open Jupiter notebook, create a new file ex09.
Step 2: Import all necessary libraries to perform data analysis and visualization.
Step 3: Load the dataset into the current directory and read using read_csv().
Step 4: Group the associated attributes of the dataset and create a chart to get the pictorial representation of the data.
Step 5: Perform data cleaning by altering the non-value added attributes.
Step 6: Import machine learning model of sentiment analysis.
Step 7: Perform data processing by re, stemming, vectorization and split the dataset into training and testing.
Step 8: Get the analytics using sentiment analysis model and gather the total distribution.
Step 9: Display a graphical representation of the distribution acquired from the analysis.
Step 10: Execute the program and display the output.

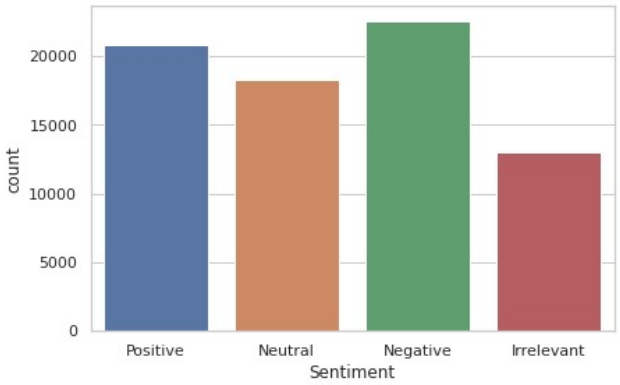
PROGRAM:

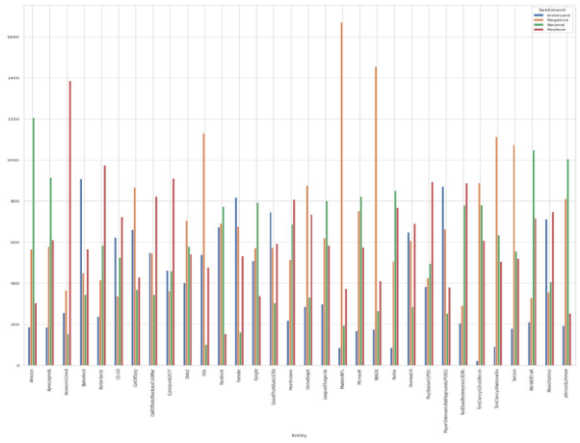
```
import numpy as np
import pandas as pd
import time
import re

from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer


#preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# machine learning

from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
```

```
from sklearn.metrics import classification_report

import seaborn as sns
sns.set(style ='whitegrid')
pd.set_option('display.max_columns',None)

headers=['Tweet_ID','Entity','Sentiment','Tweet_content']


train_df=pd.read_csv('twitter_training.csv', sep=',', names=headers)

valid_df=pd.read_csv('twitter_validation.csv', sep=',', names=headers)

train_df['Sentiment'].value_counts()
sns.countplot(x=train_df['Sentiment'])

train_df= train_df.drop_duplicates()

train_df.Sentiment.value_counts().plot(kind='pie',
autopct='%1.0f%%',figsize=(5,5),colors=["red", "yellow", "green",'blue'])

Twitter_sentiment = train_df.groupby(['Entity', 'Sentiment']).Sentiment.count().unstack()
Twitter_sentiment.plot(kind='bar',figsize=(20,20))

# SENTIMENTAL ANALYSIS #

# encoder for target feature
from sklearn import preprocessing
lb = preprocessing.LabelEncoder()
train_df['Sentiment']=lb.fit_transform(train_df['Sentiment'])

train_df.dropna(axis=0, inplace=True)
tweet_train  = train_df["Tweet_content"]
tweet_valid=valid_df["Tweet_content"]
target=train_df['Sentiment']

# Step (1): Remove Additional Letter such as
REPLACE_WITH_SPACE = re.compile("(@)")
SPACE = " "

def preprocess_reviews(reviews):
    reviews = [REPLACE_WITH_SPACE.sub(SPACE, line.lower()) for line in reviews]

    return reviews

reviews_train_clean = preprocess_reviews(tweet_train)
reviews_valid_clean = preprocess_reviews(tweet_valid)

# Step (2): Remove Stop Words
```

```
english_stop_words = stopwords.words('english')
def remove_stop_words(corpus):
    removed_stop_words = []
    for review in corpus:
        removed_stop_words.append(' '.join([word for word in review.split()  if word not in
english_stop_words]))
    return removed_stop_words

no_stop_words_train = remove_stop_words(reviews_train_clean)
no_stop_words_valid = remove_stop_words(reviews_valid_clean)


# Step(3) : Stemming
def get_stemmed_text(corpus):
    stemmer = PorterStemmer()

    return [' '.join([stemmer.stem(word) for word in review.split()]) for review in corpus]

stemmed_reviews_train = get_stemmed_text(no_stop_words_train)
stemmed_reviews_test = get_stemmed_text(no_stop_words_valid)

# Step(4) : TF-IDF
tfidf_vectorizer = TfidfVectorizer()
tfidf_vectorizer.fit(stemmed_reviews_train)
X = tfidf_vectorizer.transform(stemmed_reviews_train)
X_test = tfidf_vectorizer.transform(stemmed_reviews_test)

# Step(5) : Spliting Data
X_train, X_val, y_train, y_val = train_test_split(X, target, train_size = 0.75)


# Step(6) : Machine Learing Model
text_classifier = RandomForestClassifier(n_estimators=500, random_state=0)
text_classifier.fit(X_train, y_train)

y_pred=text_classifier.predict(X_val)
print(classification_report(y_val,y_pred))
```

OUTPUT:





	precision	recall	f1-score	support
0	0.96	0.88	0.92	723
1	0.92	0.94	0.93	1127
2	0.97	0.89	0.93	867
3	0.87	0.95	0.91	1071
accuracy			0.92	3788
macro avg	0.93	0.92	0.92	3788
weighted avg	0.92	0.92	0.92	3788

RESULT:

EX.NO:9(a)	3D PLOTTING
DATE:	

AIM:

To Write a Python program to create a 3D plot using a dataset with three numeric columns, demonstrating a three-dimensional visualization.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:10	SOCIAL MEDIA ANALYSIS
DATE:	

AIM:

To Write a Python program that collects and analyzes Instagram data from a given csv, including the number of likes and emotions, then displays the results.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT:

EX.NO:11	SENTIMENT ANALYSIS
DATE:	

AIM:

To Write a Python program that performs sentiment analysis and entity recognition on a set of Twitter data from a given csv, then displays the sentiment scores.

ALGORITHM:

PROGRAM:

OUTPUT:

CRITERIA	MAX.MARKS	MARKS OBTAINED
AIM & DESCRIPTION	15	
VIVA	10	
TOTAL	25	

RESULT: