**Lab 1**

**To demonstrate effective JSON parsing using python allowing users to extract structured data from a JSON file and utilize it for further processing or analysis. Parsing of Json file, accessing elements, updating data new elements, converting it to dataframe and perform basic analysis**

**Code:**

import json

with open("lab1.json","r") as file:

  data=json.load(file)

# Access an element

id= int(input("Enter the car id to access: "))

info= None

for car in data:

    if car["id"] == id:

        info= car

        break

if info:

    print("Car details:")

    print("Brand:", info["brand"])

    print("Model:", info["model"])

    print("Year:", info["year"])

    print("Color:", info["color"])

    print("Price:", info["price"])

else:

    print(f"No car found with id {id}")

**Output:**

Enter the car id to access: 7

Car details:

Brand: Honda

Model: Accord

Year: 2022

Color: ['Grey', 'White']

Price: 800000

**Code:**

#updation

for car in data:

    if car["id"] == 4:

        car["color"] = "Blue"

        car["price"] = 1800000

        break

for car in data:

    print(car)

**Output:**

{'id': 1, 'brand': 'Toyota', 'model': 'Corolla', 'year': 2018, 'color': ['Red', 'Silver', 'Black'], 'fuel\_type': 'petrol', 'price': 150000}

{'id': 2, 'brand': 'Honda', 'model': 'Civic', 'year': 2019, 'color': ['Black', 'Red'], 'fuel\_type': 'diesel', 'price': 180000}

{'id': 3, 'brand': 'Ford', 'model': None, 'year': 2023, 'color': ['Yellow', 'Blue'], 'fuel\_type': 'electric', 'price': None}

{'id': 4, 'brand': 'Chevrolet', 'model': 'Camaro', 'year': None, 'color': 'Blue', 'fuel\_type': 'petrol', 'price': 1800000}

{'id': 5, 'brand': 'Nissan', 'model': 'Altima', 'year': 2018, 'color': ['White', 'Black'], 'fuel\_type': 'diesel', 'price': 670000}

{'id': 6, 'brand': 'Toyota', 'model': 'RAV4', 'year': 2020, 'color': ['Silver', 'White', 'Red'], 'fuel\_type': 'electric', 'price': 930000}

{'id': 7, 'brand': 'Honda', 'model': 'Accord', 'year': 2022, 'color': ['Grey', 'White'], 'fuel\_type': 'electric', 'price': 800000}

{'id': 8, 'brand': 'Ford', 'model': 'F-150', 'year': 2018, 'color': ['Blue', 'Black'], 'fuel\_type': 'diesel', 'price': 789000}

{'id': 9, 'brand': 'Chevrolet', 'model': 'Silverado', 'year': 2019, 'color': ['Red', 'Grey'], 'fuel\_type': 'diesel', 'price': 5800000}

{'id': 10, 'brand': 'Nissan', 'model': 'Sentra', 'year': 2017, 'color': ['Black', 'White'], 'fuel\_type': 'petrol', 'price': 640000}

{'id': 11, 'brand': 'Toyota', 'model': 'Camry', 'year': 2019, 'color': ['White', 'Grey', 'Black'], 'fuel\_type': 'petrol', 'price': None}

{'id': 12, 'brand': 'Honda', 'model': 'CR-V', 'year': 2020, 'color': ['Silver', 'White', 'Red'], 'fuel\_type': 'electric', 'price': 890000}

{'id': 13, 'brand': 'Ford', 'model': None, 'year': 2018, 'color': ['Grey', 'White'], 'fuel\_type': 'diesel', 'price': 2400000}

{'id': 14, 'brand': 'Chevrolet', 'model': 'Impala', 'year': 2023, 'color': ['Blue', 'Black'], 'fuel\_type': 'electric', 'price': 520000}

{'id': 15, 'brand': 'Nissan', 'model': 'Rogue', 'year': None, 'color': ['Black', 'Blue'], 'fuel\_type': 'petrol', 'price': 910000}

{'id': 16, 'brand': 'Toyota', 'model': None, 'year': 2021, 'color': ['Red', 'Grey'], 'fuel\_type': 'petrol', 'price': 7500000}

{'id': 17, 'brand': 'Honda', 'model': 'Pilot', 'year': 2022, 'color': ['Silver', 'Black'], 'fuel\_type': 'diesel', 'price': None}

{'id': 18, 'brand': 'Ford', 'model': 'Escape', 'year': None, 'color': ['Blue', 'White'], 'fuel\_type': 'diesel', 'price': 980000}

{'id': 19, 'brand': 'Chevrolet', 'model': 'Malibu', 'year': 2020, 'color': ['White', 'Black'], 'fuel\_type': 'petrol', 'price': 780000}

{'id': 20, 'brand': 'Nissan', 'model': 'Maxima', 'year': 2018, 'color': ['Red', 'White'], 'fuel\_type': 'diesel', 'price': None}

**Code:**

#insertion

new\_data = {

    "id":21,

    "brand":"Hyundai",

    "model":"I20",

    "year":2023,

    "color":"White",

    "price":1500000

}

data.append(new\_data)

with open('cars\_data.json', 'w') as json\_file:

  print(json.dumps(data, indent=4))

**Output:**

[

{

"id": 1,

"brand": "Toyota",

"model": "Corolla",

"year": 2018,

"color": [

"Red",

"Silver",

"Black"

],

"fuel\_type": "petrol",

"price": 150000

},

{

"id": 2,

"brand": "Honda",

"model": "Civic",

"year": 2019,

"color": [

"Black",

"Red"

],

"fuel\_type": "diesel",

"price": 180000

},

{

"id": 3,

"brand": "Ford",

"model": null,

"year": 2023,

"color": [

"Yellow",

"Blue"

],

"fuel\_type": "electric",

"price": null

},

{

"id": 4,

"brand": "Chevrolet",

"model": "Camaro",

"year": null,

"color": "Blue",

"fuel\_type": "petrol",

"price": 1800000

},

{

"id": 5,

"brand": "Nissan",

"model": "Altima",

"year": 2018,

"color": [

"White",

"Black"

],

"fuel\_type": "diesel",

"price": 670000

},

{

"id": 6,

"brand": "Toyota",

"model": "RAV4",

"year": 2020,

"color": [

"Silver",

"White",

"Red"

],

"fuel\_type": "electric",

"price": 930000

},

{

"id": 7,

"brand": "Honda",

"model": "Accord",

"year": 2022,

"color": [

"Grey",

"White"

],

"fuel\_type": "electric",

"price": 800000

},

{

"id": 8,

"brand": "Ford",

"model": "F-150",

"year": 2018,

"color": [

"Blue",

"Black"

],

"fuel\_type": "diesel",

"price": 789000

},

{

"id": 9,

"brand": "Chevrolet",

"model": "Silverado",

"year": 2019,

"color": [

"Red",

"Grey"

],

"fuel\_type": "diesel",

"price": 5800000

},

{

"id": 10,

"brand": "Nissan",

"model": "Sentra",

"year": 2017,

"color": [

"Black",

"White"

],

"fuel\_type": "petrol",

"price": 640000

},

{

"id": 11,

"brand": "Toyota",

"model": "Camry",

"year": 2019,

"color": [

"White",

"Grey",

"Black"

],

"fuel\_type": "petrol",

"price": null

},

{

"id": 12,

"brand": "Honda",

"model": "CR-V",

"year": 2020,

"color": [

"Silver",

"White",

"Red"

],

"fuel\_type": "electric",

"price": 890000

},

{

"id": 13,

"brand": "Ford",

"model": null,

"year": 2018,

"color": [

"Grey",

"White"

],

"fuel\_type": "diesel",

"price": 2400000

},

{

"id": 14,

"brand": "Chevrolet",

"model": "Impala",

"year": 2023,

"color": [

"Blue",

"Black"

],

"fuel\_type": "electric",

"price": 520000

},

{

"id": 15,

"brand": "Nissan",

"model": "Rogue",

"year": null,

"color": [

"Black",

"Blue"

],

"fuel\_type": "petrol",

"price": 910000

},

{

"id": 16,

"brand": "Toyota",

"model": null,

"year": 2021,

"color": [

"Red",

"Grey"

],

"fuel\_type": "petrol",

"price": 7500000

},

{

"id": 17,

"brand": "Honda",

"model": "Pilot",

"year": 2022,

"color": [

"Silver",

"Black"

],

"fuel\_type": "diesel",

"price": null

},

{

"id": 18,

"brand": "Ford",

"model": "Escape",

"year": null,

"color": [

"Blue",

"White"

],

"fuel\_type": "diesel",

"price": 980000

},

{

"id": 19,

"brand": "Chevrolet",

"model": "Malibu",

"year": 2020,

"color": [

"White",

"Black"

],

"fuel\_type": "petrol",

"price": 780000

},

{

"id": 20,

"brand": "Nissan",

"model": "Maxima",

"year": 2018,

"color": [

"Red",

"White"

],

"fuel\_type": "diesel",

"price": null

},

{

"id": 21,

"brand": "Hyundai",

"model": "I20",

"year": 2023,

"color": "White",

"price": 1500000

}

]

**Code:**

#deletion

for car in data:

    if car["id"] == 13:

        data.remove(car)

        break

for car in data:

    print(car)

**Output:**

{'id': 1, 'brand': 'Toyota', 'model': 'Corolla', 'year': 2018, 'color': ['Red', 'Silver', 'Black'], 'fuel\_type': 'petrol', 'price': 150000}

{'id': 2, 'brand': 'Honda', 'model': 'Civic', 'year': 2019, 'color': ['Black', 'Red'], 'fuel\_type': 'diesel', 'price': 180000}

{'id': 3, 'brand': 'Ford', 'model': None, 'year': 2023, 'color': ['Yellow', 'Blue'], 'fuel\_type': 'electric', 'price': None}

{'id': 4, 'brand': 'Chevrolet', 'model': 'Camaro', 'year': None, 'color': 'Blue', 'fuel\_type': 'petrol', 'price': 1800000}

{'id': 5, 'brand': 'Nissan', 'model': 'Altima', 'year': 2018, 'color': ['White', 'Black'], 'fuel\_type': 'diesel', 'price': 670000}

{'id': 6, 'brand': 'Toyota', 'model': 'RAV4', 'year': 2020, 'color': ['Silver', 'White', 'Red'], 'fuel\_type': 'electric', 'price': 930000}

{'id': 7, 'brand': 'Honda', 'model': 'Accord', 'year': 2022, 'color': ['Grey', 'White'], 'fuel\_type': 'electric', 'price': 800000}

{'id': 8, 'brand': 'Ford', 'model': 'F-150', 'year': 2018, 'color': ['Blue', 'Black'], 'fuel\_type': 'diesel', 'price': 789000}

{'id': 9, 'brand': 'Chevrolet', 'model': 'Silverado', 'year': 2019, 'color': ['Red', 'Grey'], 'fuel\_type': 'diesel', 'price': 5800000}

{'id': 10, 'brand': 'Nissan', 'model': 'Sentra', 'year': 2017, 'color': ['Black', 'White'], 'fuel\_type': 'petrol', 'price': 640000}

{'id': 11, 'brand': 'Toyota', 'model': 'Camry', 'year': 2019, 'color': ['White', 'Grey', 'Black'], 'fuel\_type': 'petrol', 'price': None}

{'id': 12, 'brand': 'Honda', 'model': 'CR-V', 'year': 2020, 'color': ['Silver', 'White', 'Red'], 'fuel\_type': 'electric', 'price': 890000}

{'id': 14, 'brand': 'Chevrolet', 'model': 'Impala', 'year': 2023, 'color': ['Blue', 'Black'], 'fuel\_type': 'electric', 'price': 520000}

{'id': 15, 'brand': 'Nissan', 'model': 'Rogue', 'year': None, 'color': ['Black', 'Blue'], 'fuel\_type': 'petrol', 'price': 910000}

{'id': 16, 'brand': 'Toyota', 'model': None, 'year': 2021, 'color': ['Red', 'Grey'], 'fuel\_type': 'petrol', 'price': 7500000}

{'id': 17, 'brand': 'Honda', 'model': 'Pilot', 'year': 2022, 'color': ['Silver', 'Black'], 'fuel\_type': 'diesel', 'price': None}

{'id': 18, 'brand': 'Ford', 'model': 'Escape', 'year': None, 'color': ['Blue', 'White'], 'fuel\_type': 'diesel', 'price': 980000}

{'id': 19, 'brand': 'Chevrolet', 'model': 'Malibu', 'year': 2020, 'color': ['White', 'Black'], 'fuel\_type': 'petrol', 'price': 780000}

{'id': 20, 'brand': 'Nissan', 'model': 'Maxima', 'year': 2018, 'color': ['Red', 'White'], 'fuel\_type': 'diesel', 'price': None}

{'id': 21, 'brand': 'Hyundai', 'model': 'I20', 'year': 2023, 'color': 'White', 'price': 1500000}

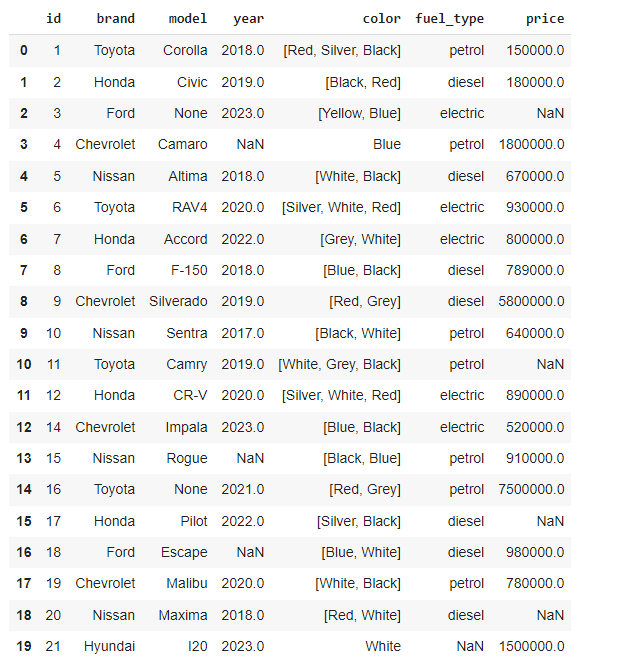
**Code:**

import pandas as pd

df=pd.DataFrame(data)

display(df)

**Output:**

****

**Code:**

count=df.isnull().sum()

print(count)

**Output:**

id 0

brand 0

model 2

year 3

color 0

fuel\_type 1

price 4

dtype: int64

**Code:**

import matplotlib.pyplot as plt

plt.scatter(count.index,count.values)

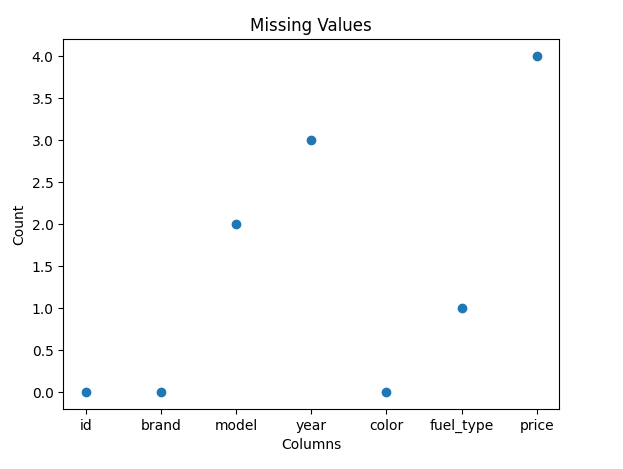
plt.xlabel("Columns")

plt.ylabel("Count")

plt.title("Missing Values")

plt.show()

**Output:**



**Code:**

#filling missing values

df['brand'].fillna(df['brand'].mode(), inplace=True)

df['model'].fillna(df['model'].mode(), inplace=True)

df['fuel\_type'].fillna(df['fuel\_type'].mode()[0], inplace=True)

df['year'].fillna(df['year'].median(), inplace=True)

df['price'].fillna(df['price'].mean(), inplace=True)

display(df)

**Output:**

****

**Code:**

#data transformation

from datetime import datetime

df['year'] = pd.to\_datetime(df['year'], format='%Y').dt.year

current\_year = datetime.now().year

df['Car Age'] = current\_year - df['year']

display(df)

**Output:**

****

**Code:**

df.isnull().sum()

**Output:**

id 0

brand 0

model 0

year 0

color 0

fuel\_type 0

price 0

No of year 0

dtype: int64

**Code:**

import pandas as pd

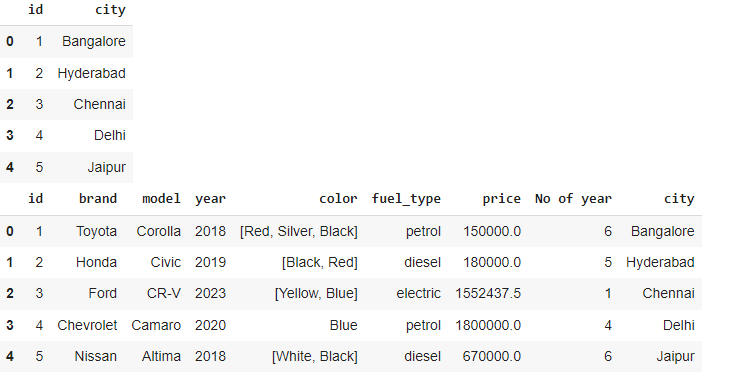
df1 = pd.DataFrame({"id": [1, 2, 3, 4, 5], "city": ["Bangalore", "Hyderabad", "Chennai", "Delhi", "Jaipur"]})

display(df1)

merged\_df = pd.merge(df, df1, on="id", how="inner")

display(merged\_df)

**Output:**

****

**Code:**

#iterating the brand names

for i in data:

  val=i["brand"]

  print(val)

**Output:**

Toyota

Honda

Ford

Chevrolet

Nissan

Toyota

Honda

Ford

Chevrolet

Nissan

Toyota

Honda

Chevrolet

Nissan

Toyota

Honda

Ford

Chevrolet

Nissan

Hyundai

**Code:**

#iterating over unique brand names

unique\_brands =set()

for i in data:

    val = i["brand"]

    unique\_brands.add(val)

for brand in unique\_brands:

    print(brand)

**Output:**

Nissan

Chevrolet

Toyota

Hyundai

Honda

Ford

**Inference:**

**LAB 2**

**To demonstrate effective Xml parsing using python, allowing users to extract structured data from a xml file and utilize it for further processing or analysis (includes: parsing of xml file, accessing elements, updating data, adding new elements, converting it to dataframe and perform basic analysis)**

**Code:**

import xml.etree.ElementTree as ET

# Parse the XML file

tree = ET.parse("C:\\Users\\bache\\OneDrive\\Desktop\\Data Science\\computer.xml")

root = tree.getroot()

# Print the root element

print(f"Root element: {root.tag}")

**Output:**

Root element: sentences

**Code:**

#Printing the attribute of the root

print(root.attrib)

**Output:**

**{}**

**Code:**

# Iterate over each sentence

for sentence in root.findall('sentence'):

sentence\_id = sentence.attrib['id']

text = sentence.find('text').text

print(f"Sentence ID: {sentence\_id}\nText: {text}\n")

**Output:**

Sentence ID: 2339

Text: I charge it at night and skip taking the cord with me because of the good battery life.

Sentence ID: 812

Text: I bought a HP Pavilion DV4-1222nr laptop and have had so many problems with the computer.

Sentence ID: 1316

Text: The tech guy then said the service center does not do 1-to-1 exchange and I have to direct my concern to the "sales" team, which is the retail shop which I bought my netbook from.

Sentence ID: 2328

Text: I investigated netbooks and saw the Toshiba NB305-N410BL.

Sentence ID: 2193

Text: The other day I had a presentation to do for a seminar at a large conference in town- lots of people, little time to prep and have to set up a computer to a projector, etc.

Sentence ID: 2005

Text: It is of high quality, has a killer GUI, is extremely stable, is highly expandable, is bundled with lots of very good applications, is easy to use, and is absolutely gorgeous.

Sentence ID: 2789

Text: Easy to start up and does not overheat as much as other laptops.

Sentence ID: 76

Text: I even got my teenage son one, because of the features that it offers, like, iChat, Photobooth, garage band and more!

Sentence ID: 80

Text: Great laptop that offers many great features!

Sentence ID: 917

Text: One night I turned the freaking thing off after using it, the next day I turn it on, no GUI, screen all dark, power light steady, hard drive light steady and not flashing as it usually does.

Sentence ID: 1630

Text: I took it back for an Asus and same thing- blue screen which required me to remove the battery to reset.

Sentence ID: 337

Text: However, the multi-touch gestures and large tracking area make having an external mouse unnecessary (unless you're gaming).

Sentence ID: 2568

Text: I love the way the entire suite of software works together.

Sentence ID: 2759

Text: This laptop meets every expectation and Windows 7 is great!

Sentence ID: 2980

Text: I can barely use any usb devices because they will not stay connected properly.

Sentence ID: 1054

Text: -No backlit keyboard, but not an issue for me.

Sentence ID: 2587

Text: One suggestion I do have, is to not bother getting Microsoft office for the mac expecting it will work just like you knew it to on a PC.

Sentence ID: 2202

Text: Pairing it with an iPhone is a pure pleasure - talk about painless syncing - used to take me forever - now it's a snap.

Sentence ID: 92

Text: The machine is slow to boot up and occasionally crashes completely.

Sentence ID: 325

Text: I did have to replace the battery once, but that was only a couple months ago and it's been working perfect ever since.

Sentence ID: 915

Text: A month or so ago, the freaking motherboard just died.

Sentence ID: 997

Text: Drivers updated ok but the BIOS update froze the system up and the computer shut down.

Sentence ID: 463

Text: Spent 2 hours on phone with HP Technical Support.

Sentence ID: 147

Text: The keyboard is too slick.

Sentence ID: 2644

Text: It's so much easier to navigate through the operating system, to find files, and it runs a lot faster!

Sentence ID: 3124

Text: The machine came loaded with bloatware, and a day after I got it I did a fresh install of Windows 7 Home.

Sentence ID: 29

Text: Absolutely the best computer I have ever owned.

Sentence ID: 1671

Text: It's easy to use, very fast, light weight, and has all the features I need.

**Code:**

#Iterating the elements of xml

tag\_list = [elem.tag for elem in root.iter()]

# Print the list of tag names

print(tag\_list)

**Output:**

['sentences', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'sentence', 'text', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'sentence', 'text', 'aspectTerms', 'aspectTerm', 'aspectTerm']

**Code:**

#updation

tree = ET.parse("C:\\Users\\bache\\OneDrive\\Desktop\\Data Science\\computer.xml")

root = tree.getroot()

sentence\_id = "2339"

new\_aspectTerm = ET.Element('aspectTerm', {'term': 'new\_aspect', 'polarity': 'positive', 'from': '100', 'to': '110'})

for sentence in root.findall('.//sentence[@id="' + sentence\_id + '"]'):

aspectTerms = sentence.find('aspectTerms')

if aspectTerms is None:

aspectTerms = ET.SubElement(sentence, 'aspectTerms')

aspectTerms.append(new\_aspectTerm)

updated\_xml\_file = 'updated\_xml\_file.xml'

tree.write(updated\_xml\_file, encoding='utf-8', xml\_declaration=True)

print(f"Updated XML saved to {updated\_xml\_file}")

**Output:**

Updated XML saved to updated\_xml\_file.xml

**Code:**

# Example of adding a new aspect term to a specific sentence

sentence\_to\_modify = root.find("./sentence[@id='2339']")

new\_aspect\_term = ET.SubElement(sentence\_to\_modify.find('aspectTerms'), 'aspectTerm', {'term': 'screen', 'polarity': 'negative', 'from': '10', 'to': '15'})

tree.write('modified\_xml\_file.xml')

import xml.etree.ElementTree as ET

import pandas as pd

tree = ET.parse("C:\\Users\\bache\\OneDrive\\Desktop\\Data Science\\computer.xml")

root = tree.getroot()

data = {

'sentence\_id': [],

'text': [],

'aspect\_term': [],

'polarity': [],

'from': [],

'to': []

}

for sentence in root.findall('sentence'):

sentence\_id = sentence.attrib['id']

text = sentence.find('text').text

aspect\_terms = sentence.find('aspectTerms')

if aspect\_terms is not None:

for aspect\_term in aspect\_terms.findall('aspectTerm'):

term = aspect\_term.attrib['term']

polarity = aspect\_term.attrib['polarity']

start = int(aspect\_term.attrib['from']) if 'from' in aspect\_term.attrib else None

end = int(aspect\_term.attrib['to']) if 'to' in aspect\_term.attrib else None

data['sentence\_id'].append(sentence\_id)

data['text'].append(text)

data['aspect\_term'].append(term)

data['polarity'].append(polarity)

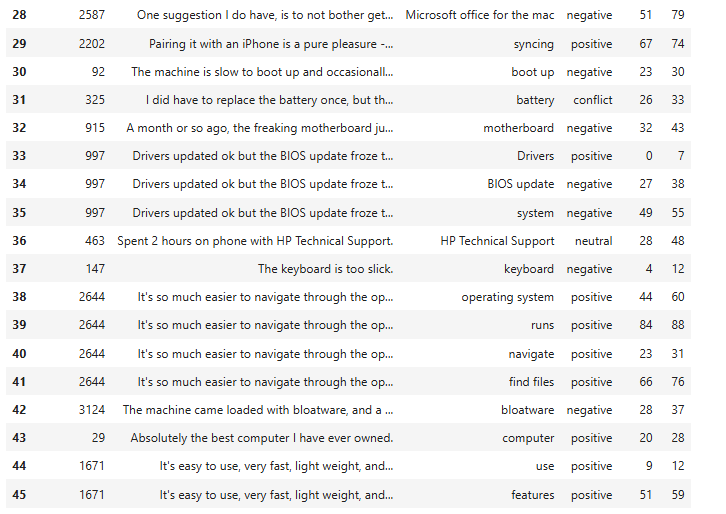
data['from'].append(start)

data['to'].append(end)

df = pd.DataFrame(data)

display(df)

**Output:**



**Code:**

df.info()

**Output:**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 46 entries, 0 to 45

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 sentence\_id 46 non-null object

1 text 46 non-null object

2 aspect\_term 46 non-null object

3 polarity 46 non-null object

4 from 46 non-null int64

5 to 46 non-null int64

dtypes: int64(2), object(4)

memory usage: 2.3+ KB

**Code:**

#checking missing values

df.isnull().sum()

**Output:**

sentence\_id 0

text 0

aspect\_term 0

polarity 0

from 0

to 0

dtype: int64

**Code:**

#filtering data

positive\_df = df[df['polarity'] == 'positive']

battery\_df = df[df['aspect\_term'] == 'battery']

display(positive\_df)

display(battery\_df)

**Output:**

****

**Code:**

# Group by polarity and count occurrences

polarity\_counts = df.groupby('polarity').size()

print(polarity\_counts)

**Output:**

polarity

conflict 2

negative 13

neutral 7

positive 24

dtype: int64

**Code:**

# Group by polarity and count occurrences

import seaborn as sns

import matplotlib.pyplot as plt

polarity\_counts = df.groupby('polarity').size().reset\_index(name='count')

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

sns.barplot(x='polarity', y='count', data=polarity\_counts, palette='pastel')

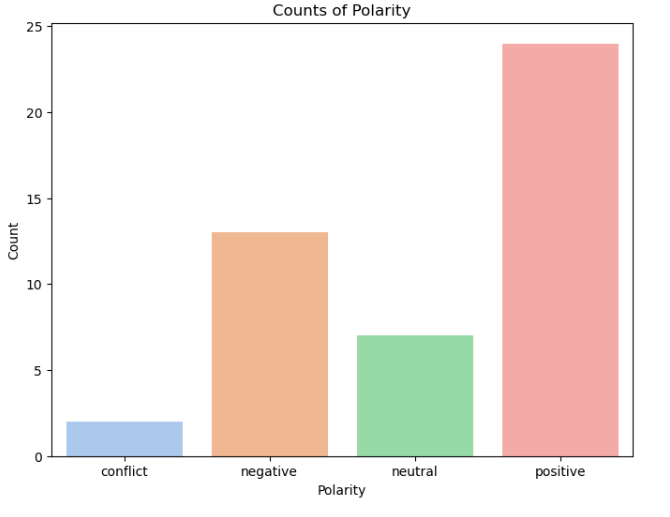
plt.title('Counts of Polarity')

plt.xlabel('Polarity')

plt.ylabel('Count')

plt.show()

**Output:**



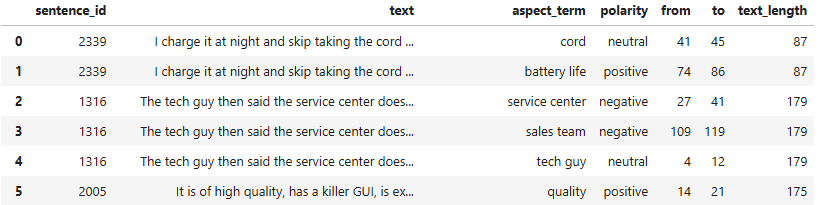
**Code:**

#data transformation

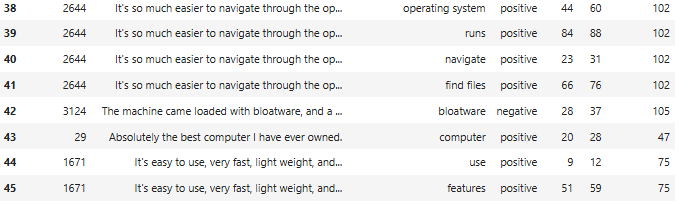
df['text\_length'] = df['text'].apply(len)

display(df)

**Output:**

****

****

****

**Code:**

#adding company name

company\_names = [

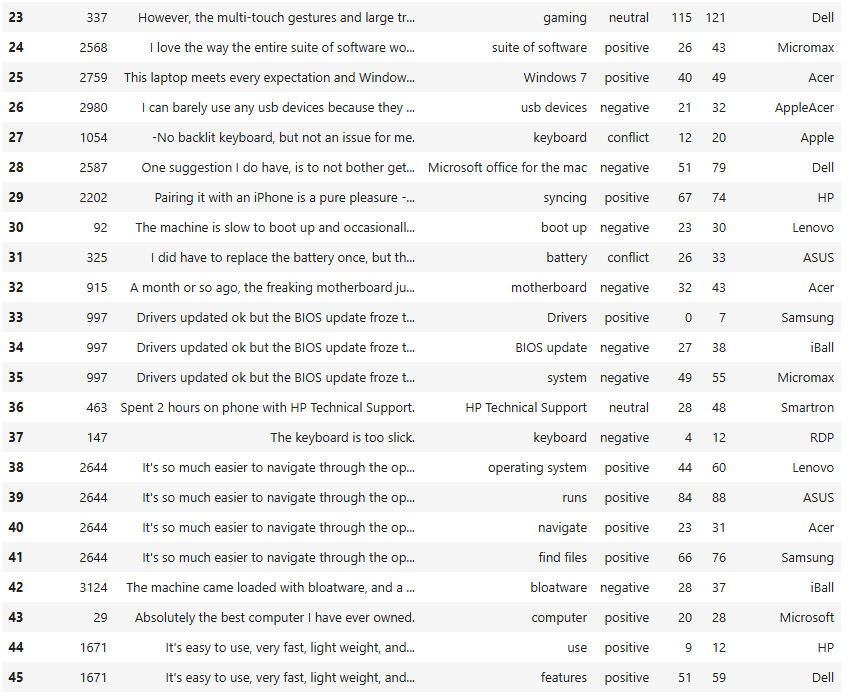
"Lenovo", "Dell","Acer","Apple","Dell","HP","Lenovo","ASUS","Acer","Samsung","iBall","Micromax","Smartron","RDP", "Dell",

"Microsoft","Razer","HP","Acer","ASUS","iBall", "HP", "HP", "Dell","Micromax","Acer","Apple""Acer","Apple","Dell","HP","Lenovo","ASUS","Acer","Samsung","iBall","Micromax","Smartron","RDP","Lenovo","ASUS","Acer","Samsung","iBall","Microsoft","HP","Dell"]

df["company\_name"] = company\_names

display(df)

**Output:**



**Inference:**

**LAB 3**

**To demonstrate effective excel parsing using python convert excel file into pandas dataframe and perform analysis and visualization**

**Code:**

#installing necessary packages

pip install xlrd

pip install openpyxl

import pandas as pd

file\_path ="standard data.xls"

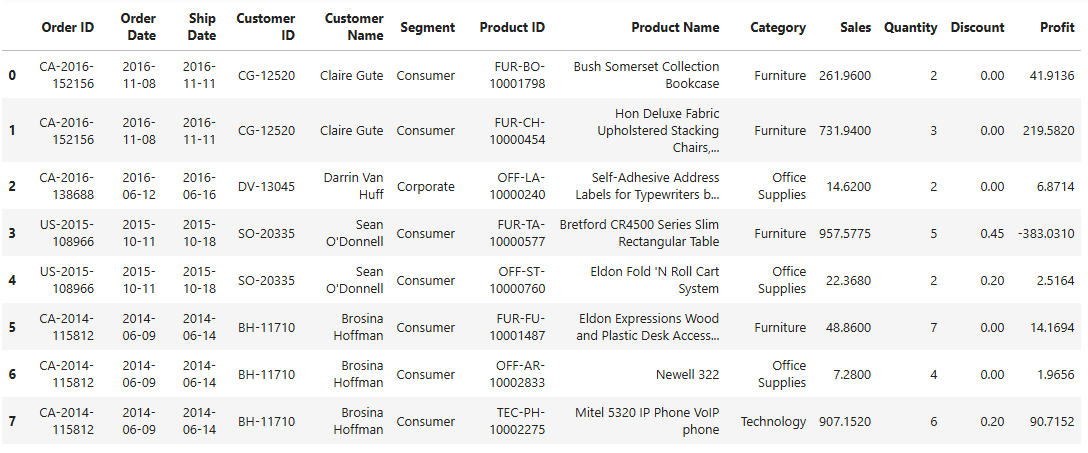
data = pd.read\_excel(file\_path, engine='xlrd')

display(data.head(3))

df=data[["Order ID","Order Date","Ship Date","CustomerID","CustomerName","Segment","Product ID","Product Name","Category","Sales","Quantity","Discount","Profit"]]

df.head(8)

**Output:**

****

**Code:**

xls = pd.ExcelFile(file\_path, engine='xlrd')

print("Sheet names:", xls.sheet\_names)

sheets = {}

for sheet\_name in xls.sheet\_names:

sheets[sheet\_name] = pd.read\_excel(file\_path, sheet\_name=sheet\_name, engine='xlrd')

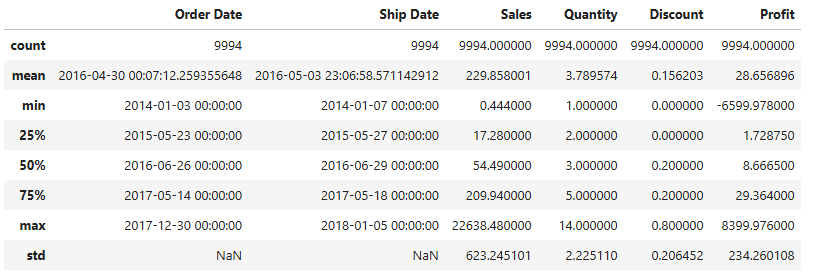
**Output:**

Sheet names: ['Orders', 'Returns', 'People']

**Code:**

#describe

df.describe()  
**Output:**

****

**Code:**

df.info()

**Output:**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 9994 entries, 0 to 9993

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Order ID 9994 non-null object

1 Order Date 9994 non-null datetime64[ns]

2 Ship Date 9994 non-null datetime64[ns]

3 Customer ID 9994 non-null object

4 Customer Name 9994 non-null object

5 Segment 9994 non-null object

6 Product ID 9994 non-null object

7 Product Name 9994 non-null object

8 Category 9994 non-null object

9 Sales 9994 non-null float64

10 Quantity 9994 non-null int64

11 Discount 9994 non-null float64

12 Profit 9994 non-null float64

dtypes: datetime64[ns](2), float64(3), int64(1), object(7)

memory usage: 1015.1+ KB

**Code:**

df.isnull().sum()

**Output:**

Order ID 0

Order Date 0

Ship Date 0

Customer ID 0

Customer Name 0

Segment 0

Product ID 0

Product Name 0

Category 0

Sales 0

Quantity 0

Discount 0

Profit 0

dtype: int64

**Code:**

df.dtypes

**Output:**

Order ID object

Order Date datetime64[ns]

Ship Date datetime64[ns]

Customer ID object

Customer Name object

Segment object

Product ID object

Product Name object

Category object

Sales float64

Quantity int64

Discount float64

Profit float64

dtype: object

**Code:**

# Check the range of dates

print(f"Minimum Order Date: {df['Order Date'].min()}")

print(f"Maximum Order Date: {df['Order Date'].max()}")

**Output:**

Minimum Order Date: 2014-01-03 00:00:00

Maximum Order Date: 2017-12-30 00:00:00

**Code:**

print(f"Minimum Ship Date: {df['Ship Date'].min()}")

print(f"Maximum Ship Date: {df['Ship Date'].max()}")

**Output:**

Minimum Ship Date: 2014-01-07 00:00:00

Maximum Ship Date: 2018-01-05 00:00:00

**Code**

#yearly sales

import matplotlib.pyplot as plt

df['Order Date'] = pd.to\_datetime(df['Order Date'])

df['Order Year'] = df['Order Date'].dt.to\_period('Y')

yearly\_sales = df.groupby('Order Year')['Sales'].sum().reset\_index()

plt.figure(figsize=(6, 4))

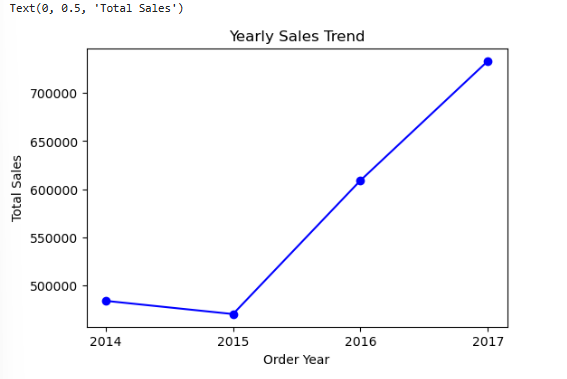
plt.plot(yearly\_sales['Order Year'].astype(str), yearly\_sales['Sales'], marker='o', color='b')

plt.title('Yearly Sales Trend')

plt.xlabel('Order Year')

plt.ylabel('Total Sales')

**Output:**



**Code:**

# Top customers by sales

import seaborn as sns

top\_customers = df.groupby('Customer Name').agg({'Sales': 'sum'}).sort\_values(by='Sales', ascending=False).head(5)

plt.figure(figsize=(6, 4))

sns.barplot(x='Sales', y=top\_customers.index, data=top\_customers, palette='rocket')

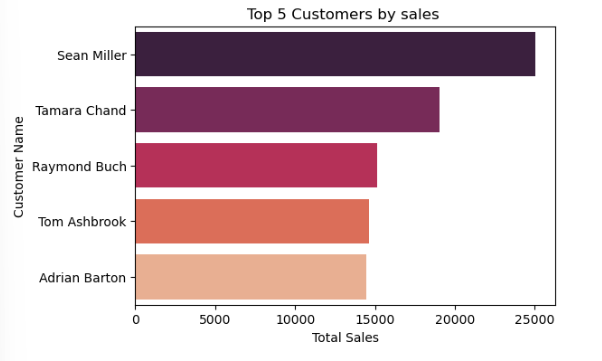
plt.title('Top 5 Customers by sales ')

plt.xlabel('Total Sales')

plt.ylabel('Customer Name')

plt.show()

**Output:**

****

**Code:**

# Profitability by product category

df['Category'] = df['Product ID'].str.split('-').str[0]

category\_profit = df.groupby('Category')['Profit'].sum().sort\_values(ascending=False)

plt.figure(figsize=(6,4))

sns.barplot(x=category\_profit.index, y=category\_profit,palette='coolwarm')

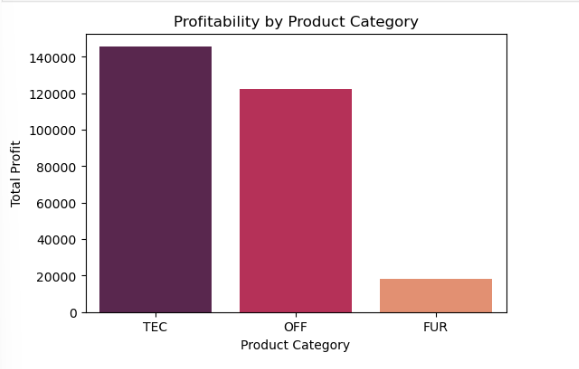
plt.title('Profitability by Product Category')

plt.xlabel('Product Category')

plt.ylabel('Total Profit')

plt.show()

**Output:**

****

**Code:**

#product sold the most

sales = df.groupby(['Product ID', 'Product Name', 'Category'])['Quantity'].sum().reset\_index()

max\_sold = sales.loc[sales['Quantity'].idxmax()]

print("Product sold the most:")

print(max\_sold[['Product ID', 'Product Name', 'Category', 'Quantity']])

**Output:**

Product sold the most:

Product ID OFF-BI-10001524

Product Name GBC Premium Transparent Covers with Diagonal L...

Category Office Supplies

Quantity 67

Name: 704, dtype: object

**Code:**

#product sold the least

min\_sold = sales.loc[sales['Quantity'].idxmin()]

print("\nProduct sold the least:")

print(min\_sold[['Product ID', 'Product Name', 'Category', 'Quantity']])

**Output:**

Product sold the least:

Product ID FUR-BO-10002206

Product Name Bush Saratoga Collection 5-Shelf Bookcase, Han...

Category Furniture

Quantity 1

Name: 17, dtype: object

**Code:**

#Total sales by category

category\_sales = df.groupby('Category').agg({'Sales': 'sum', 'Profit': 'sum'}).reset\_index()

plt.figure(figsize=(6,4))

sns.barplot(x='Category', y='Sales', data=category\_sales, palette='viridis')

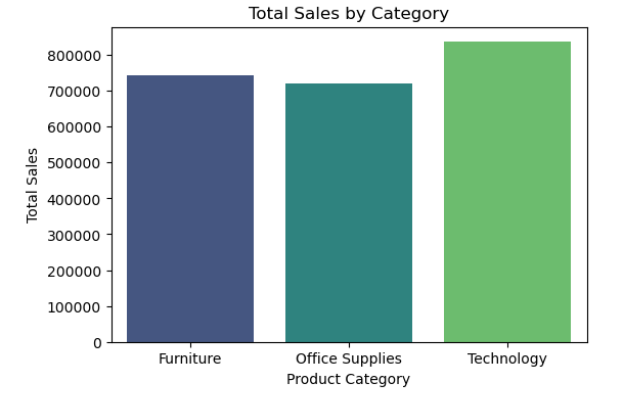
plt.title('Total Sales by Category')

plt.xlabel('Product Category')

plt.ylabel('Total Sales')

plt.show()

**Output:**

****

**Code:**

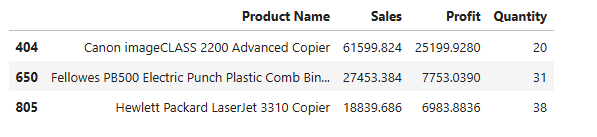
# top 3 products with max profit

top\_products = df.groupby('Product Name').agg({'Sales': 'sum', 'Profit': 'sum', 'Quantity': 'sum'}).reset\_index()

top\_products\_profit = top\_products.sort\_values(by='Profit', ascending=False).head(3)

display(top\_products\_profit)

**Output:**



**Code:**

# Selecting only the top 3 cities by sales

city\_sales = data.groupby('City').agg({'Sales': 'sum'}).reset\_index().sort\_values(by='Sales', ascending=False)

top\_cities = city\_sales.head(3)

plt.figure(figsize=(5,3))

sns.barplot(x='City', y='Sales', data=top\_cities, palette='coolwarm')

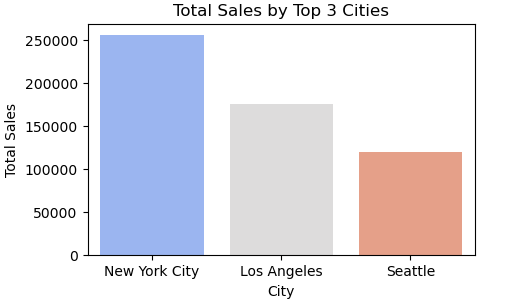
plt.title('Total Sales by Top 3 Cities')

plt.xlabel('City')

plt.ylabel('Total Sales')

plt.show()

**Output:**

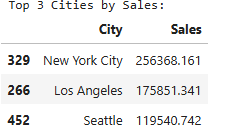


**Code:**

print("Top 3 Cities by Sales:")

display(top\_cities[['City', 'Sales']])

**Output:**



**Code:**

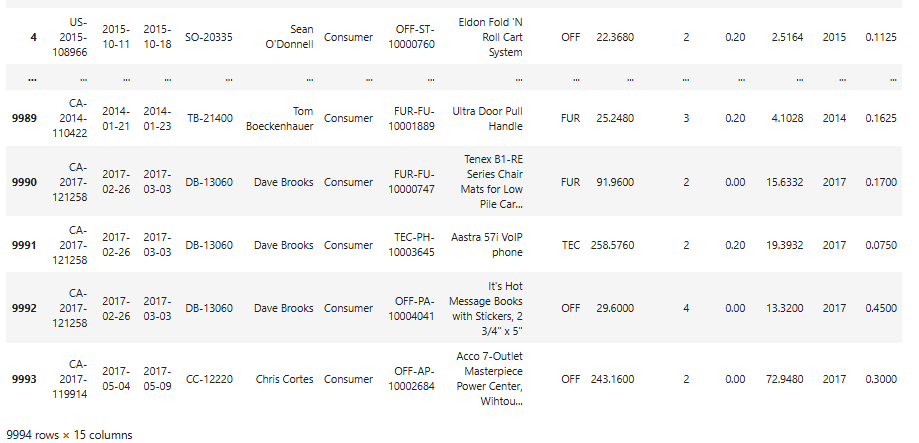
# Calculate profit margin

df['Profit Margin'] = df['Profit'] / df['Sales']

# Display the DataFrame with new features

display(df)

**Output:** ****

****

**Code:**

# Profit margin distribution

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

sns.histplot(df['Profit Margin'], bins=20, kde=True)

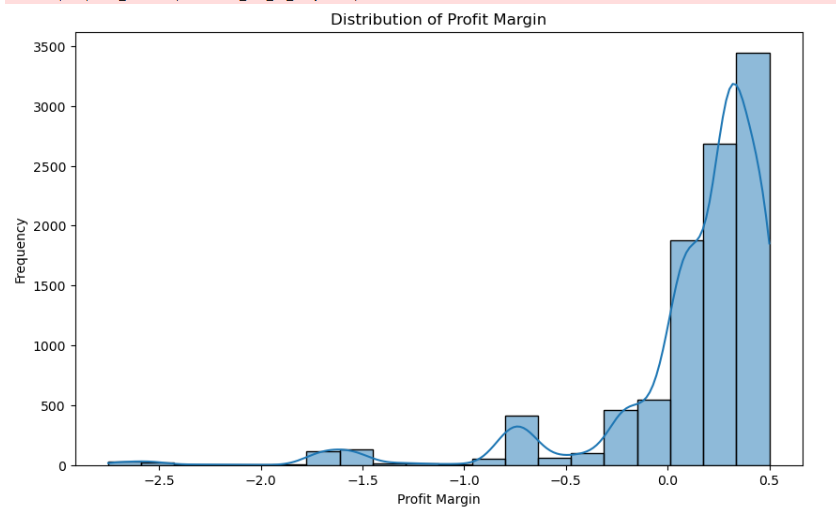
plt.title('Distribution of Profit Margin')

plt.xlabel('Profit Margin')

plt.ylabel('Frequency')

plt.show()

**Output:**



**Inference:**

**LAB 4**

**To demonstrate effective pdf parsing using python**

**1) parsing the pdf, file extracting text ,links, images**

**2)merging 2 pdf using pdf merger and saving it**

**3)extracting table from pdf**

**4) rotating the particular page**

#installing necessary libraries

pip install PyPDF2

pip install pycryptodome

pip install pypdf

pip install PyMuPDF

pip install pdfplumber

pip install tabula-py

**Code:**

from pdfminer.high\_level import extract\_text

with open('TOOTH-FAIRY.pdf','rb')as file:

text=extract\_text(file)

print("text extracted:")

print(text)

**Output:**

text extracted:

By Zani Mathoo

Illustrated by maaillustrations.com

By Zani Mathoo

A wonderful story that weaves its magic in the

child’s imagination, even while imparting an

invaluable lesson in oral health and hygiene.

This is a story that the children will return to

time and again for a fun bedtime read.

Published by Monkey Pen Ltd

Dear Supporter,

Thank you for downloading our children's books.

Monkey Pen's Vision is to provide thousands of free children's

books to young readers around the globe.

Please share our books with your friends and family to

support our mission. Thank you

Please make a donation on Patreon to support

Monkey Pens Free Book Project:

Deep in the New Forest where no big people ever went there was a tiny little

magical village called the Ordon Gardens. Here, three little tooth fairies

named Incisor, Canine and Molar were busy at work in their factory. The

three tooth fairies were all named after types of teeth.

Incisor was tall and skinny with short pink hair, Canine was also tall with blue

hair and Molar was the short, wise one with curly, silver locks.

All day long the busy little fairies cleaned and shined the children’s milk

One day Incisor realised that the teeth coming into the cleaning factory

were not as strong as they used be.

“Oh dear!” said Incisor “These teeth are fragile and brittle, with holes and

black spots. They crumble as soon as I try to polish them with my cloth.

Whatever are we to do?”

“Yes.”said Canine “I know exactly what’s wrong, The children are

beginning to eat far too many sweets and are not looking after their

teeth at all.” “I know…we can fix it!” Said Molar the wise one. She

sounded quite excited by her grand idea.

The three fairies looked at each other and waited while Molar

explained all. As usual, it was a fabulous idea. The three of them

wasted no time at all and got to work straight away.

By the time Canine arrived, Junior was fast asleep. She flew in through the

top window that was ajar and glided quickly down to Junior’s pillow. She

perched beside him and braced herself in preparation for the task…

“Heave……..” she lifted the pillow and wedged herself between the pillow and

the bed. She quickly grabbed the tooth and tossed it over to her backpack.

She placed two coins and the fairy’s special card in exchange under the

pillow and flew off into the night sky to find more milk teeth.

In the morning, as soon as Junior woke up, he rummaged under the pillow to

see if the tooth fairy had left any money for him.

He was thrilled to find the coins but looked blankly at the card he found in

his hand. Mummy didn’t mention anything about a card…he wondered.

He read the card out aloud…

The Tooth Fairy’s guide to healthy teeth:

1.Eat Healthy.

2.Cut down on sugary sweets & drinks.

3.Brush your teeth twice a day.

4.Use a fluoride toothpaste.

5.Visit the Dentist regularly.

Back in the village of Ordon Gardens, the fairies were very happy about the

new information card they were giving to the children. Soon enough the milk

teeth coming into the factory were strong and healthy again.

Advertisement

Your Story Book!

A book specially made,

with you as the main hero

or heroine!

Personalised children’s gifts by [www.monkeypen.com](https://www.monkeypen.com/)

By Zani Mathoo

A wonderful story that weaves its magic in the

child’s imagination, even while imparting an

invaluable lesson in oral health and hygiene.

This is a story that the children will return to time

and again for a fun bedtime read.

Please share our books with your friends and

family to support our mission. Thank you

Published by MonkeyPen.com

Illustrated by [www.maaillustrations.com](https://www.maaillustrations.com/)

**Code:**

from pypdf import PdfReader

reader = PdfReader("Snowy Owl.pdf")

print(f"There are {len(reader.pages)} Pages")

**Output:**

There are 84 Pages

**Code:**

**#**extracting images from a particular page

import fitz # PyMuPDF

from PIL import Image

import io

from IPython.display import display

def images(pdf\_file, page\_number):

doc = fitz.open(pdf\_file)

if page\_number < 1 or page\_number > len(doc):

print(f"Page number {page\_number} is out of range.")

return

page = doc.load\_page(page\_number - 1) # page\_number is 1-based, so adjust to 0-based index

# Extract and display images from the page

for img\_index, img in enumerate(page.get\_images(full=True)):

xref = img[0]

base\_image = doc.extract\_image(xref)

image\_bytes = base\_image['image']

image = Image.open(io.BytesIO(image\_bytes))

display(image)

doc.close()

pdf\_file = 'TOOTH-FAIRY.pdf'

user\_input\_pgnum= int(input("Enter the page number to extract and display images: "))

images(pdf\_file, user\_input\_pgnum)

**Output:**

Enter the page number to extract and display images: 4



**Code:**

import fitz

def extract\_links\_from\_pdf(pdf\_file):

doc = fitz.open(pdf\_file)

links = []

for page\_num in range(len(doc)):

page = doc.load\_page(page\_num)

page\_links = page.get\_links()

for link in page\_links:

if 'uri' in link:

link\_info = {

'page\_number': page\_num + 1,

'uri': link['uri']

}

if 'rect' in link:

link\_info['rectangle'] = link['rect']

else:

link\_info['rectangle'] = None

links.append(link\_info)

doc.close()

return links

pdf\_file = 'TOOTH-FAIRY.pdf'

extracted\_links = extract\_links\_from\_pdf(pdf\_file)

for link in extracted\_links:

print(f"Page {link['page\_number']}: {link['uri']} at {link['rectangle']}")

**Output:**

**Page 1:** [**https://monkeypen.com/?ref=pdfbooks**](https://monkeypen.com/?ref=pdfbooks) **at None**

**Page 2:** [**https://monkeypen.com/?ref=pdfbooks**](https://monkeypen.com/?ref=pdfbooks) **at None**

**Page 3:** [**http://www.facebook.com/sharer/sharer.php?u=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare**](http://www.facebook.com/sharer/sharer.php?u=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare) **at None**

**Page 3:** [**http://www.linkedin.com/shareArticle?mini=true&url=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare**](http://www.linkedin.com/shareArticle?mini=true&url=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare) **at None**

**Page 3:** [**http://twitter.com/home/?status=Download+FREE+Childrens+Books+@monkeypenbooks+https://bit.ly/2YLY6sM**](http://twitter.com/home/?status=Download+FREE+Childrens+Books+@monkeypenbooks+https://bit.ly/2YLY6sM) **at None**

**Page 3:** [**whatsapp://send?text=Hey,+I+found+some+great+FREE+children's+books+online+from+MonkeyPen.com+https://monkeypen.com/pages/free-childrens-books/?ref=socialshare**](whatsapp://send?text=Hey,+I+found+some+great+FREE+children%27s+books+online+from+MonkeyPen.com+https:/monkeypen.com/pages/free-childrens-books/?ref=socialshare) **at None**

**Page 20:** [**https://monkeypen.com/collections/personalized-books/?ref=pdfbooks**](https://monkeypen.com/collections/personalized-books/?ref=pdfbooks) **at None**

**Page 20:** [**https://monkeypen.com/?ref=pdfbooks**](https://monkeypen.com/?ref=pdfbooks) **at None**

**Page 20:** [**https://monkeypen.com/?ref=pdfbooks**](https://monkeypen.com/?ref=pdfbooks) **at None**

**Page 21:** [**http://www.facebook.com/sharer/sharer.php?u=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare**](http://www.facebook.com/sharer/sharer.php?u=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare) **at None**

**Page 21:** [**http://www.linkedin.com/shareArticle?mini=true&url=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare**](http://www.linkedin.com/shareArticle?mini=true&url=https://monkeypen.com/pages/free-childrens-books/?ref=socialshare) **at None**

**Page 21:** [**http://twitter.com/home/?status=Download+FREE+Childrens+Books+@monkeypenbooks+https://bit.ly/2YLY6sM**](http://twitter.com/home/?status=Download+FREE+Childrens+Books+@monkeypenbooks+https://bit.ly/2YLY6sM) **at None**

**Page 21:** [**whatsapp://send?text=Hey,+I+found+some+great+FREE+children's+books+online+from+MonkeyPen.com+https://monkeypen.com/pages/free-childrens-books/?ref=socialshare**](whatsapp://send?text=Hey,+I+found+some+great+FREE+children%27s+books+online+from+MonkeyPen.com+https:/monkeypen.com/pages/free-childrens-books/?ref=socialshare) **at None**

**Page 21:** [**https://www.maaillustrations.com/?ref=pdfbooks**](https://www.maaillustrations.com/?ref=pdfbooks) **at None**

**Page 21:** [**https://monkeypen.com/?ref=pdfbooks**](https://monkeypen.com/?ref=pdfbooks) **at None**

**Code:**

import fitz

def merge\_pdfs(pdf\_file1, pdf\_file2, output\_pdf):

doc1 = fitz.open(pdf\_file1)

doc2 = fitz.open(pdf\_file2)

merged\_pdf = fitz.open()

for page\_num in range(len(doc1)):

merged\_pdf.insert\_pdf(doc1, from\_page=page\_num, to\_page=page\_num)

for page\_num in range(len(doc2)):

merged\_pdf.insert\_pdf(doc2, from\_page=page\_num, to\_page=page\_num)

merged\_pdf.save(output\_pdf)

doc1.close()

doc2.close()

merged\_pdf.close()

pdf\_file1 = 'TOOTH-FAIRY.pdf'

pdf\_file2 = 'Beauty-and-the-Beast.pdf'

output\_pdf = 'mergedpdf.pdf'

merge\_pdfs(pdf\_file1, pdf\_file2, output\_pdf)

print(f"Merged PDF saved as {output\_pdf}")

**Output:**

Merged PDF saved as merged\_document.pdf

**Code:**import tabula

from tabulate import tabulate

df = tabula.read\_pdf("office.pdf", pages = "all")

try:

df = df[1].values.tolist()

print(tabulate(df, headers = "firstrow"))

print("Table extracted successfully")

except:

print("Table not extracted")

**Output:**

Item1 1000 1050 1100 1200

------- ------ ------ ------ ------

Item2 950 1050 1150 1200

Item3 1100 1200 1200 1300

Table extracted successfully

**Code:**

import fitz

def rotate\_page(pdf\_file, page\_number, rotation\_angle, output\_pdf):

doc = fitz.open(pdf\_file)

page = doc.load\_page(page\_number - 1)

page.set\_rotation(rotation\_angle)

doc.save(output\_pdf)

doc.close()

pdf\_file = 'TOOTH-FAIRY.pdf'

page\_number = int(input("Enter the page number: "))

rotation\_angle = 90

output\_pdf = 'rotate.pdf'

rotate\_page(pdf\_file, page\_number, rotation\_angle, output\_pdf)

print(f"Page {page\_number} rotated by {rotation\_angle} degrees and saved to {output\_pdf}")

**Output:**

Enter the page number: 7

Page 7 rotated by 90 degrees and saved to rotate.pdf

**Inference:**

**LAB 5**

**perform various operations within a postgres sql database using the psycopg2 library in python(operations include table creation,data insertion,updating data,perform filtering of data using where clause an performing joining operations)**

**Code:**

import psycopg2

conn=psycopg2.connect(host='localhost',dbname='postgres',user='postgres',password='123456')

cr=conn.cursor()

create\_table\_query="""CREATE TABLE IF NOT EXISTS employee

(id int primary key not null,

name text not null,

gender text not null,

salary real);"""

cr.execute(create\_table\_query)

print("Table created successfully")

conn.commit()

**Output:**

Table created successfully

**Code:**

cr.execute('''insert into employee values(1,'Anil','Male',45000),

(2,'Anusha','Female',34000),

(3,'Ram','Male',40000),

(4,'Sanjay','Male',38000),

(5,'Alia','Female',50000)''')

conn.commit()

cr.execute('SELECT \* FROM employee')

rows = cr.fetchall()

for row in rows:

print(row)

**Output:**

(1, 'Anil', 'Male', 45000.0)

(2, 'Anusha', 'Female', 34000.0)

(3, 'Ram', 'Male', 40000.0)

(4, 'Sanjay', 'Male', 38000.0)

(5, 'Alia', 'Female', 50000.0)

**Code:**

cr.execute('SELECT id,name FROM employee')

rows=cr.fetchall()

for row in rows:

print(row)

**Output:**

(1, 'Anil')

(2, 'Anusha')

(3, 'Ram')

(4, 'Sanjay')

(5, 'Alia')

**Code:**

**#**updating

cr.execute('''UPDATE employee SET salary = 55000 WHERE id = 1''')conn.commit()

**Output:**

(1,’Anil’,’Male’,55000.0)

**Code:**

cr.execute('''SELECT \* FROM employee WHERE salary >= 40000''')

rows = cr.fetchall()

for row in rows:

print(row)

**Output:**

(3, 'Ram', 'Male', 40000.0)

(5, 'Alia', 'Female', 50000.0)

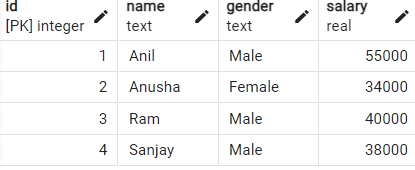
(1, 'Anil', 'Male', 55000.0)

**Code:**

cr.execute('''DELETE FROM employee WHERE id=5''')

conn.commit()

**Output:**



**Code:**

cr.execute('''SELECT \* FROM employee ORDER BY salary desc''')

rows=cr.fetchall()

for row in rows:

print(row)

**Output:**

(1, 'Anil', 'Male', 55000.0)

(3, 'Ram', 'Male', 40000.0)

(4, 'Sanjay', 'Male', 38000.0)

(2, 'Anusha', 'Female', 34000.0)

**Code:**

cr.execute('''select min(salary) from employee''')

rows=cr.fetchall()

for row in rows:

print(row)

**Output:**

(34000.0,)

**Code:**

conn=psycopg2.connect(host='localhost',dbname='postgres',user='postgres',password='123456')

cr=conn.cursor()

create\_table\_query='''CREATE TABLE IF NOT EXISTS department (

id INT PRIMARY KEY NOT NULL,

name TEXT NOT NULL,

location TEXT);'''

cr.execute(create\_table\_query)

print("Table created successfully")

conn.commit()

**Output:**

Table created successfully

**Code:**

cr.execute('''insert into department values(1,'IT','Bangalore'),

(2,'HR','Delhi'),

(3,'Finance','Chennai'),

(4,'Marketing','Bangalore'),

(5,'HR','Gurgoan')''')

conn.commit()

cr.execute('''select\*from department''')

rows=cr.fetchall()

for row in rows:

print(row)

**Output:**

(1, 'IT', 'Bangalore')

(2, 'HR', 'Delhi')

(3, 'Finance', 'Chennai')

(4, 'Marketing', 'Bangalore')

(5, 'HR', 'Gurgoan')

**Code:**

SELECT e.id, e.name AS employee\_name, d.name AS department\_name FROM employee e

CROSS JOIN department d;

**Output:**

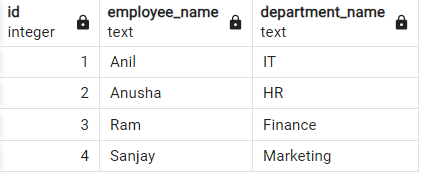


**Code:**

SELECT e.id, e.name AS employee\_name,d.name AS department\_name FROM employee e

Left JOIN department d ON e.id = d.id

**Output:**



**Inference:**

**LAB 6**

**To perform data cleaning on a dataset. Clean the dataset by handling missing values, converting data types and then explore the cleaned data to draw insights.**

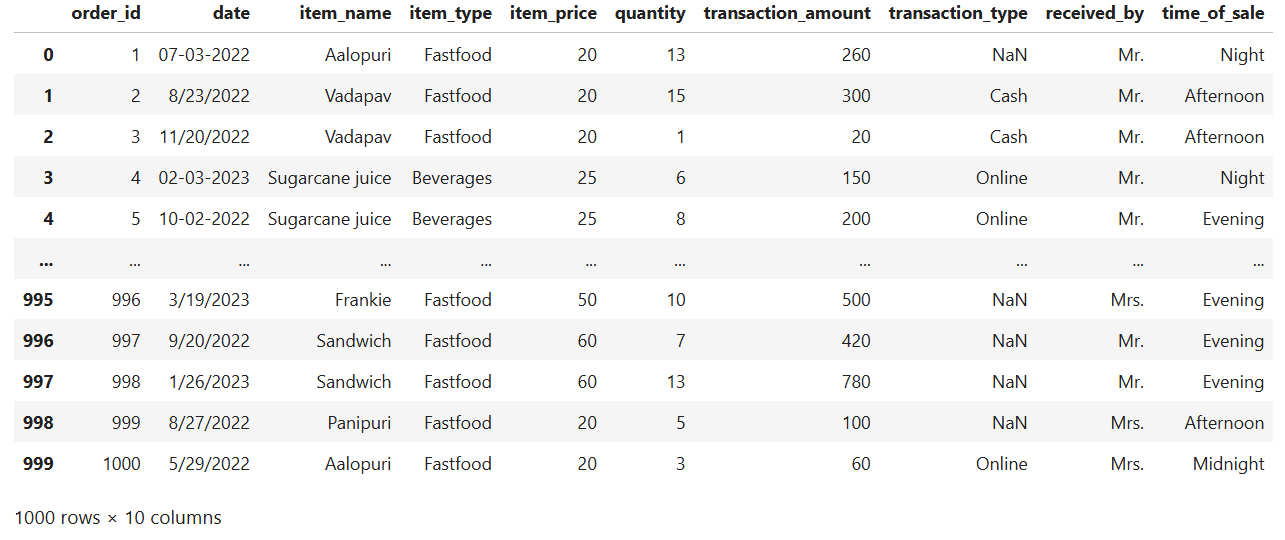
**Code:**

import pandas as pd

df=pd.read\_csv("Fast Food Sales.csv")

display(df)

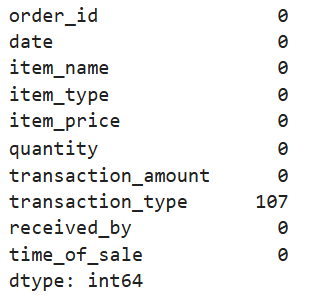
**Output:**



**Code:**

df.isna().sum()

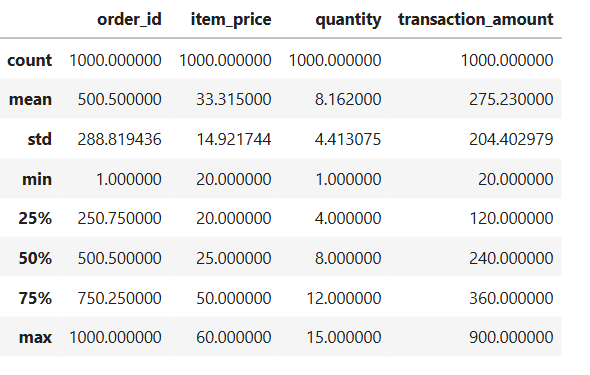
**Output:**

****

**Code:**

df.describe()

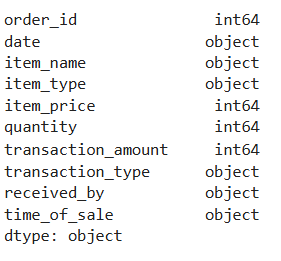
**Output:**

****

**Code:**

df.dtypes

**Output:**

****

**Code:**

df.columns

**Output:**

Index(['order\_id', 'date', 'item\_name', 'item\_type', 'item\_price', 'quantity', 'transaction\_amount', 'transaction\_type', 'received\_by','time\_of\_sale'],dtype='object')

**Code:**

#Standardize Date Format

import pandas as pd

from datetime import datetime

def parse\_date(date\_str):

date\_formats = [

'%d-%m-%Y', # Format like 07-03-2022

'%m/%d/%Y', # Format like 8/23/2022

'%m-%d-%Y', # Format like 10-02-2022

'%Y-%m-%d', # Format like 2022-07-03

'%d/%m/%Y', # Format like 02/03/2023

]

if pd.isna(date\_str) or not isinstance(date\_str, str):

return pd.NaT

for fmt in date\_formats:

try:

return datetime.strptime(date\_str, fmt)

except ValueError:

continue

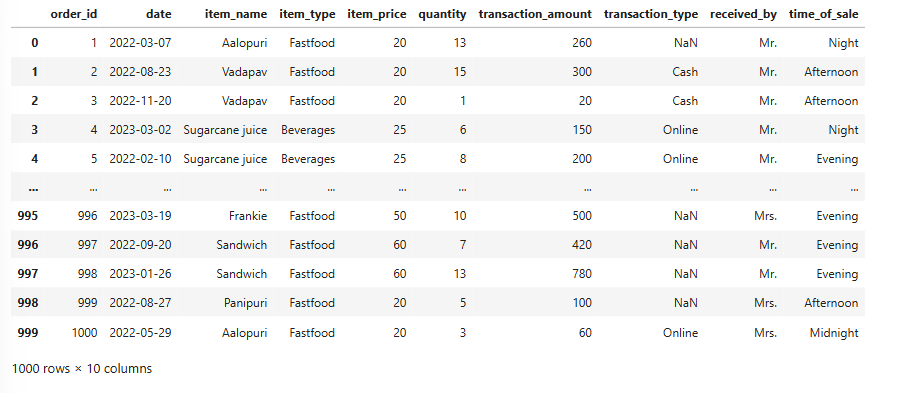
return pd.NaT

df['date'] = df['date'].apply(parse\_date)

df['date'] = pd.to\_datetime(df['date'], errors='coerce')

display(df)

**Output:**

****

**Code:**

# Calculate correlation matrix

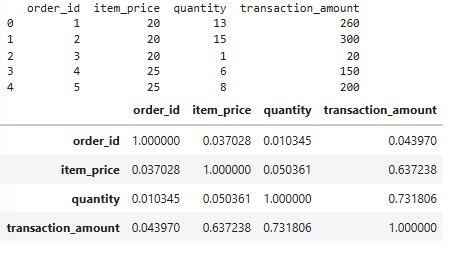
numerical\_df = df.select\_dtypes(include=['number'])

print(numerical\_df.head())

correlation\_matrix = numerical\_df.corr()

display(correlation\_matrix)

**Output:**

****

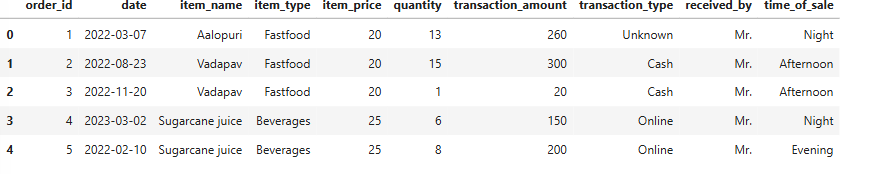
**Code:**

#Handle Missing Values

df['transaction\_type'] = df['transaction\_type'].fillna('Unknown')

df.head()

**Output:**

****

**Code:**

#Ensure Consistent Data Types

df.dtypes

df['order\_id'] = df['order\_id'].astype(int)

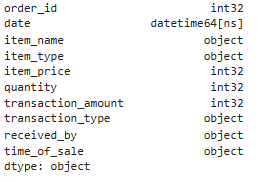
df['item\_price'] = df['item\_price'].astype(int)

df['quantity'] = df['quantity'].astype(int)

df['transaction\_amount'] = df['transaction\_amount'].astype(int)

df.dtypes

Output:



**Code:**

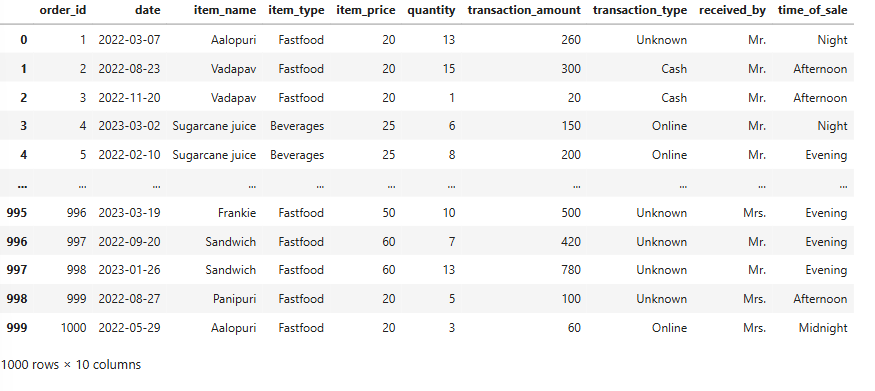
#Remove Duplicates

duplicates=df[df.duplicated()]

df=df.drop\_duplicates()

display(df)

**Output:**



**Code:**

#outliers

import matplotlib.pyplot as plt

import seaborn as sns

Q1 = df['transaction\_amount'].quantile(0.25)

Q3 = df['transaction\_amount'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers = df[(df['transaction\_amount'] < lower\_bound) | (df['transaction\_amount'] > upper\_bound)]

print("Outliers:")

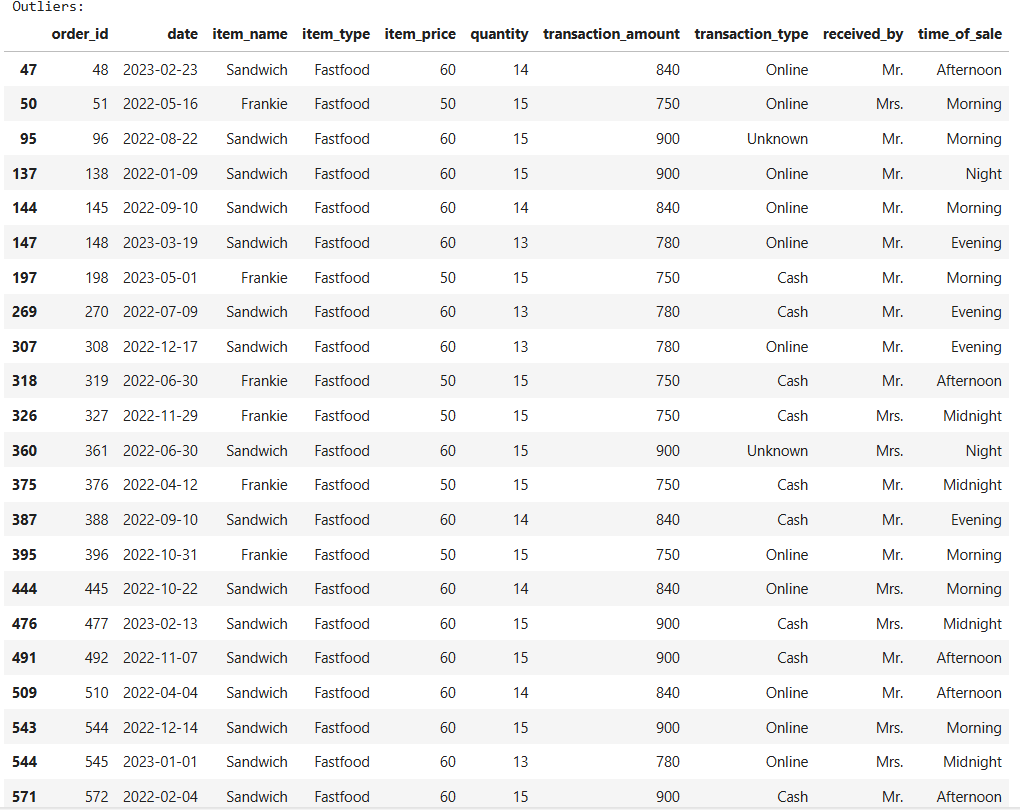
display(outliers)

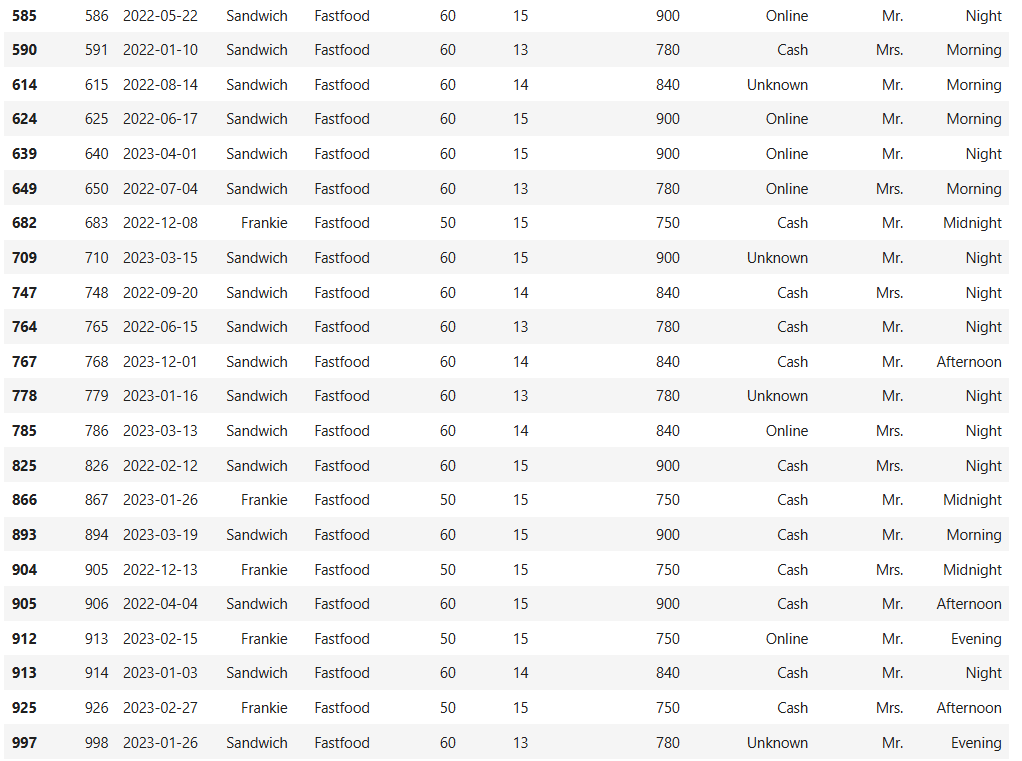
plt.figure(figsize=(10, 6))

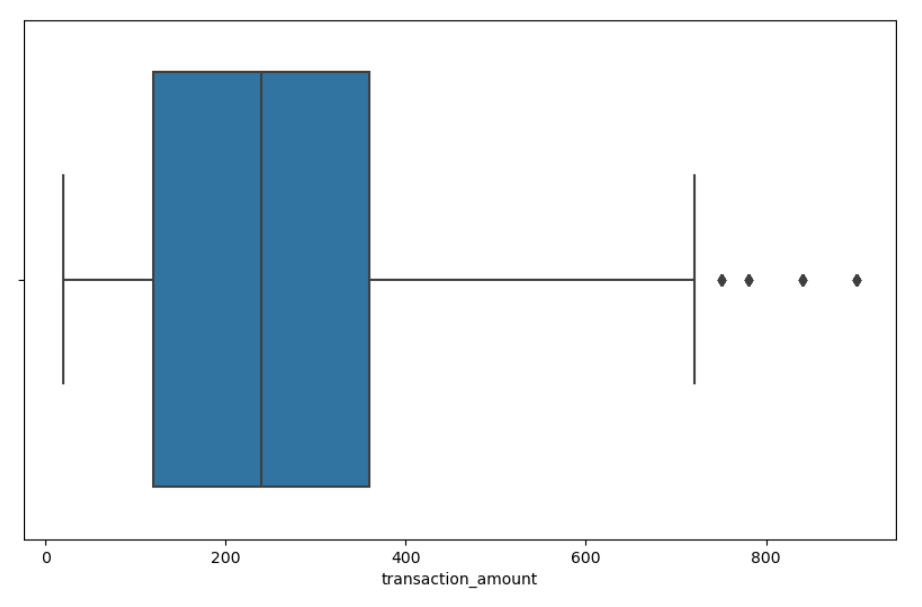
sns.boxplot(x=df['transaction\_amount'])

plt.show()

**Output:**







**Code:**

# Cap the outliers

df['transaction\_amount'] = df['transaction\_amount']

df.loc[df['transaction\_amount'] > upper\_bound, 'transaction\_amount'] = upper\_bound

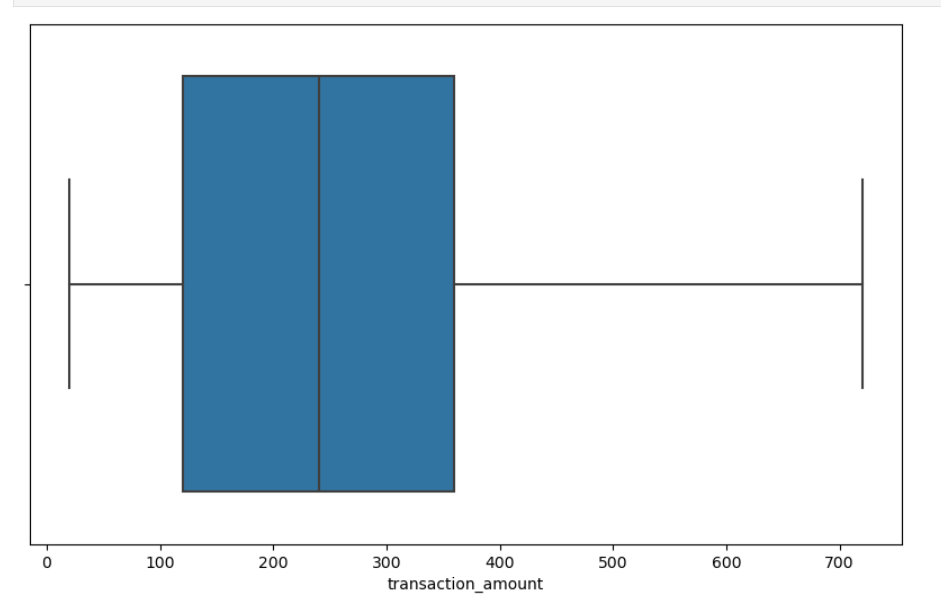
df.loc[df['transaction\_amount'] < lower\_bound, 'transaction\_amount'] = lower\_bound

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

sns.boxplot(x=df['transaction\_amount'])

plt.show()

**Output:**



**Code:**

# Replace 'Fastfood' with 'Fast Food'

df['item\_type'] = df['item\_type'].str.replace(r'Fastfood', 'Fast Food', regex=True)

display(df)

**Output:**



**Code:**

df['item\_name'].unique()

**Output:**

array(['Aalopuri', 'Vadapav', 'Sugarcane juice', 'Panipuri', 'Frankie','Sandwich', 'Cold coffee'], dtype=object)

**Code:**

item\_popularity = df.groupby('item\_name')['quantity'].sum().reset\_index().sort\_values(by='quantity', ascending=False)

plt.figure(figsize=(12, 6))

sns.barplot(x='quantity', y='item\_name', data=item\_popularity)

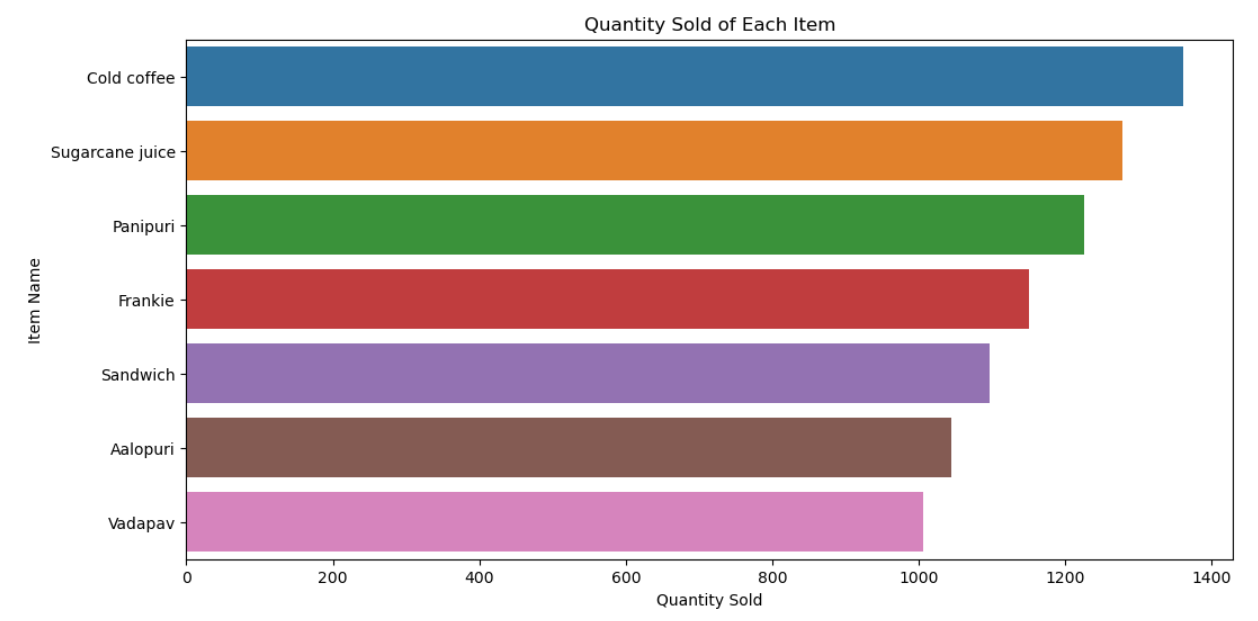
plt.title('Quantity Sold of Each Item')

plt.xlabel('Quantity Sold')

plt.ylabel('Item Name')

plt.show()

**Output:**



**Code:**

transaction\_type\_distribution = df['transaction\_type'].value\_counts().reset\_index()

transaction\_type\_distribution.columns = ['transaction\_type', 'count']

plt.figure(figsize=(7, 4))

sns.barplot(x='transaction\_type', y='count', data=transaction\_type\_distribution)

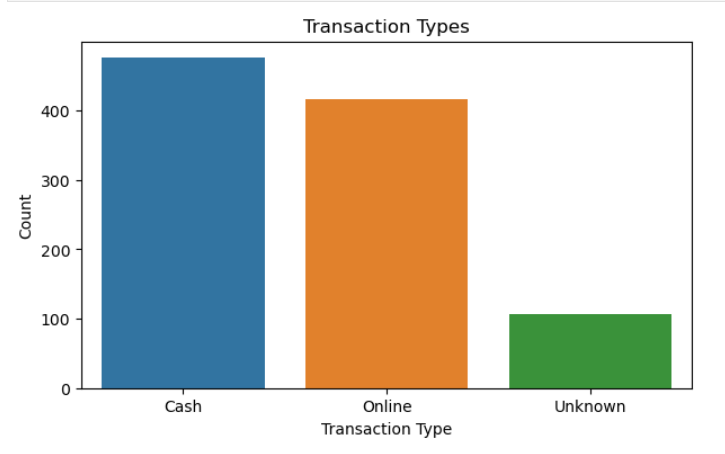
plt.title('Transaction Types')

plt.xlabel('Transaction Type')

plt.ylabel('Count')

plt.show()

**Output:**



**Code:**

time\_of\_sale\_distribution = df.groupby('time\_of\_sale')['transaction\_amount'].sum().reset\_index().sort\_values(by='transaction\_amount', ascending=False)

plt.figure(figsize=(9, 5))

sns.barplot(x='transaction\_amount', y='time\_of\_sale', data=time\_of\_sale\_distribution)

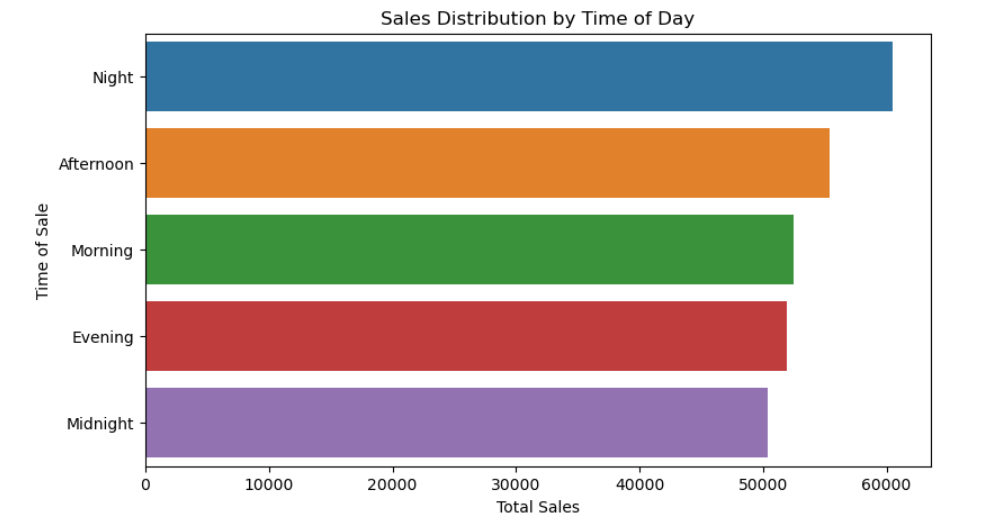
plt.title('Sales Distribution by Time of Day')

plt.xlabel('Total Sales')

plt.ylabel('Time of Sale')

plt.show()

**Output:**



**Code:**

#Total sales per month

df['month'] = df['date'].dt.to\_period('M').astype(str)

monthly\_sales = df.groupby('month')['transaction\_amount'].sum().reset\_index()

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

sns.lineplot(x='month', y='transaction\_amount', data=monthly\_sales, marker='o')

plt.title('Total Sales per Month')

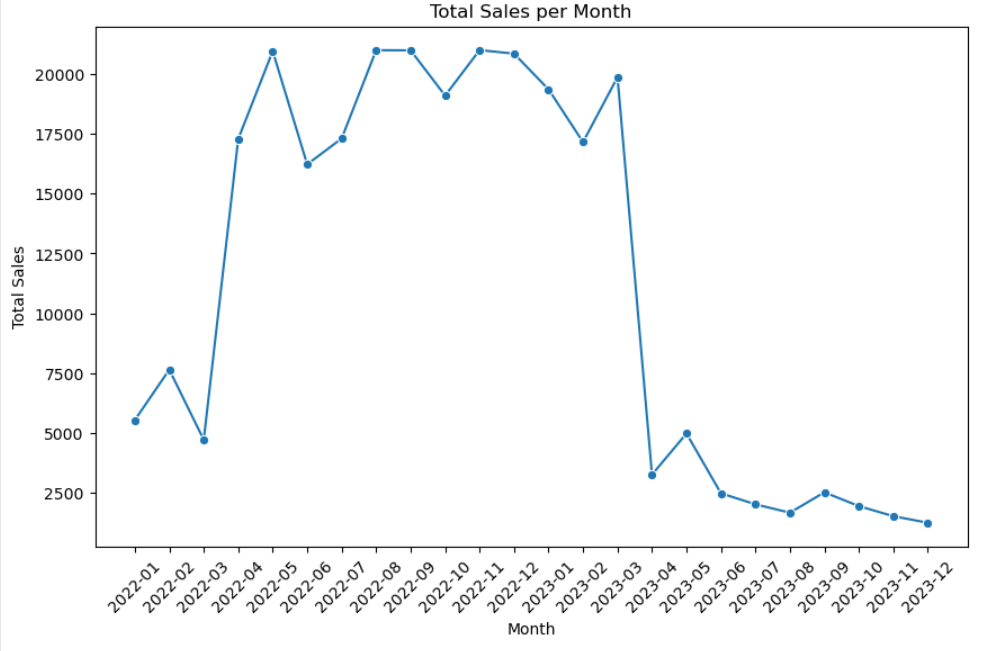
plt.xlabel('Month')

plt.ylabel('Total Sales')

plt.xticks(rotation=45)

plt.show()

**Output:**



**Inference:**

**LAB 7**

**To perform exploratory data analysis. Clean the dataset and perform required preprocessing explore the cleaned data by performing data visualization and draw conclusion.**

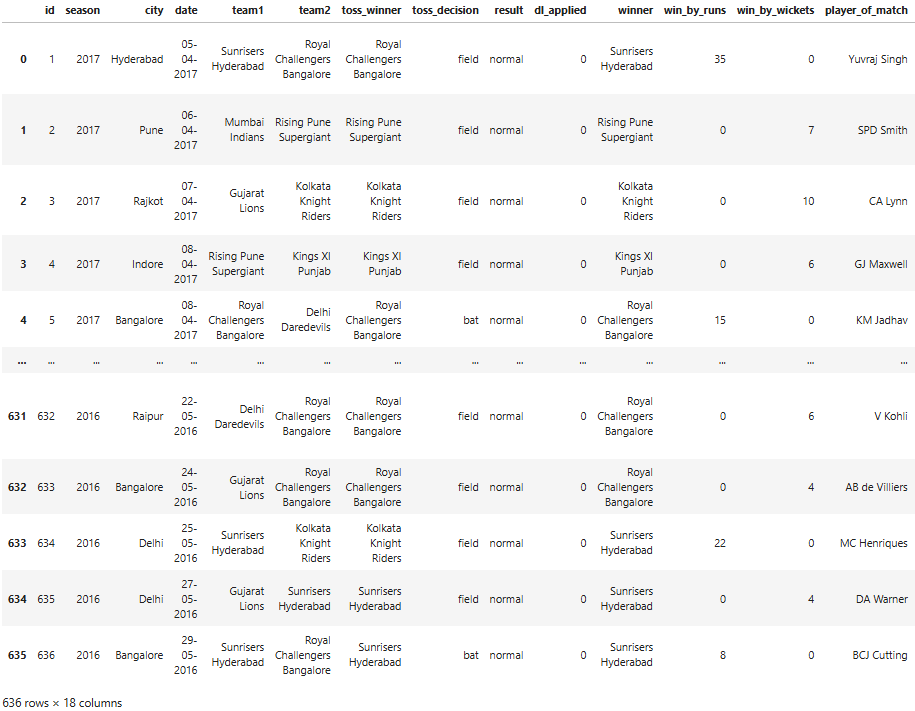
**Code:**

import pandas as pd

df=pd.read\_csv("IPL\_matches.csv")

df

**Output:**

****

**Code:**

df.isna().sum()

**Output:**

id 0

season 0

city 7

date 0

team1 0

team2 0

toss\_winner 0

toss\_decision 0

result 0

dl\_applied 0

winner 3

win\_by\_runs 0

win\_by\_wickets 0

player\_of\_match 3

venue 0

umpire1 1

umpire2 1

umpire3 636

dtype: int64

**Code:**

df.drop(columns=['umpire3','id','dl\_applied'],inplace=True)

df.columns

**output:**

Index(['season', 'city', 'date', 'team1', 'team2', 'toss\_winner','toss\_decision', 'result', 'winner','win\_by\_runs', 'win\_by\_wickets','player\_of\_match', 'venue', 'umpire1', 'umpire2'], dtype='object')

**Code:**

df.dropna(inplace=True)

df.isna().sum()

**Output:**

season 0

city 0

date 0

team1 0

team2 0

toss\_winner 0

toss\_decision 0

result 0

winner 0

win\_by\_runs 0

win\_by\_wickets 0

player\_of\_match 0

venue 0

umpire1 0

umpire2 0

dtype: int64

**Code:**

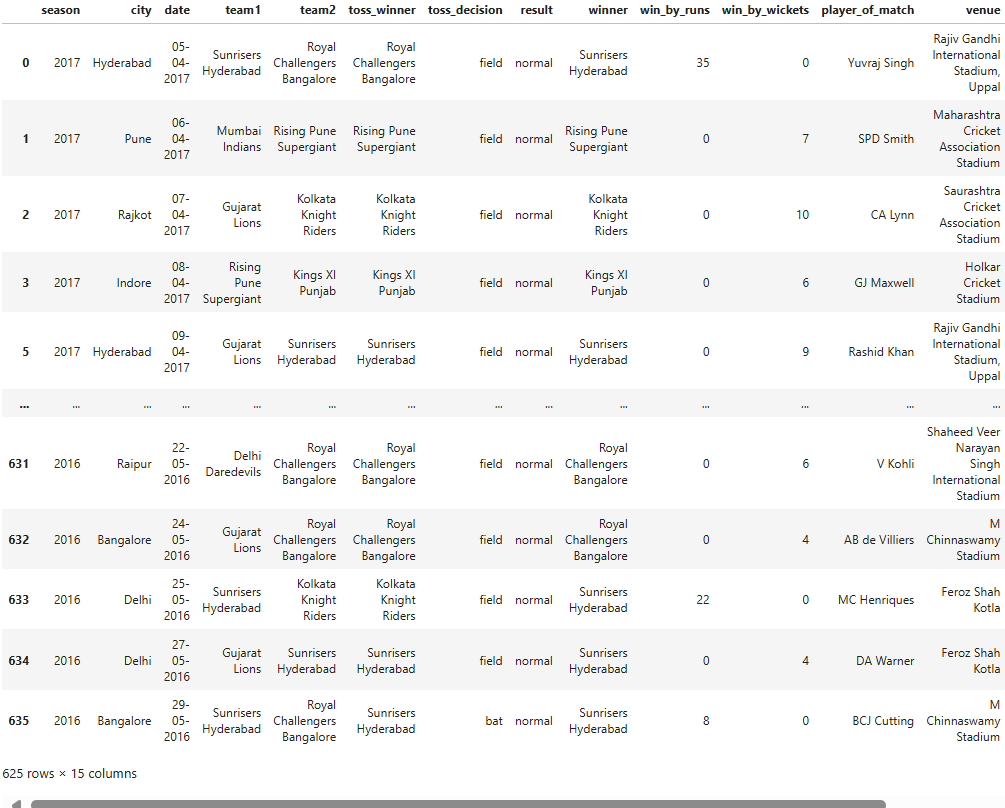
#Remove Duplicates

duplicates=df[df.duplicated()]

df=df.drop\_duplicates()

display(df)

**Ouput:**



**Code:**

df.dtypes

**output:**

season int64

city object

date object

team1 object

team2 object

toss\_winner object

toss\_decision object

result object

winner object

win\_by\_runs int64

win\_by\_wickets int64

player\_of\_match object

venue object

umpire1 object

umpire2 object

dtype: object

**Code:**

df['date'] = pd.to\_datetime(df['date'], format='%d-%m-%Y')

df.dtypes

**Ouput:**

season int64

city object

date datetime64[ns]

team1 object

team2 object

toss\_winner object

toss\_decision object

result object

winner object

win\_by\_runs int64

win\_by\_wickets int64

player\_of\_match object

venue object

umpire1 object

umpire2 object

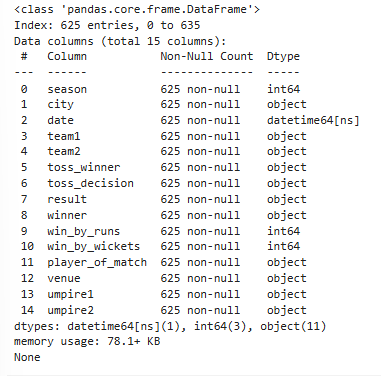
dtype: object

**Basic statistic**

**Code:**

print(df.info())

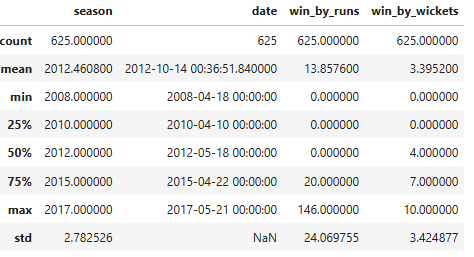
**Output:**



**Code:**

df.describe()

**output:**

****

**Univariate Analysis**

**Code:**

# Plot the frequency of 'winner'

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(9, 5))

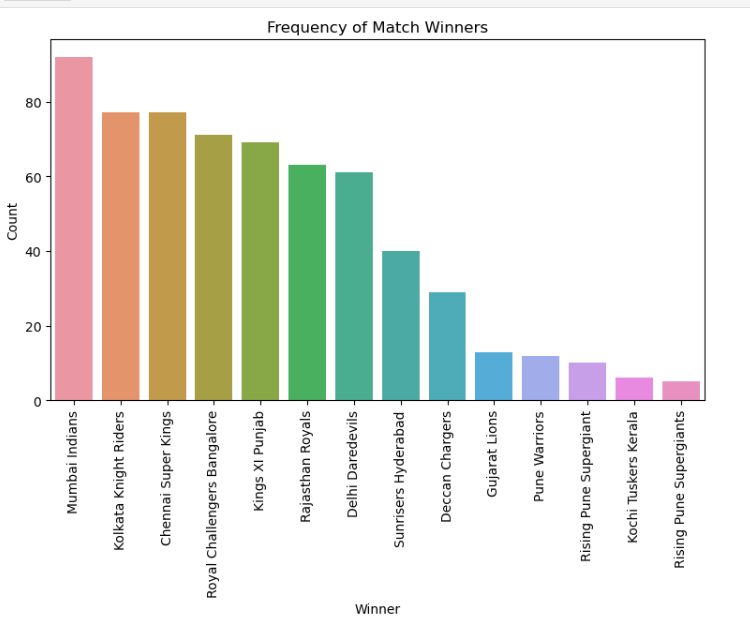
sns.countplot(x='winner', data=df, order=df['winner'].value\_counts().index)

plt.title('Frequency of Match Winners')

plt.xlabel('Winner')

plt.ylabel('Count')

plt.show()



**Code:**

# Count the occurrences of each toss decision

toss\_decision\_counts = df['toss\_decision'].value\_counts()

# Plot pie chart

plt.figure(figsize=(5,3))

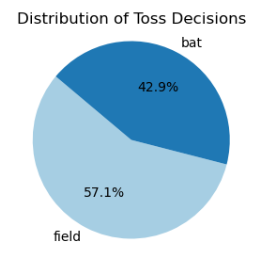
plt.pie(toss\_decision\_counts, labels=toss\_decision\_counts.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired(range(len(toss\_decision\_counts))))

plt.title('Distribution of Toss Decisions')

plt.axis('equal')

plt.show()

**Output:**



**Code:**

# Count the occurrences of each city

import matplotlib.pyplot as plt

city\_counts = df['city'].value\_counts()

top\_5\_cities = city\_counts.head(5)

plt.figure(figsize=(7, 4))

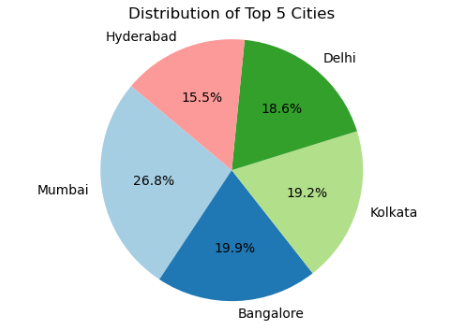
plt.pie(top\_5\_cities, labels=top\_5\_cities.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired(range(len(top\_5\_cities))))

plt.title('Distribution of Top 5 Cities')

plt.axis('equal')

plt.show()

**Output:**



**Code:**

#Distribution of Wins by Runs

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

sns.histplot(df['win\_by\_runs'], bins=30, kde=True)

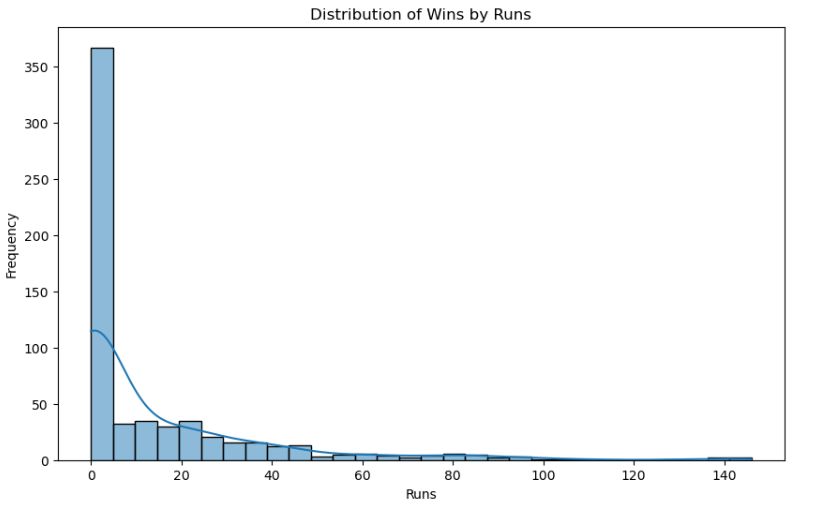
plt.title('Distribution of Wins by Runs')

plt.xlabel('Runs')

plt.ylabel('Frequency')

plt.show()

**Output:**



**Code:**

player\_award\_counts = df['player\_of\_match'].value\_counts()

print(player\_award\_counts.head(5))

top\_players = player\_award\_counts.head(5)

plt.figure(figsize=(7, 4))

ax = sns.barplot(x=top\_players.index, y=top\_players.values, palette='viridis')

plt.title('Top 5 Players of the Match')

plt.xlabel('Player')

plt.ylabel('Number of Awards')

ax.yaxis.get\_major\_locator().set\_params(integer=True)

plt.xticks(rotation=90)

plt.show()

**Ouput:**

player\_of\_match

CH Gayle 18

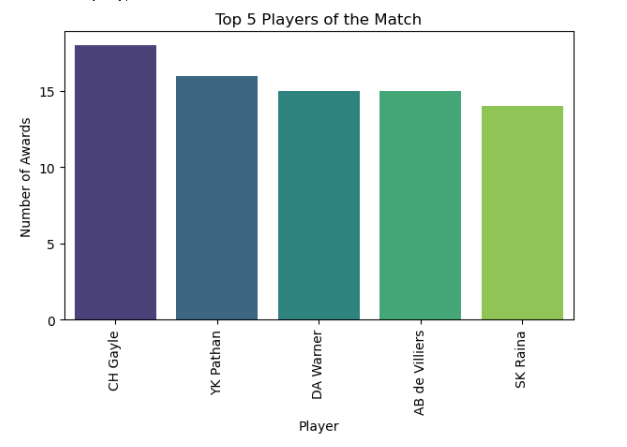
YK Pathan 16

DA Warner 15

AB de Villiers 15

SK Raina 14

Name: count, dtype: int64



**Bivariate Analysis**

**Code**

# Compute the correlation matrix

correlation\_matrix = df[['win\_by\_runs', 'win\_by\_wickets']].corr()

print("Correlation Matrix:")

print(correlation\_matrix)

**output:**

Correlation Matrix:

win\_by\_runs win\_by\_wickets

win\_by\_runs 1.000000 -0.571653

win\_by\_wickets -0.571653 1.000000

**Code:**

# Get the top 5 teams based on average wins by runs

average\_wins\_by\_team = df.groupby('team1')['win\_by\_runs'].mean()

top\_5\_teams = average\_wins\_by\_team.nlargest(5)

plt.figure(figsize=(6, 4))

sns.barplot(x=top\_5\_teams.index, y=top\_5\_teams.values, palette='viridis')

plt.title('Average Wins by Runs by Top 5 Teams')

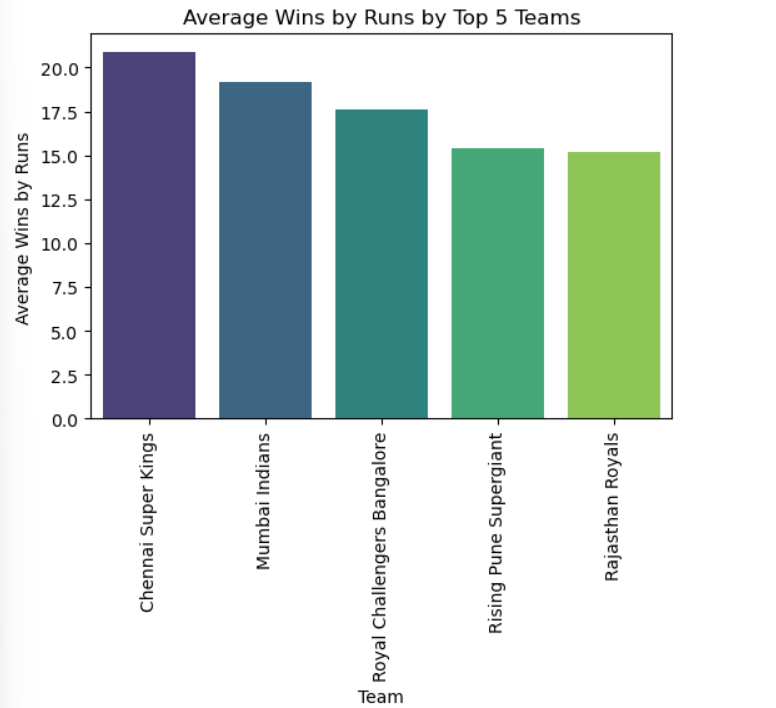
plt.xlabel('Team')

plt.ylabel('Average Wins by Runs')

plt.xticks(rotation=90)

plt.show()

**Output:**



**Multivariate Analysis**

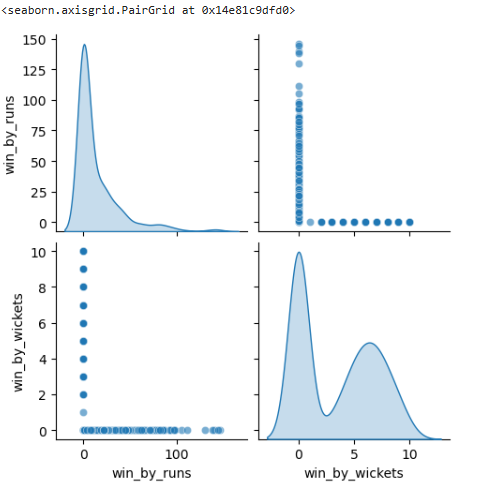
**Code:**

numerical\_columns = ['win\_by\_runs', 'win\_by\_wickets']

df\_numerical = df[numerical\_columns]

sns.pairplot(df\_numerical, diag\_kind='kde', plot\_kws={'alpha':0.6})

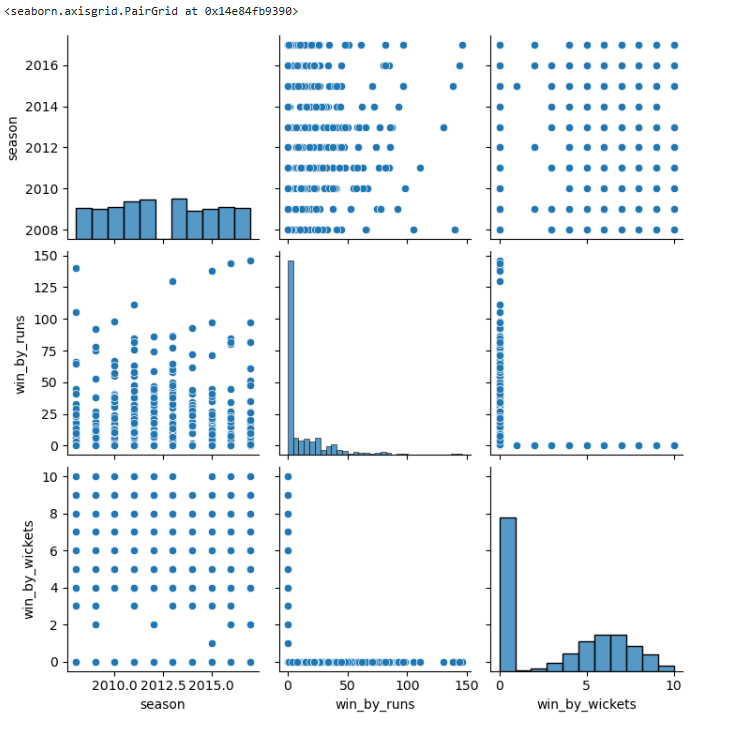
**Output:**



**Code:**

sns.pairplot(df)

**Output:**



**Inference:**

**LAB 8**

**Image processing: Perform resize image ,change filters ,perform rotation, edge detection and rgb percentage**

**Code:**

import cv2

from google.colab.patches import cv2\_imshow

from PIL import Image

import matplotlib.pyplot as plt

from IPython.display import display

filepath = 'disnep.png'

image\_cv2 = cv2.imread(filepath)

image\_pil = Image.open(filepath)

while True:

    print("\nImage Processing Menu")

    print("1. Display Original Image (OpenCV)")

    print("2. Display Grayscale Image (OpenCV)")

    print("3. Resize Image ")

    print("4. Rotate Image ")

    print("5. Blur Image (OpenCV)")

    print("6. Display Image Info (PIL)")

    print("7. Edge detection")

    print("8. RGB Pie Chart")

    print("9. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':

        print("Original Image (OpenCV)")

        cv2\_imshow(image\_cv2)

    elif choice == '2':

        gray\_image\_cv2 = cv2.cvtColor(image\_cv2, cv2.COLOR\_BGR2GRAY)

        print("Grayscale Image (OpenCV)")

        cv2\_imshow(gray\_image\_cv2)

    elif choice == '3':

        new\_size = (300, 300)

        resized\_image\_cv2 = cv2.resize(image\_cv2, new\_size)

        cv2.imwrite('resized\_image.png', resized\_image\_cv2)

        print(f"Resized image saved as resized\_image.png with size {new\_size}")

        cv2\_imshow(resized\_image\_cv2)

    elif choice == '4':

        angle = 90

        rotated\_image\_pil = image\_pil.rotate(angle, expand=True)

        rotated\_image\_pil.save('rotated\_image.png')

        print(f"Rotated image saved as rotated\_image.png")

        display(rotated\_image\_pil)

    elif choice == '5':

        blurred\_image\_cv2 = cv2.GaussianBlur(image\_cv2, (5, 5), 0)

        print("Original Image (OpenCV)")

        cv2\_imshow(image\_cv2)

        print("Blurred Image (OpenCV)")

        cv2\_imshow(blurred\_image\_cv2)

    elif choice == '6':

        print("Image Information:")

        print(f"Format: {image\_pil.format}")

        print(f"Size: {image\_pil.size}")

        print(f"Mode: {image\_pil.mode}")

        display(image\_pil)

    elif choice == '7':

        edges = cv2.Canny(image\_cv2, 40, 60)

        cv2\_imshow(edges)

        plt.show()

    elif choice == '8':

        chans = cv2.split(image\_cv2)

        total\_pixels = image\_cv2.shape[0] \* image\_cv2.shape[1]

        colors = ('Blue', 'Green', 'Red')

        pixel\_sums = [chan.sum() for chan in chans]

        plt.figure()

        plt.title("RGB Proportion Pie Chart")

        plt.pie(pixel\_sums, labels=colors, colors=colors, autopct='%1.1f%%')

        plt.show()

    elif choice == '9':

        print("Exit")

        break

    else:

        print("Invalid choice. Please try again.")

**OUTPUT:**

Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 2

Grayscale Image (OpenCV)



Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 3

Resized image saved as resized\_image.png with size (300, 300)



Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 4

Rotated image saved as rotated\_image.png



Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 5

Original Image (OpenCV)





Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 6

Image Information:

Format: PNG

Size: (475, 473)

Mode: RGBA

Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 7



Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 8

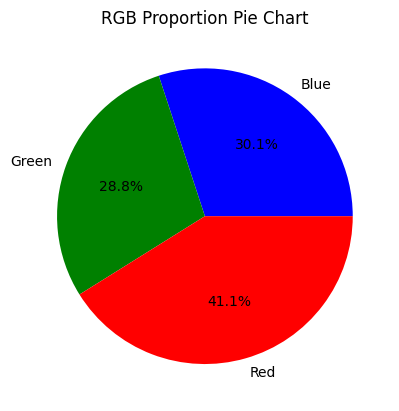


Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 1

Original Image (OpenCV)



Image Processing Menu

1. Display Original Image (OpenCV)

2. Display Grayscale Image (OpenCV)

3. Resize Image

4. Rotate Image

5. Blur Image (OpenCV)

6. Display Image Info (PIL)

7. Edge detection

8. RGB Pie Chart

9. Exit

Enter your choice: 9

Exit

**Inference:**

**LAB 9**

**Video processing**

**Code:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

from google.colab.patches import cv2\_imshow

video=cv2.VideoCapture("/content/world.mp4")

if not video.isOpened():

    print("error")

else:

    print("Video is uploaded successfully")

while True:

    print("\n MENU:")

    print("1. Display a frame")

    print("2. Get video info")

    print("3. Resize and display a frame")

    print("4. convert frame to grayscale")

    print("5. Display multiple frames as thumbnails")

    print("6. Apply Gausssian blur to a frame")

    print("7. calculate and display RGB percentage")

    print("8. Edge detection on a frame")

    print("9. Exit")

    choice=int(input("Enter your choice:"))

    if choice==1:

        ret,frame=video.read()

        if ret:

            cv2\_imshow(frame)

        else:

            print("Error:could not read a frame")

    elif choice==2:

        frame\_count=int(video.get(cv2.CAP\_PROP\_FRAME\_COUNT))

        frame\_width=int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

        frame\_height=int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

        print("Frame count:",frame\_count)

        print("Frame width:",frame\_width)

        print("frame height:",frame\_height)

    elif choice==3:

        ret,frame=video.read()

        if ret:

            resized\_frame=cv2.resize(frame,(300,300))

            cv2\_imshow(resized\_frame)

        else:

            print("Error: could not read frame")

    elif choice==4:

        ret,frame=video.read()

        if ret:

            gray\_frame=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

            cv2\_imshow(gray\_frame)

        else:

            print("Error: could not read frame")

    elif choice==5:

        num\_frame=7

        frame\_interval=int(video.get(cv2.CAP\_PROP\_FRAME\_COUNT)/num\_frame)

        plt.figure(figsize=(20,10))

        for i in range(num\_frame):

            video.set(cv2.CAP\_PROP\_POS\_FRAMES,i\*frame\_interval)

            ret,frame=video.read()

            if ret:

                frame=cv2.cvtColor(frame,cv2.COLOR\_BGR2RGB)

                plt.subplot(2,5,i+1)

                plt.imshow(frame)

                plt.axis('off')

                plt.title(f'Frame{i\*frame\_interval}')

                plt.tight\_layout()

                plt.show()

            else:

                break

    elif choice==6:

        ret, frame = video.read()

        if ret:

            blurred\_frame = cv2.GaussianBlur(frame, (5, 5), 0)

            cv2\_imshow(blurred\_frame)

        else:

            print("Error: Could not read frame")

    elif choice == 7:

        ret, frame = video.read()

        if ret:

            sum\_b = np.sum(frame[:, :, 0])

            sum\_g = np.sum(frame[:, :, 1])

            sum\_r = np.sum(frame[:, :, 2])

            total\_sum = sum\_b + sum\_g + sum\_r

            percent\_b = (sum\_b / total\_sum) \* 100

            percent\_g = (sum\_g / total\_sum) \* 100

            percent\_r = (sum\_r / total\_sum) \* 100

            labels = ['Blue', 'Green', 'Red']

            sizes = [percent\_b, percent\_g, percent\_r]

            colors = ['blue', 'green', 'red']

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

            plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)

            plt.title('RGB Percentage in Frame')

            plt.axis('equal')

            plt.show()

        else:

            print("Error: Could not read frame")

    elif choice == 8:

        ret, frame =video.read()

        if ret:

            new\_frame = cv2.Canny(frame, 100, 200)

            cv2\_imshow(new\_frame)

        else:

            print("Error: Could not read frame")

    elif choice==9:

        print("Exit")

        break

    else:

        print("Invalid choice.Enter valid choice")

        ch=input("Do you want to continue?(yes/no):").strip().lower()

        if ch!='yes':

            break

video.release()

**Output:**

Video is uploaded successfully

MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:2

Frame count: 537

Frame width: 2160

frame height: 4096

MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

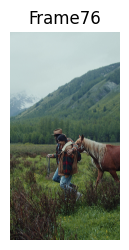
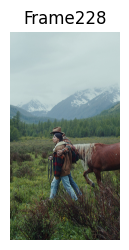
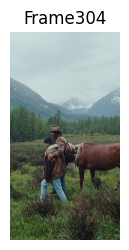
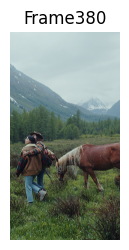
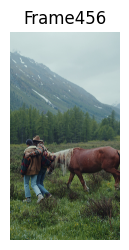
6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:5

MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:6

­ 

MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

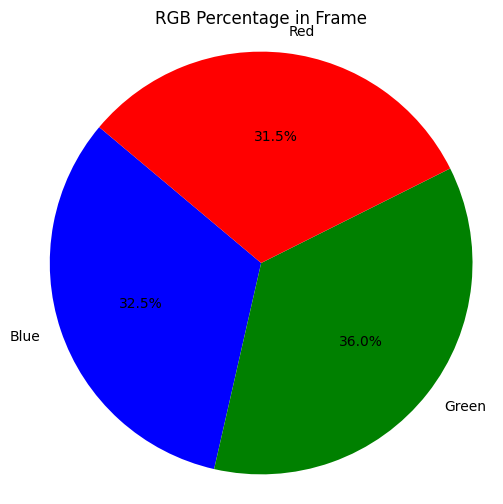
6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:7



MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:8



MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:4

MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:1

  
MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:3



MENU:

1. Display a frame

2. Get video info

3. Resize and display a frame

4. convert frame to grayscale

5. Display multiple frames as thumbnails

6. Apply Gausssian blur to a frame

7. calculate and display RGB percentage

8. Edge detection on a frame

9. Exit

Enter your choice:9

Exit

**Inference:**