ROYAL COLLEGE OF ARTS, SCIENCE & COMMERCE

(AUTONOMOUS)

Mira Road (East), Mira Bhayandar, Maharashtra 401107.

DEPARTMENT OF DATA SCIENCE

M.Sc. Data Science (Part I) – **Semester I**

Course Code: **RPDSDVMJP102**

Course Name: **Data Analysis & Visualization**

Practical Journal

(Academic Year: 2025-2026)

Seat No:

Since 1989

Royal Higher Education Society's

ROYAL COLLEGE
OF ARTS, SCIENCE & COMMERCE

Empowerment through Value Education

(A Minority Autonomous Institution)

Devoted to Serve

ROYAL COLLEGE OF ARTS, SCIENCE & COMMERCE (AUTONOMOUS)

Mira Road (East), Mira Bhayandar, Maharashtra 401107.

Department of Data Science

CERTIFICATE

This is to certify that **Mr./Ms.___Hammad Ansari**, a student of **M.Sc. Data Science (Part I) - Semester I**, having **Roll No. 15** and **Seat No.** _______________ has satisfactorily completed **10 Practicals** in the subject of **Data Analysis & Visualization** as part of the **M.Sc. Data Science Programme** during the academic year **2025–2026.**

Place:

Date:

Subject In-charge

Co-Ordinator,
M.Sc. Data Science

Signature of Examiner

INDEX

Practical No: 01

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Implement data loading, storage, and file formats by reading data, storing them in text format, and demonstrating data wrangling on a dataset

Theory: Practical 1-A

Data loading, or simply loading, is a part of data processing where data is moved between two systems so that it ends up in a staging area on the target system.

With the traditional extract, transform and load (ETL) method, the load job is the last step, and the data that is loaded has already been transformed. With the alternative method extract, load and transform (ELT), the loading job is the middle step, and the transformed data is loaded in its original format for data transformation in the target system.

Traditionally, loading jobs on large systems have taken a long time, and have typically been run at night outside a company's opening hours.

Two main goals of data loading are to obtain fresher data in the systems after loading, and that the loading is fast so that the data can be updated frequently. For full data refresh, faster loading can be achieved by turning off referential integrity, secondary indexes and logging, but this is usually not allowed with incremental update or trickle feed.

Data loading can be done either by complete update (immediate), incremental loading and updating (immediate), or trickle feed (deferred). The choice of technique may depend on the amount of data that is updated, changed or added, and how up-to-date the data must be. The type of data delivered by the source system, and whether historical data delivered by the source system can be trusted are also important factors.

Data loading ensures that data is available in a centralized location for easy access and use by all teams in an organization. It also allows for accurate and up-to-date information to be populated in a target system for reporting, analysis, and decision-making.

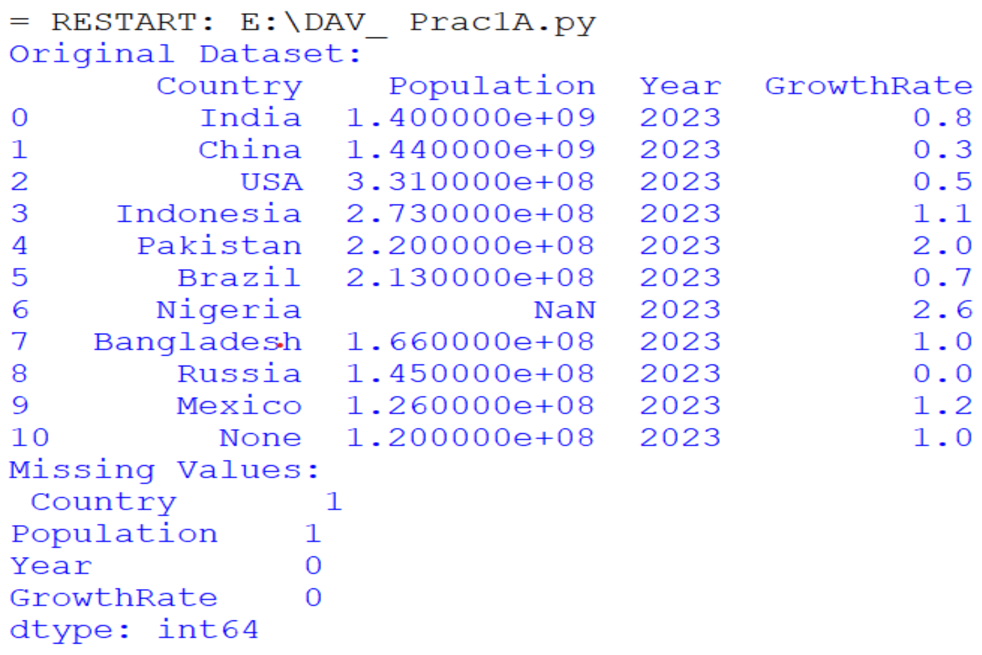
In the past, organizations used manual processes to transfer data, which was time-consuming and prone to errors. Today, advanced platforms like Airbyte offer no-code solutions to simplify the data loading process.

Code:

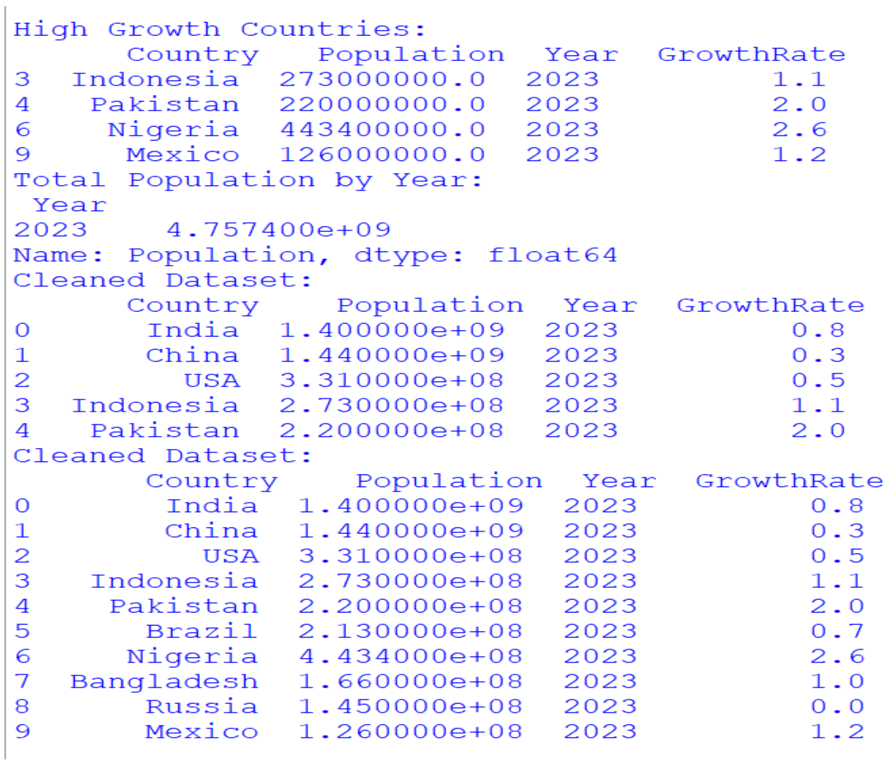
```
import pandas as pd

import numpy as np

# Sample population dataset

data = {
    'Country': ['India', 'China', 'USA', 'Indonesia', 'Pakistan', 'Brazil', 'Nigeria', 'Bangladesh',
'Russia', 'Mexico', None],
    'Population': [1400000000, 1440000000, 331000000, 273000000, 220000000, 213000000,
None, 166000000, 145000000, 126000000, 120000000],
    'Year': [2023]*11,
    'GrowthRate': [0.8, 0.3, 0.5, 1.1, 2.0, 0.7, 2.6, 1.0, 0.0, 1.2, 1.0]
}
df = pd.DataFrame(data)
# STEP 1: DATA LOADING & DISPLAY
print("Original Dataset:")
print(df)
# STORE DATA IN TEXT FORMATS
# Save as CSV
df.to_csv("population_data.csv", index=False)
# Save as JSON
df.to_json("population_data.json", orient="records", lines=True)
# Save as TXT (tab-separated)
df.to_csv("population_data.txt", sep="\t", index=False)

# STEP 3: DATA WRANGLING
#Detect & Handle Missing Values
print("Missing Values:\n", df.isnull().sum())
# Fill missing population with mean
df['Population'].fillna(df['Population'].mean(), inplace=True)
# Drop rows with missing country
df.dropna(subset=['Country'], inplace=True)
# Remove Duplicates
# Add a duplicate row for demo
df = df._append(df.iloc[0], ignore_index=True)
df.drop_duplicates(inplace=True)
# Data Type Conversion
```

```
df['Year'] = df['Year'].astype(int)
df['Country'] = df['Country'].astype(str)
# Filtering & Aggregation
high_growth = df[df['GrowthRate'] > 1.0]
print("High Growth Countries:\n", high_growth)
# Group by Year and get total population
grouped = df.groupby("Year")["Population"].sum()
print("Total Population by Year:\n", grouped)
# STORE WRANGLED DATA
df.to_csv("cleaned_population_data.csv", index=False)
# Final Cleaned Data Preview
print("Cleaned Dataset:\n", df.head())
print("Cleaned Dataset:\n", df)
```

Output:



```
= RESTART: E:\DAV_ Prac1A.py
Original Dataset:
       Country    Population  Year  GrowthRate
0        India  1.400000e+09  2023         0.8
1        China  1.440000e+09  2023         0.3
2          USA  3.310000e+08  2023         0.5
3    Indonesia  2.730000e+08  2023         1.1
4     Pakistan  2.200000e+08  2023         2.0
5       Brazil  2.130000e+08  2023         0.7
6      Nigeria           NaN  2023         2.6
7   Bangladesh  1.660000e+08  2023         1.0
8       Russia  1.450000e+08  2023         0.0
9       Mexico  1.260000e+08  2023         1.2
10        None  1.200000e+08  2023         1.0
Missing Values:
 Country       1
Population    1
Year          0
GrowthRate    0
dtype: int64
```



```
High Growth Countries:
      Country   Population  Year  GrowthRate
3  Indonesia  273000000.0  2023         1.1
4   Pakistan  220000000.0  2023         2.0
6    Nigeria  443400000.0  2023         2.6
9     Mexico  126000000.0  2023         1.2
Total Population by Year:
 Year
2023    4.757400e+09
Name: Population, dtype: float64
Cleaned Dataset:
      Country    Population  Year  GrowthRate
0       India  1.400000e+09  2023         0.8
1       China  1.440000e+09  2023         0.3
2         USA  3.310000e+08  2023         0.5
3   Indonesia  2.730000e+08  2023         1.1
4    Pakistan  2.200000e+08  2023         2.0
Cleaned Dataset:
      Country    Population  Year  GrowthRate
0       India  1.400000e+09  2023         0.8
1       China  1.440000e+09  2023         0.3
2         USA  3.310000e+08  2023         0.5
3   Indonesia  2.730000e+08  2023         1.1
4    Pakistan  2.200000e+08  2023         2.0
5      Brazil  2.130000e+08  2023         0.7
6     Nigeria  4.434000e+08  2023         2.6
7  Bangladesh  1.660000e+08  2023         1.0
8      Russia  1.450000e+08  2023         0.0
9      Mexico  1.260000e+08  2023         1.2
```

Theory: Practical 1-B

File Formats
A file format is a standard way that information is encoded for storage in a computer file. It specifies how bits are used to encode information in a digital storage medium. File formats may be either proprietary or free.

Some file formats are designed for very particular types of data: PNG files, for example, store bitmapped images using lossless data compression. Other file formats, however, are designed for storage of several different types of data: the Ogg format can act as a container for different types of multimedia including any combination of audio and video, with or without text (such as subtitles), and metadata. A text file can contain any stream of characters, including possible control characters, and is encoded in one of various character encoding schemes. Some file formats, such as HTML, scalable vector graphics, and the source code of computer software are text files with defined syntaxes that allow them to be used for specific purposes.

Most Used file formats
Explore a wide range of common file formats and learn how to work with them effectively.

Text: This type of file contains only text without any formatting and can be opened with any text editor.
Different types of text formats include: .doc, .docx, .rtf, .pdf, .wpd

Image: This file type includes binary information about images and defines how the image will be stored and compressed.
Different types of Image File Format include: .JPEG, .PNG, .GIF, .HEIF

Audio: This type of file format stores audio data. It stores raw data in an encoded format and uses codec to perform compression and decompression.
Different types of Audio file formats include .aac, .mp3, .wav

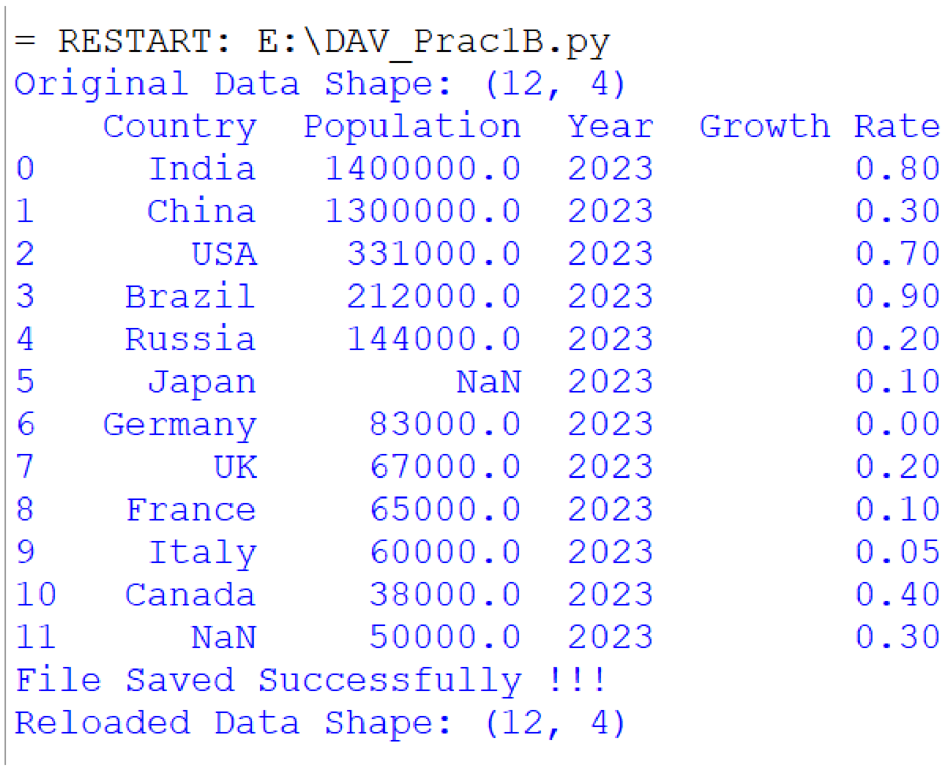
Video: This type of file format contains digital video data. It performs lossy compression to store video data where audio and video are separately encoded and stored.
Different types of Video File Formats include: .amv, .mpeg, .flv, .avi

Code:

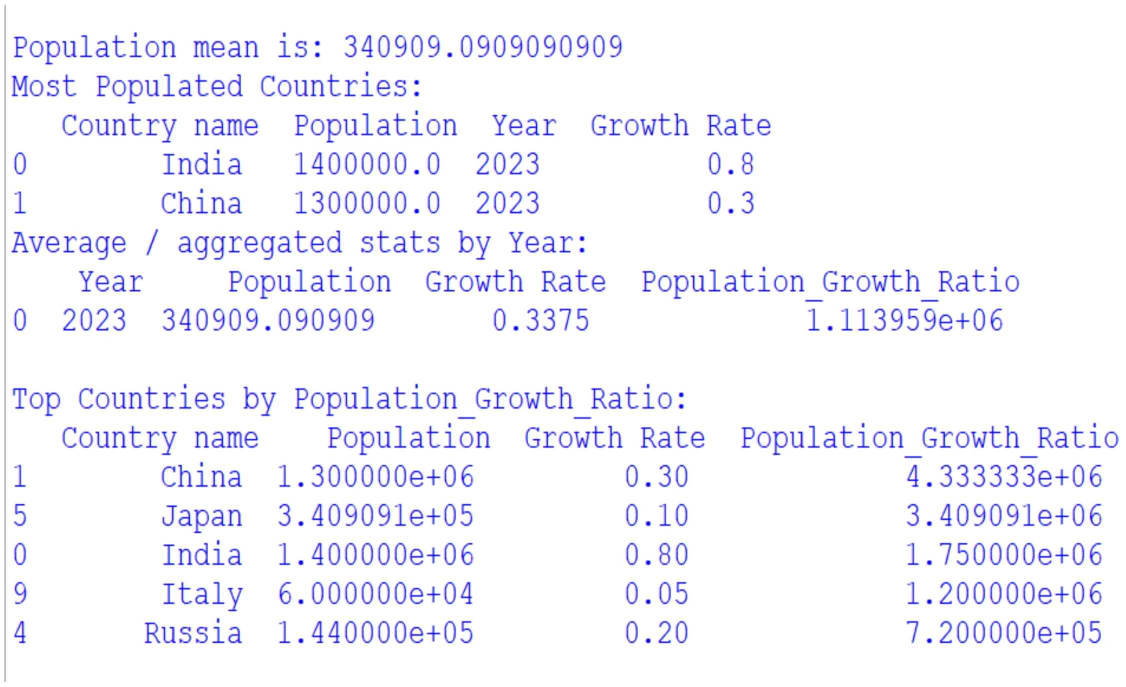
```
import pandas as pd

import numpy as np

# 1. Load dataset from CSV

df = pd.read_excel(r"C:\Users\Administrator\Documents\population_data.xlsx")

print("Original Data Shape:", df.shape)

print(df)

# 2. Save as a text format (pipe-separated file)

df.to_csv(r"C:\Users\User\population_data_1.txt", sep='|', index=False)

print('File Saved Successfully !!!')

# 3. Load back the data from the text format

text_data = pd.read_csv(r"C:\Users\User\population_data_1.txt", sep='|')

print("Reloaded Data Shape:", text_data.shape)

# 4. Data Wrangling

# a. Rename columns for simplicity

text_data.rename(columns={

   'Country': 'Country name',

   'GrowthRate': 'Growth Rate',

}, inplace=True)

# b. Handle missing values

text_data['Population'].fillna(text_data['Population'].mean(), inplace=True)

print("Population mean is:", text_data['Population'].mean())

# c. Filter: countries with population > population.mean()

most_populated_countries = text_data[text_data['Population'] >
text_data['Population'].mean()]
```

```
print('Most Populated Countries:\n', most_populated_countries)
# d. Create new column: Population to Growth Ratio
# (Handle division by zero or NaN to avoid 'inf')
text_data['Population_Growth_Ratio'] = np.where(
    (text_data['Growth Rate'] != 0) & (text_data['Growth Rate'].notna()),
    text_data['Population'] / text_data['Growth Rate'],
    0
)
# e. Group by Year (there is no Regional indicator in this dataset)
if 'Year' in text_data.columns:
    yearly_group = text_data.groupby('Year').agg({
        'Population': 'mean',
        'Growth Rate': 'mean',
        'Population_Growth_Ratio': 'mean'
    }).reset_index()
    print("Average / aggregated stats by Year:\n", yearly_group, "\n")
else:
    print("No 'Year' column to group by.\n")
# f. Sort countries by Population_Growth_Ratio
sorted_data = text_data.sort_values(by='Population_Growth_Ratio', ascending=False)
print("Top Countries by Population_Growth_Ratio:\n",
      sorted_data[['Country name', 'Population', 'Growth Rate',
'Population_Growth_Ratio']].head(), "\n")
```

Output:



```
= RESTART: E:\DAV_Prac1B.py
Original Data Shape: (12, 4)
    Country  Population  Year  Growth Rate
0     India   1400000.0  2023         0.80
1     China   1300000.0  2023         0.30
2       USA    331000.0  2023         0.70
3    Brazil    212000.0  2023         0.90
4    Russia    144000.0  2023         0.20
5     Japan         NaN  2023         0.10
6   Germany     83000.0  2023         0.00
7        UK     67000.0  2023         0.20
8    France     65000.0  2023         0.10
9     Italy     60000.0  2023         0.05
10   Canada     38000.0  2023         0.40
11      NaN     50000.0  2023         0.30
File Saved Successfully !!!
Reloaded Data Shape: (12, 4)
```



```
Population mean is: 340909.0909090909
Most Populated Countries:
  Country name  Population  Year  Growth Rate
0        India   1400000.0  2023          0.8
1        China   1300000.0  2023          0.3
Average / aggregated stats by Year:
   Year     Population  Growth Rate  Population_Growth_Ratio
0  2023  340909.090909       0.3375             1.113959e+06

Top Countries by Population_Growth_Ratio:
  Country name    Population  Growth Rate  Population_Growth_Ratio
1        China  1.300000e+06         0.30             4.333333e+06
5        Japan  3.409091e+05         0.10             3.409091e+06
0        India  1.400000e+06         0.80             1.750000e+06
9        Italy  6.000000e+04         0.05             1.200000e+06
4       Russia  1.440000e+05         0.20             7.200000e+05
```

Practical No: 02

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Implement code to interact with web APIs and perform web scraping.

Theory:

Web Scrapping
Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. Web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.

Web scraping is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications. There are many different ways to perform web scraping to obtain data from websites. These include using online services, particular API's or even creating your code for web scraping from scratch. Many large websites, like Google, Twitter, Facebook, StackOverflow, etc. have API's that allow you to access their data in a structured format. This is the best option, but there are other sites that don't allow users to access large amounts of data in a structured form or they are simply not that technologically advanced. In that situation, it's best to use Web Scraping to scrape the website for data.

Web scraping requires two parts, namely the crawler and the scraper. The crawler is an artificial intelligence algorithm that browses the web to search for the particular data required by following the links across the internet. The scraper, on the other hand, is a specific tool created to extract data from the website. The design of the scraper can vary greatly according to the complexity and scope of the project so that it can quickly and accurately extract the data.

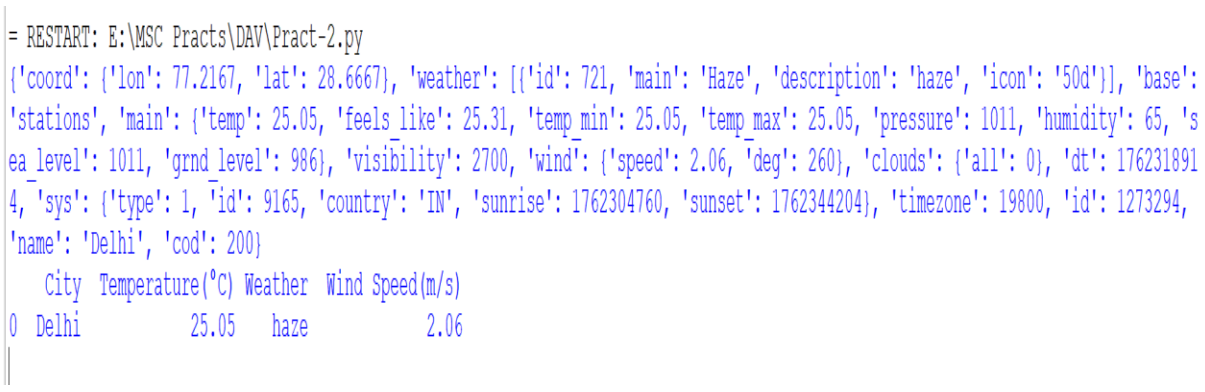
Web Scrapers can extract all the data on particular sites or the specific data that a user wants. Ideally, it's best if you specify the data, you want so that the web scraper only extracts that data quickly. For example, you might want to scrape an Amazon page for the types of juicers available, but you might only want the data about the models of different juicers and not the customer reviews.

Code:

```
import requests

import pandas as pd

API_KEY ="edd413efce672b15fdde269ac9cda1b5"

city = "Delhi"
```

```
url =
f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}&units=me
tric"

#Fetch data

response = requests.get(url)

data = response.json()

print(data)

#Extract info

weather_info = {

    "City": data["name"],

    "Temperature(°C)":data["main"]["temp"],

    "Weather": data["weather"][0]["description"],

    "Wind Speed(m/s)": data["wind"]["speed"]}

#Convert to DataFrame for analysis

df_weather = pd.DataFrame([weather_info])

print(df_weather)


#Perform Web Scrapping

import requests

from bs4 import BeautifulSoup

import pandas as pd

# Target website

urls =
["https://edition.cnn.com/","https://edition.cnn.com/science","https://edition.cnn.com/health",
"https://edition.cnn.com/business/tech"]

# Fetch HTML content

for url in urls:

    response = requests.get(url, headers={"User-Agent": "Mozilla/5.0"})

    soup = BeautifulSoup(response.text, "html.parser")

# Extract headlines

headlines = []

for item in soup.find_all("span", class_="container__headline-text"):
    text = item.get_text().strip()
    if text:   # ignore empty
        headlines.append(text)
# Convert to DataFrame
df_news = pd.DataFrame(headlines, columns=["Headline"])
# Show first 10
print(df_news.head(20))
```

Output:



```
= RESTART: E:\MSC Practs\DAV\Pract-2.py
{'coord': {'lon': 77.2167, 'lat': 28.6667}, 'weather': [{'id': 721, 'main': 'Haze', 'description': 'haze', 'icon': '50d'}], 'base':
'stations', 'main': {'temp': 25.05, 'feels_like': 25.31, 'temp_min': 25.05, 'temp_max': 25.05, 'pressure': 1011, 'humidity': 65, 's
ea_level': 1011, 'grnd_level': 986}, 'visibility': 2700, 'wind': {'speed': 2.06, 'deg': 260}, 'clouds': {'all': 0}, 'dt': 176231891
4, 'sys': {'type': 1, 'id': 9165, 'country': 'IN', 'sunrise': 1762304760, 'sunset': 1762344204}, 'timezone': 19800, 'id': 1273294,
'name': 'Delhi', 'cod': 200}
    City  Temperature(°C) Weather  Wind Speed(m/s)
0  Delhi            25.05    haze             2.06
```

```
                                             Headline
0   Starbucks once seemed unstoppable in China. It...
1   Ultra-fast fashion and 'childlike' sex dolls: ...
2   SNAP isn't just a moral imperative. It's good ...
3       No tennis coach? No problem thanks to AI boom
4            Want to get ahead? Talk to your (AI) CEO
5   Saudi Arabia is making a massive bet on becomi...
6   Big Tech keeps splurging on AI. The pressure i...
7   Three billionaires went out for chicken and be...
8   Amazon says it didn't cut 14,000 people becaus...
9   Apple just provided its first glimpse at how i...
10  The world's most valuable company just blew th...
11  After a wave of lawsuits, Character.AI will no...
12  Nvidia CEO highlights his close relationship w...
13  The world's most valuable public company just ...
14        Elon Musk launches his version of Wikipedia
15  How a tiny bug spiraled into a massive outage ...
16  Microsoft AI CEO: We're making an AI that you ...
17  What is the heliosphere? A new mission could u...
18  Black hole collision confirms decades-old pred...
19  Rock discovery contains 'clearest sign' yet of...
```

Practical No: 03

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Demonstrate data cleaning, handle missing data, and perform string manipulation.

Theory:

Data Cleaning

Data cleansing or data cleaning is the process of identifying and correcting (or removing) corrupt, inaccurate, or irrelevant records from a dataset, table, or database. It involves detecting incomplete, incorrect, or inaccurate parts of the data and then replacing, modifying, or deleting the affected data.[1] Data cleansing can be performed interactively using data wrangling tools, or through batch processing often via scripts or a data quality firewall.

After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores. Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

Data Preparation

Data preparation is the act of manipulating (or pre-processing) raw data (which may come from disparate data sources) into a form that can readily and accurately be analysed, e.g. for business purposes.

Data preparation is the first step in data analytics projects and can include many discrete tasks such as loading data or data ingestion, data fusion, data cleaning, data augmentation, and data delivery.

Data preparation is the process of making raw data ready for after processing and analysis. The key methods are to collect, clean, and label raw data in a format suitable for machine learning (ML) algorithms, followed by data exploration and visualization. The process of cleaning and combining raw data before using it for machine learning and business analysis is known as data preparation, or sometimes "pre-processing." But it may not be the most attractive of duties, careful data preparation is essential to the success of data analytics. Clear and important ideas from raw data require careful validation, cleaning, and an addition. Any business analysis or model created will only be as strong and validating as the very first information preparation.

Code:

```
In [1]: import pandas as pd
        import numpy as np

        data = {
            "Name":["Alice","Bob","Charlie",None,"David","Eve"],
            "Age":[25,np.nan,30,22,None,28],
            "City":["Newyork","London","Paris","Delhi",None,"Mumbai"],
            "salary":["50000","60000","unknown","45000","55000",""]
        }
        df = pd.DataFrame(data)
        print("original DataFrame:\n",df)

        original DataFrame:
               Name   Age     City   salary
        0    Alice  25.0  Newyork    50000
        1      Bob   NaN   London    60000
        2  Charlie  30.0    Paris  unknown
        3     None  22.0    Delhi    45000
        4    David   NaN     None    55000
        5      Eve  28.0   Mumbai

In [2]: type(df)

Out[2]: pandas.core.frame.DataFrame

In [3]: data1 = pd.Series ([ "Erica","Tanya","tejaswi" ,None ,"siya",np.nan])
        data1

Out[3]: 0      Erica
        1      Tanya
        2    tejaswi
        3       None
        4       siya
        5        NaN
        dtype: object

In [4]: data1.isnull

Out[4]: <bound method Series.isnull of 0      Erica
        1      Tanya
        2    tejaswi
        3       None
        4       siya
        5        NaN
        dtype: object>

In [5]: data1.isnull()

Out[5]: 0    False
        1    False
        2    False
        3     True
        4    False
        5     True
        dtype: bool

In [6]: data1
```

```
Out[6]: 0      Erica
        1      Tanya
        2    tejaswi
        3       None
        4       siya
        5        NaN
        dtype: object
```

```
In [7]: data1 [4] = None
        data1
```

```
Out[7]: 0      Erica
        1      Tanya
        2    tejaswi
        3       None
        4       None
        5        NaN
        dtype: object
```

```
In [8]: data1.dropna()
```

```
Out[8]: 0      Erica
        1      Tanya
        2    tejaswi
        dtype: object
```

```
In [9]: data1 [0] = "Sadiya"
        data1
```

```
Out[9]: 0     Sadiya
        1      Tanya
        2    tejaswi
        3       None
        4       None
        5        NaN
        dtype: object
```

```
In [10]: data1.notnull()
```

```
Out[10]: 0     True
         1     True
         2     True
         3    False
         4    False
         5    False
         dtype: bool
```

string manipulation

```
In [11]: name="Tara  ,  Sara  ,  Zara"
         name
```

```
Out[11]: 'Tara  ,  Sara  ,  Zara'
```

```
In [12]: name.split()
```

```
Out[12]: ['Tara', ',', 'Sara', ',', 'Zara']
```

```
In [13]: name.strip()

Out[13]: 'Tara  ,  Sara  ,  Zara'

In [14]: t=name.strip().split(",")
         t

Out[14]: ['Tara  ', '  Sara  ', '  Zara']

In [15]: p=["hii","msc","data","science"]
         first, second, third ,fourth=p

In [16]: p

Out[16]: ['hii', 'msc', 'data', 'science']

In [17]: first

Out[17]: 'hii'

In [18]: second

Out[18]: 'msc'

In [19]: third

Out[19]: 'data'

In [20]: fourth

Out[20]: 'science'

In [21]: '::' .join(p)

Out[21]: 'hii::msc::data::science'

In [22]: name.replace('  ','::')

Out[22]: 'Tara::,::Sara::,::Zara'

In [23]: name1="siya        jiya\t  riya\t     priya  piya"

In [24]: name1

Out[24]: 'siya        jiya\t  riya\t     priya  piya'

In [25]: import regex as re

In [26]: re.split('\s',name1)
```

```
Out[26]: ['siya',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          'jiya',
          '',
          '',
          'riya',
          '',
          '',
          '',
          '',
          '',
          'priya',
          '',
          'piya']
```

```
In [27]: mydata=" siya siyarajput@gmail.com  nyasa  nyasarao12@gamial.com  kavya  kavyash
         mydata
```

```
Out[27]: ' siya siyarajput@gmail.com  nyasa  nyasarao12@gamial.com  kavya  kavyasharma@g
         mail.com'
```

```
In [28]: pattern = r'[a-z0-9_.+-]+@[a-z0-9-]+\.[a-z0-9-.]'
```

```
In [29]: data2 = re.compile(pattern ,re.IGNORECASE)
```

```
In [30]: data2
```

```
Out[30]: regex.Regex('[a-z0-9_.+-]+@[a-z0-9-]+\\.[a-z0-9-.]', flags=regex.I | regex.V0)
```

```
In [31]: data2.findall(mydata)
```

```
Out[31]: ['siyarajput@gmail.c', 'nyasarao12@gamial.c', 'kavyasharma@gmail.c']
```

```
In [32]: string=" hello this is my program "
```

```
In [33]: result = string.lower()
         print(result)
```

```
 hello this is my program
```

```
In [34]: result = string.upper()
         print(result)
```

```
 HELLO THIS IS MY PROGRAM
```

```
In [35]: result = string.isalnum()
         result
```

```
Out[35]: False
```

```
In [36]: result = string.capitalize()
         result
```

```
Out[36]:  ' hello this is my program '
```

```
In [37]:  result = string.casefold()
           result
```

```
Out[37]:  ' hello this is my program '
```

```
In [38]:  result = string.encode()
           result
```

```
Out[38]:  b' hello this is my program '
```

```
In [39]:  result = string.count("hello")
           result
```

```
Out[39]:  1
```

```
In [40]:  result = string.count("l")
           result
```

```
Out[40]:  2
```

```
In [41]:  result = string.center(55)
           result
```

```
Out[41]:  '                 hello this is my program                '
```

```
In [42]:  result = string.center(111)
           result
```

```
Out[42]:  '                                             hello this is my program
          '
```

```
In [43]:  result = string.count("i")
           result
```

```
Out[43]:  2
```

```
In [44]:  result = string.count("a")
           result
```

```
Out[44]:  1
```

```
In [45]:  result = string.endswith("m")
           result
```

```
Out[45]:  False
```

```
In [46]:  result = string.endswith(" ")
           result
```

```
Out[46]:  True
```

```
In [47]:  result = string.expandtabs(5)
           result
```

```
Out[47]:  ' hello this is my program '
```

Practical No: 04

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Using R, execute statistical functions such as mean, median, mode, quartiles, range, interquartile range, and create a histogram.

Theory:

Statistical functions play a crucial role in data analysis as they help summarize, interpret, and draw conclusions from raw data. These functions provide insights into both the central tendency and dispersion of a dataset.

The mean represents the average value of the data and gives a general idea of its central point. The median is the middle value that divides the dataset into two equal halves, making it useful when data contains extreme values or outliers. The mode indicates the most frequently occurring value in the dataset and is helpful for understanding common or dominant characteristics.

Measures of dispersion such as the range, quartiles, and interquartile range (IQR) describe how spread out the data values are. The range is the difference between the maximum and minimum values, giving a simple measure of variability. Quartiles divide the data into four equal parts, while the interquartile range (IQR) the difference between the third quartile (Q3) and the first quartile (Q1) shows the spread of the middle 50% of the data, minimizing the effect of outliers.

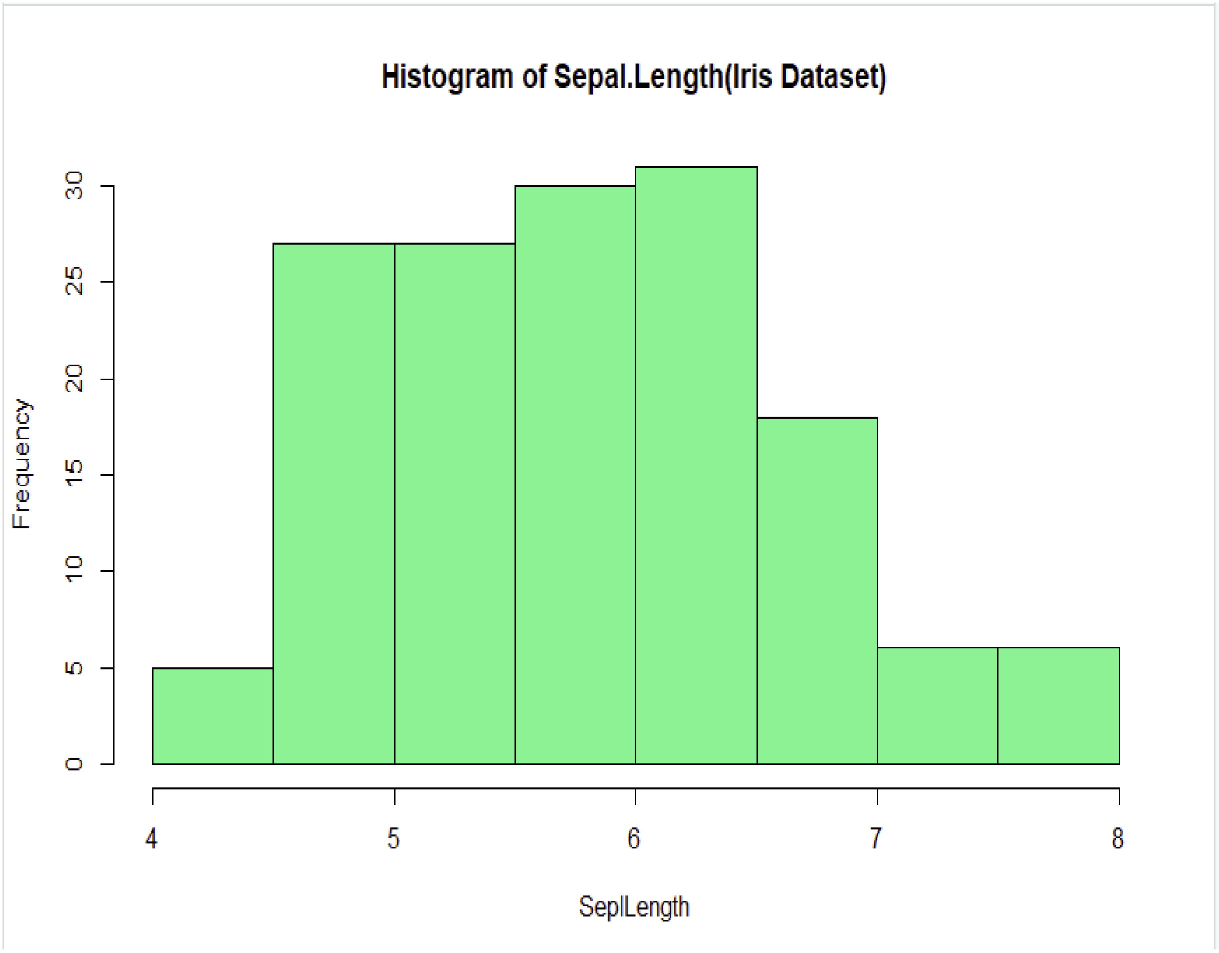
A histogram is a graphical representation of data distribution. It divides the data into intervals (bins) and displays the frequency of values within each interval using bars. This visualization helps in identifying the shape, skewness, and spread of the data.

By using R programming, these statistical functions and visualizations can be executed efficiently. R provides built-in commands and functions to compute measures of central tendency and dispersion, and to generate histograms. This allows researchers and analysts to perform comprehensive statistical analysis and gain meaningful insights from data in a systematic way.

Code:

```
> #Sample Data
> data<-c(39,45,92,56,34,56,39,25,39,25,39,95,36,45,78,86,92,68)
>
> #Function to calculate Mode
> get_mode<-function(x){
+   unique_x<-unique(x)
+   unique_x[which.max(tabulate(match(x, unique_x)))]
+ }
>
> #Statistical Calculations
> mean_value<-mean(data)
> median_value<-median(data)
> mode_value<-get_mode(data)
> quartiles_value<-quantile(data)
> range_value<-range(data)
> iqr_value<-IQR(data)
> cat("Mean:",mean_value,"\n")
Mean: 54.94444
> cat("Mode:",mode_value,"\n")
Mode: 39
> cat("Quartiles:",quartiles_value,"\n")
Quartiles: 25 39 45 75.5 95
> cat("Range:",range_value,"\n")
Range: 25 95
> cat("Interquartile Range(IQR):",iqr_value,"\n")
Interquartile Range(IQR): 36.5
>
> #Create a Histogram
> hist(data,
+      main="Histogram of Data",
+      xlab="Values",
+      ylab="Frequency",
+      col="blue",
+      border="black")
>
>
> #R with in built datasets IRIS
> data(iris)
> x<-iris$Sepal.Length
>
> #Mean
> mean_val<-mean(x)
> cat("Mean of Sepal.Length:",mean_val,"\n")
Mean of Sepal.Length: 5.843333
>
> #Median
> median_val<-median(x)
> cat("Median of Sepal.Length:",median_val,"\n")
Median of Sepal.Length: 5.8
>
> #Mode
> get_mode<-function(v){
+   uniq_vals<-unique(v)
+   uniq_vals[which.max(tabulate(match(v,uniq_vals)))]
+ }
> mode_val<-get_mode(x)
> cat("Mode of Sepal.Length:",mode_val,"\n")
Mode of Sepal.Length: 5
>
> #Quartiles
> quartiles<-quantile(x)
> cat("Quartiles of Sepal.Length:\n")
Quartiles of Sepal.Length:
> print(quartiles)
  0%  25%  50%  75% 100% 
 4.3  5.1  5.8  6.4  7.9 
>
> #Range
> range_vals<-range(x)
> range_diff<-diff(range(x))
> cat("Range(min to max):",range_vals,"\n")
```

```
Range(min to max): 4.3 7.9
> cat("Range(difference):",range_diff,"\n")
Range(difference): 3.6
>
> #Interquartile Range
> iqr_val<-IQR(x)
> cat("Interquartile Range(IQR):",iqr_val,"\n")
Interquartile Range(IQR): 1.3
>
> #Histogram
> hist(x,
+      main="Histogram of Sepal.Length(Iris Dataset)",
+      xlab="SeplLength",
+      ylab="Frequency",
+      col="lightgreen",
+      border="black")
```



Practical No: 05

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Write an R program to manage data and perform operations using the List data structure and Data Frames.

Theory:

In R programming, data management involves storing, organizing, and manipulating data efficiently to perform meaningful analysis. Two of the most important data structures used for this purpose are lists and data frames.

A list in R is a versatile data structure that can hold elements of different types such as numbers, strings, vectors, matrices, or even other lists. This makes lists highly flexible for managing complex datasets or grouping related information together. Operations on lists may include creating, accessing, modifying, or merging elements, which allows users to organize and process heterogeneous data conveniently.

A data frame, on the other hand, is a tabular data structure similar to a spreadsheet or SQL table, where data is organized in rows and columns. Each column can contain values of different data types, making it ideal for storing structured datasets such as survey results, experimental data, or CSV file contents. Operations on data frames include adding or removing rows and columns, filtering data, sorting, merging, and performing statistical summaries.

By writing an R program that utilizes lists and data frames, one can efficiently manage and manipulate data in various forms. These operations form the foundation for data analysis, allowing users to clean, organize, and prepare data for further statistical or graphical interpretation. Thus, understanding how to work with lists and data frames is essential for effective data handling and analysis in R programming.

Code:

```
> student_list<-list(
+   Names = c("Alice" ,"Lily" ,"Alex"),
+   Ages = c(20, 21, 23),
+   Marks = c(78, 88, 80)
+ )
> cat("Original Student Names:\n")
Original Student Names:
> print(student_list)

$Names
[1] "Alice" "Lily"  "Alex"
$Ages
[1] 20 21 23

$Marks
[1] 78 88 80
```

```
>
> cat("\n Student Names:\n")

 Student Names:
> print(student_list$Names)
[1] "Alice" "Lily"  "Alex"
>
> cat("\n Student Ages:\n")

 Student Ages:
> print(student_list$Ages)
[1] 20 21 23
>
> cat("\n Student Marks:\n")

 Student Marks:
> print(student_list$Marks)
[1] 78 88 80
>
> #Modifying an element in the list
> student_list$Marks[2]<-92 #Update Lily's Marks
> cat("\n Modified Marks in the list:\n")

 Modified Marks in the list:
> print(student_list$Marks)
[1] 78 92 80
>
> #Converting List to Data Frame
> student_df<-data.frame(
+   Names = student_list$Names,
+   Age = student_list$Age,
+   Marks = student_list$Marks
+ )
> cat("\n student Data Frame: \n")

 student Data Frame:
> print(student_df)
  Names Age Marks
1 Alice  20    78
2  Lily  21    92
3  Alex  23    80
>
> new_data<-data.frame(Names = "David", Age = 22 , Marks = 60)
> student_df<-rbind(student_df,new_data)
>
> #Display the Updated Data Frame
> cat("\n Updated student Data Frame(after adding David):\n")

 Updated student Data Frame(after adding David):
> print(student_df)
  Names Age Marks
1 Alice  20    78
2  Lily  21    92
3  Alex  23    80
4 David  22    60
>
> #Subsetting Data Frame:Students with marks greater than 88)
> high_Marks<-subset(student_df, Marks>88)
>
>
> cat("\n student with Marks greater than 88: \n")

 student with Marks greater than 88:
> print(high_Marks)
  Names Age Marks
2  Lily  21    92
>
> avg_Marks<-mean(student_df$Marks)
> cat("\n Average Marks of student:", avg_Marks, "\n")

 Average Marks of student: 77.5
> cat("\n Summary statistics of the data frame:\n")

 Summary statistics of the data frame:
> print(summary (student_df))
    Names                Age            Marks
 Length:4           Min.   :20.00   Min.   :60.0
 Class :character   1st Qu.:20.75   1st Qu.:73.5
 Mode  :character   Median :21.50   Median :79.0
                    Mean   :21.50   Mean   :77.5
                    3rd Qu.:22.25   3rd Qu.:83.0
                    Max.   :23.00   Max.   :92.0
```

Practical No: 06

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Write an R program to create, analyze, visualize, and write data to CSV files

Theory:

Data interfacing in R refers to the process of importing, manipulating, analyzing, and exporting data between R and external file formats such as CSV (Comma-Separated Values) files. CSV files are one of the most commonly used formats for storing tabular data because they are simple, portable, and compatible with various software applications such as Excel, Python, and databases.

In R programming, data interfacing with CSV files typically involves four main steps creating, reading, analyzing/visualizing, and writing data. Using functions like read.csv() or read.table(), data can be easily imported from a CSV file into R as a data frame for further analysis. Once imported, users can perform operations such as data cleaning, filtering, summarizing, and applying statistical functions to derive insights.

For visualization, R provides powerful libraries such as ggplot2, plotly, and base plotting functions to represent data graphically using charts and plots like histograms, bar graphs, scatter plots, or line charts. Visual representation helps in identifying trends, relationships, and patterns within the data more effectively.

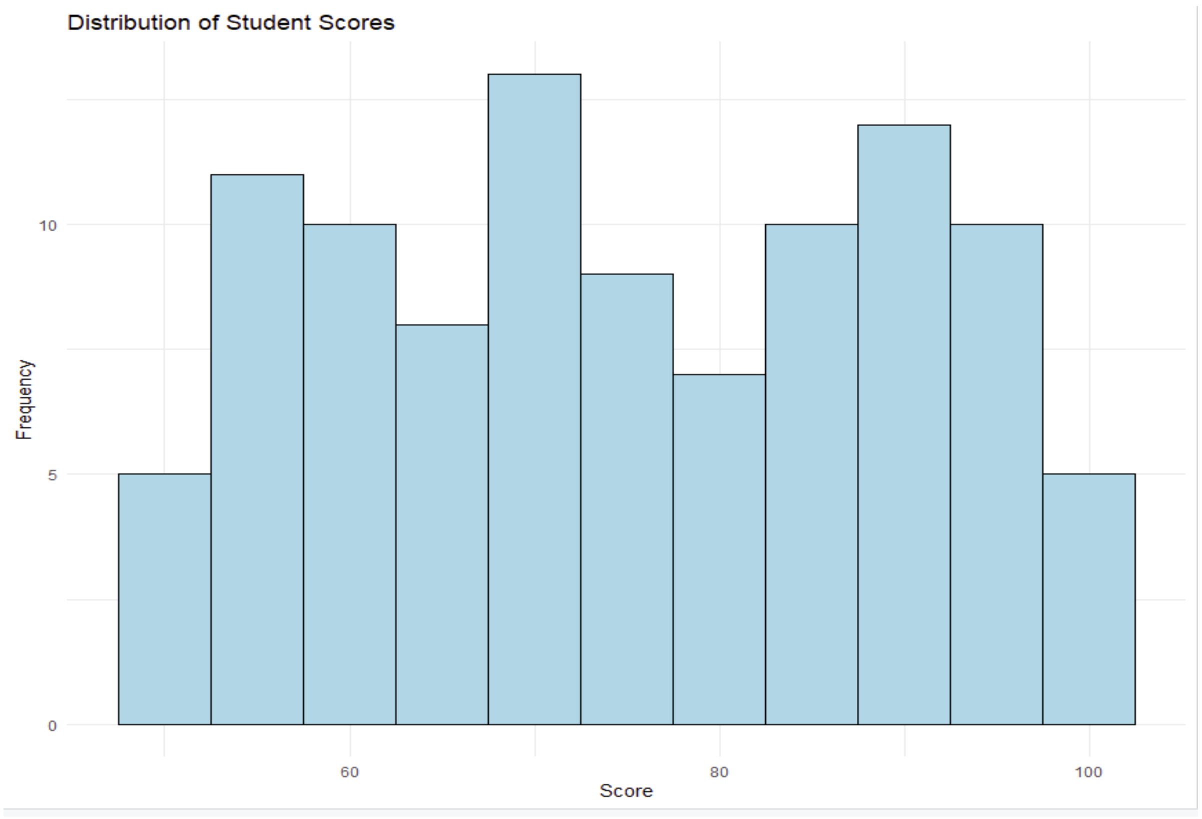
After processing and visualizing, the modified or analyzed data can be exported back to a CSV file using the write.csv() function. This allows the results to be shared or used in other tools for reporting or further analysis.

Thus, through this practical, learn how to interface R with external data sources, perform data analysis and visualization, and manage data flow efficiently between R and CSV files a crucial skill in real world data analytics and research applications.

Code:

```
> # Step 1: Create data
> set.seed(123)
> num_students <- 100
> student_data <- data.frame(
+   Name = paste("Student", 1:num_students),
+   Score = round(runif(num_students, 50, 100)),
+   Age = sample(18:25, num_students, replace = TRUE)
+ )
> cat("Sample Student Data:\n")
Sample Student Data:
> print(head(student_data))
       Name Score Age
1 Student 1    64  18
2 Student 2    89  21
3 Student 3    70  23
4 Student 4    94  18
```

```
5 Student 5    97  25
6 Student 6    52  23
>
> # Step 2: Analyze data
> cat("\nSummary of Scores:\n")
Summary of Scores:
> print(summary(student_data$Score))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  50.00   62.00   73.00   74.95   88.00  100.00
> mean_score <- mean(student_data$Score)
> sd_score <- sd(student_data$Score)
> cat("\nMean Score:", mean_score, "\n")
Mean Score: 74.95
> cat("Standard Deviation of Scores:", sd_score, "\n")
Standard Deviation of Scores: 14.29585
> # Step 3: Visualize data
> library(ggplot2)
> ggplot(student_data, aes(x = Score)) +
+   geom_histogram(binwidth = 5, fill = "lightblue", color = "black") +
+   labs(title = "Distribution of Student Scores",
+        x = "Score",
+        y = "Frequency") +
+   theme_minimal()
```



Practical No: 07

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Create common charts in Tableau with appropriate titles, labels, and descriptions

Theory:

Bar Chart

A bar chart, also known as a bar graph, displays categorical data using rectangular bars whose length or height is proportional to the values they represent. The bars can be oriented either vertically or horizontally. Bar charts are ideal for comparing quantities across different categories, identifying trends, and showing relative differences between groups.

Pie Chart

A pie chart is a circular statistical graphic divided into slices to illustrate numerical proportions. Each slice represents a category's contribution to the total, with its angle and area proportional to the corresponding value. Pie charts are best used to show percentage or part-to-whole relationships in a dataset, making it easy to compare the relative size of categories at a glance.

Table

A table is a structured arrangement of data organized in rows and columns, allowing easy comparison, sorting, and reference of detailed values. Tables are ideal for presenting exact numerical information, summarizing multiple variables, and providing clarity when precision and detailed data presentation are required.

Bubble Chart

A bubble chart is a graphical representation that displays three dimensions of data. Each point on the chart is represented by a bubble, where the X and Y positions show two variables, and the bubble's size (and sometimes color) represents a third variable. Bubble charts are useful for identifying patterns, relationships, and clusters among data points, especially in complex datasets.

Histogram

A histogram is a type of bar graph that represents the frequency distribution of continuous data. Each bar corresponds to a range (or bin) of values, showing how many data points fall within that interval. Histograms are helpful for understanding the shape, spread, and central tendency of data, revealing patterns such as skewness, gaps, or outliers.

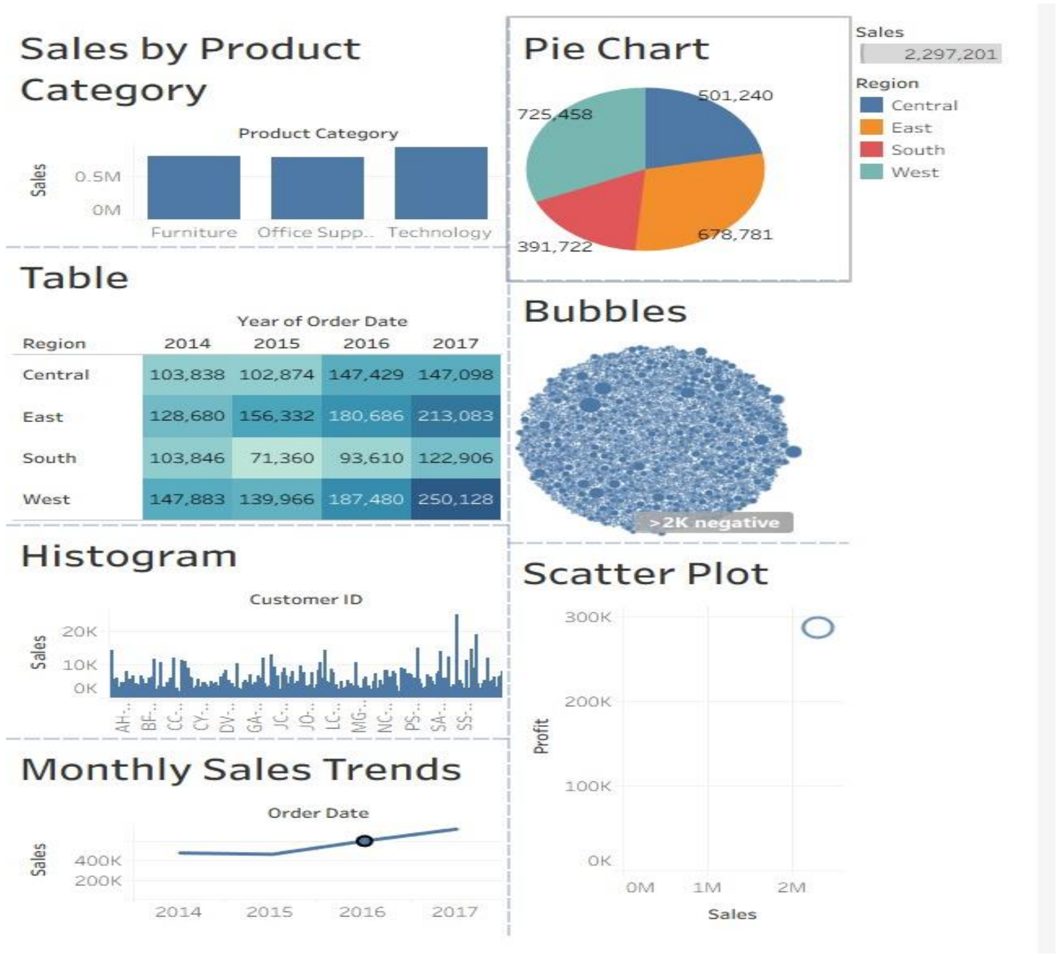
Scatter Plot

A scatter plot displays data points on a two-dimensional graph, showing the relationship between two numerical variables. Each point represents an observation, positioned according to its X and Y values. Scatter plots are commonly used to detect correlations, clusters, or trends between variables, and to identify outliers in the dataset.

Line Chart (Monthly Sales Trends)

A line chart visualizes data trends over time by connecting individual data points with straight lines. It effectively shows changes, patterns, and fluctuations in data, such as growth or decline over a specific period. Line charts are especially useful for time series analysis, helping identify overall trends and seasonal variations in performance or behavior.

Output:



Table

	Year of Order Date			
Region	2014	2015	2016	2017
Central	103,838	102,874	147,429	147,098
East	128,680	156,332	180,686	213,083
South	103,846	71,360	93,610	122,906
West	147,883	139,966	187,480	250,128

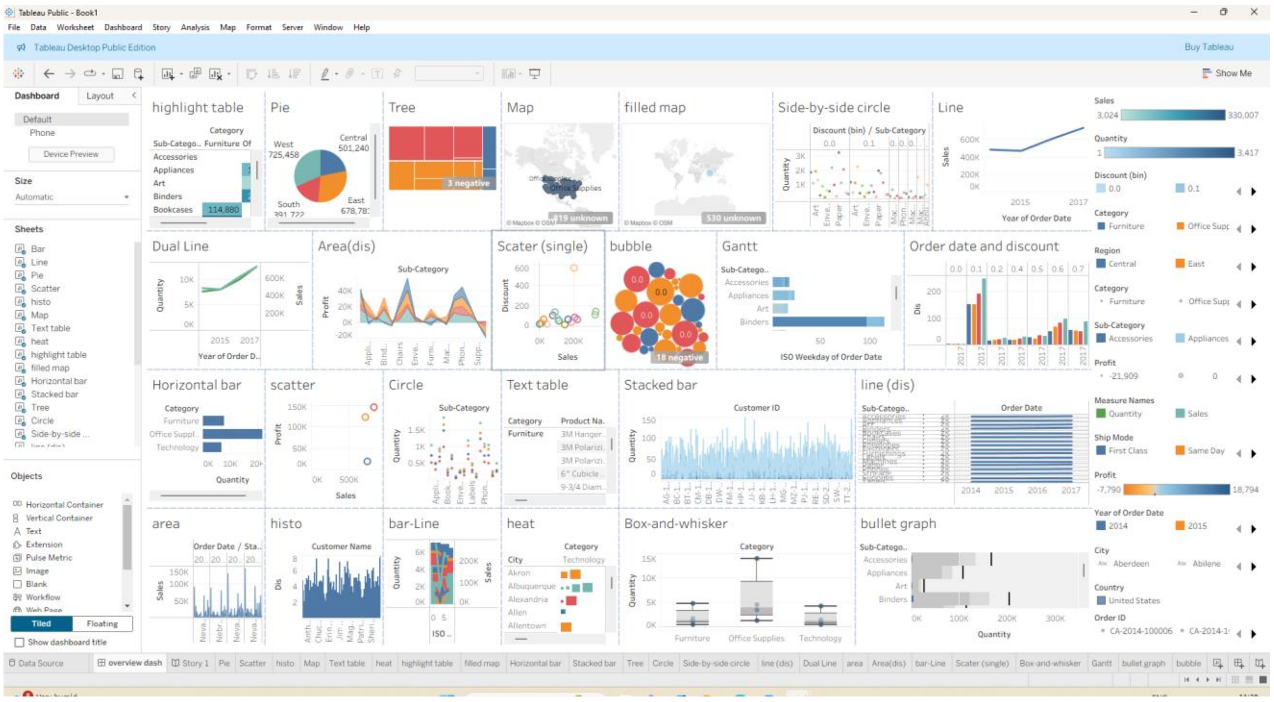


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Show Me
Dashboard
Layout
Default
Phone
Device Preview
Size
Automatic
Sheets
Bar
Line
Pie
Scatter
histo
Map
Text table
heat
highlight table
filled map
Horizontal bar
Stacked bar
Tree
Circle
Side-by-side ...
Objects
Horizontal Container
Vertical Container
Text
Extension
Pulse Metric
Image
Blank
Workflow
Tiled
Floating
Show dashboard title
highlight table
Pie
Tree
Map
filled map
Side-by-side circle
Line
Dual Line
Area(dis)
Scater (single)
bubble
Gantt
Order date and discount
Horizontal bar
scatter
Circle
Text table
Stacked bar
line (dis)
area
histo
bar-Line
heat
Box-and-whisker
bullet graph
Sales
Quantity
Discount (bin)
Category
Region
Sub-Category
Profit
Measure Names
Ship Mode
Year of Order Date
City
Country
Order ID
Data Source
overview dash
Story 1

Practical No: 08

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Perform sorting and filtering in Tableau, create visualizations, and publish them on Tableau Cloud.

Theory:

Data Filtering

Data filtering is the process of narrowing down the most relevant information from a large dataset using specific conditions or criteria. It makes the analysis more focused and efficient.

Data filtering lets you quickly analyze relevant data without sifting through the entire dataset. You can filter data regardless of type, including numbers, categories, text, and complex time-series data.

Benefits of using data filtering

Organizations prioritizing data filtering are better positioned to derive valuable insights from their data. Here is how data filtering can help you gain a competitive advantage.

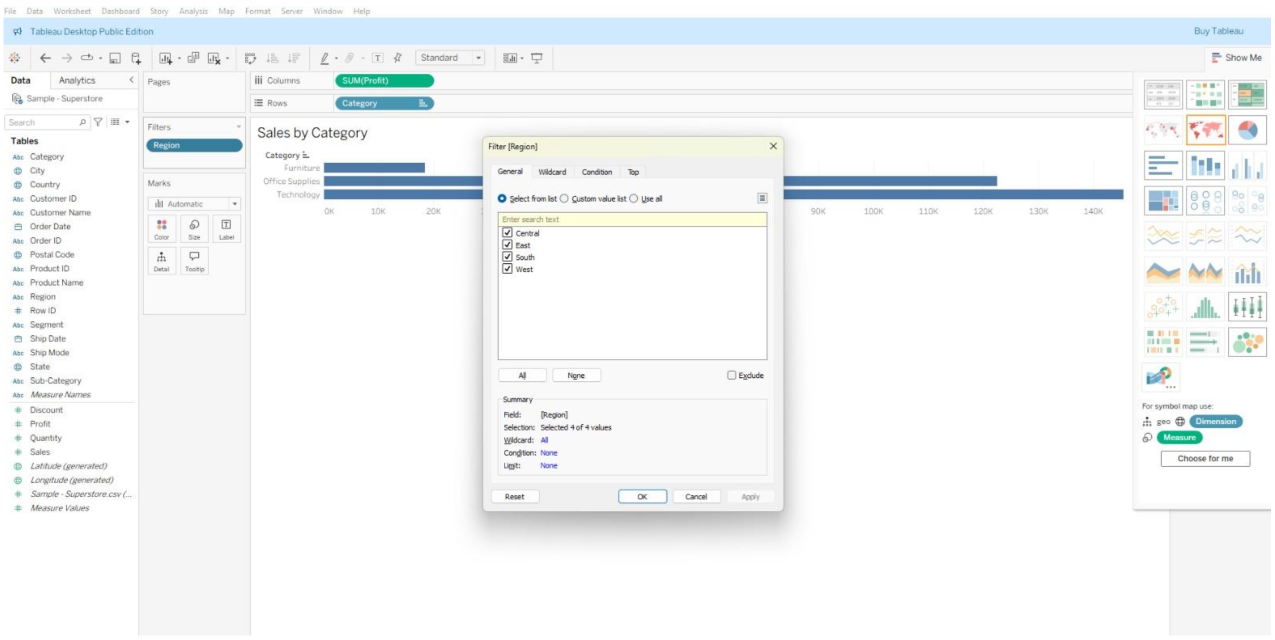
- **Enhances Focus**: Data filtering allows you to ignore irrelevant data, enabling a sharper focus on information that aligns with their goals, which can improve the quality of insights.
- **Increases Accuracy**: Filtering out outliers and erroneous records contributes to a more reliable data analysis process and improves the accuracy of the results.
- **Optimizes Resource Use**: Working with smaller, filtered datasets can reduce the resources needed for analysis, leading to potential cost savings.
- **Supports Custom Analysis**: Data filtering accommodates unique analytical needs across various projects or departments by creating datasets tailored to specific criteria.

What is Data Filtering used for

Data filtering plays an instrumental role in reducing computational time and enhancing the accuracy of AI models. Given the increasing need for organizations to manage large volumes of data, leveraging data filtering has become indispensable.

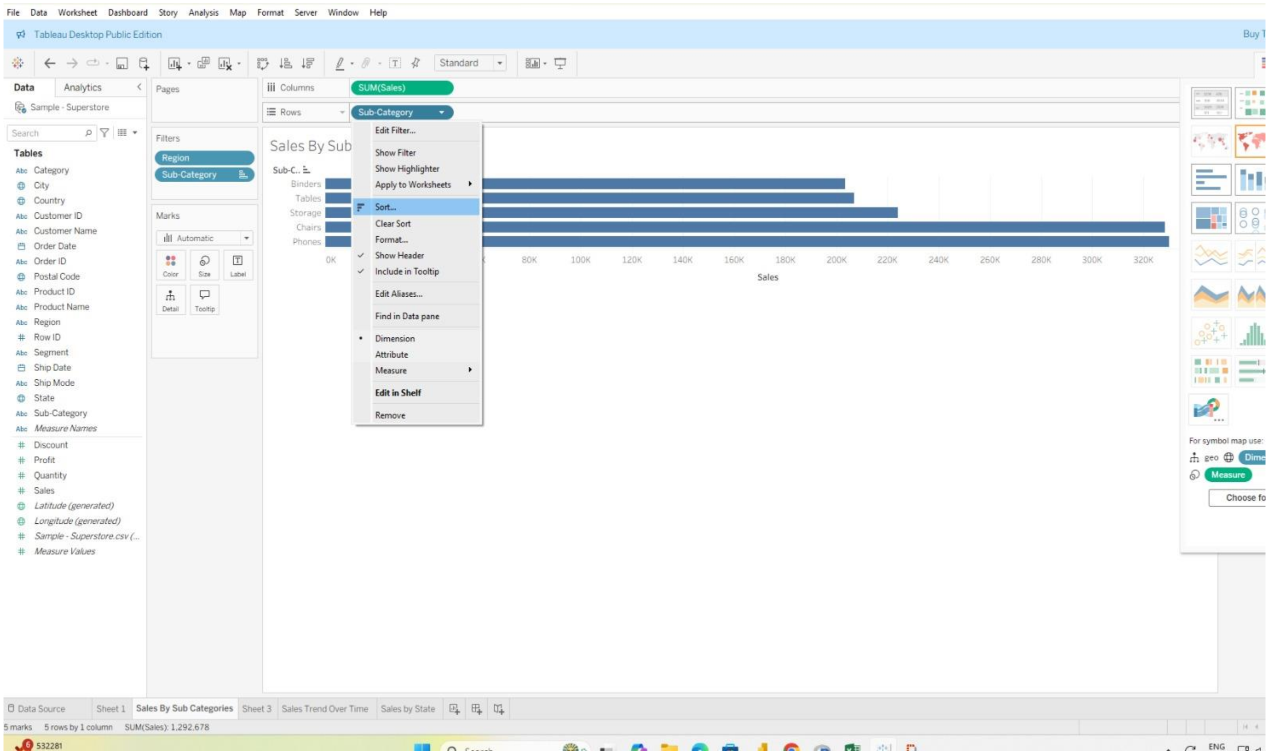
- **Evaluating a Dataset:** Filtering aids in exploratory data analysis by helping identify patterns, trends, or anomalies within a dataset.
- **Processing Records:** Data filtering streamlines workflows by processing records based on predefined criteria.
- **Remove Irrelevant Data:** Filtered data can help remove irrelevant data before restructuring via pivoting, grouping/aggregating, or other means.

Output:



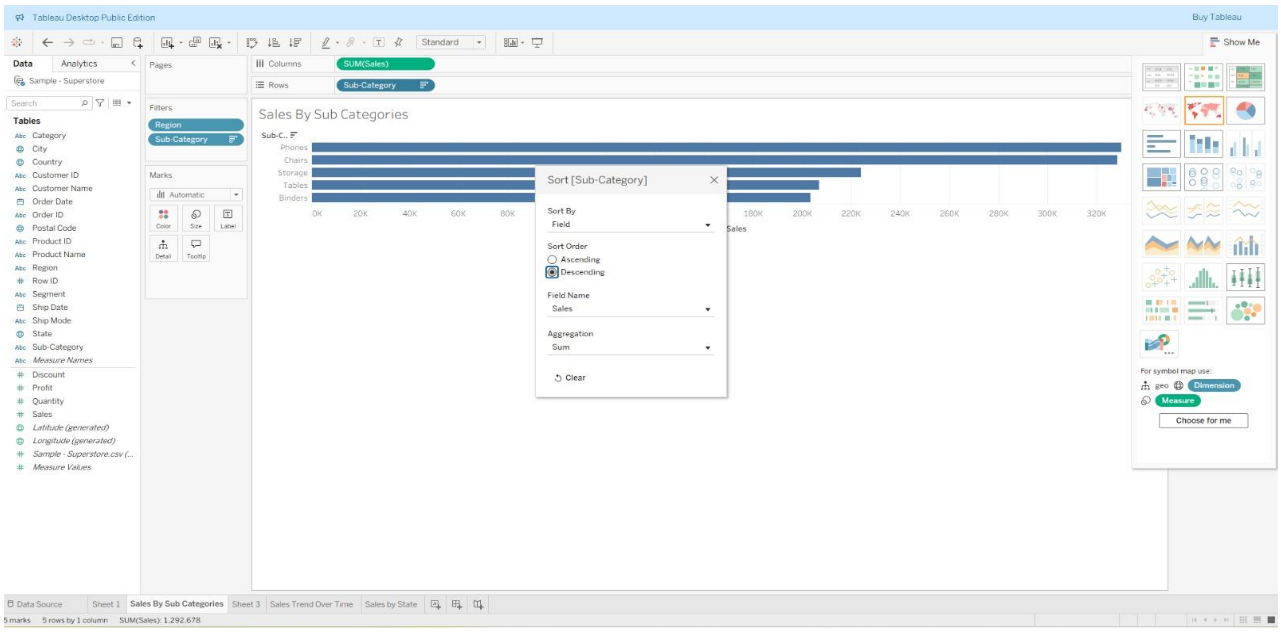
Columns: SUM(Profit)
Rows: Category
Filters: Region

Summary
Field: [Region]
Selection: Selected 4 of 4 values
Wildcard: All
Condition: None
Limit: None



Columns: SUM(Sales)
Rows: Sub-Category
Filters: Region, Sub-Category

5 marks 5 rows by 1 column SUM(Sales): 1,292,678



Sales By Sub Categories

Sort [Sub-Category]

Sort By: Field

Sort Order: Ascending [✓ Descending]

Field Name: Sales

Aggregation: Sum

Clear

5 marks 5 rows by 1 column SUM(Sales): 1,292,678

Practical No: 09

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Perform data visualization and create reports using Power BI.

Theory:

Data Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

Data visualization translates complex data sets into visual formats that are easier for the human brain to comprehend. This can include a variety of visual tools such as:

- **Charts**: Bar charts, line charts, pie charts, etc.
- **Graphs**: Scatter plots, histograms, etc.
- **Maps**: Geographic maps, heat maps, etc.
- **Dashboards**: Interactive platforms that combine multiple visualizations.

The primary goal of data visualization is to make data more accessible and easier to interpret, allowing users to identify patterns, trends, and outliers quickly. This is particularly important in the context of big data, where the sheer volume of information can be overwhelming without effective visualization techniques.

Data visualization is commonly used to spur idea generation across teams. They are frequently leveraged during brainstorming or Design Thinking sessions at the start of a project by supporting the collection of different perspectives and highlighting the common concerns of the collective. While these visualizations are usually unpolished and unrefined, they help set the foundation within the project to ensure that the team is aligned on the problem that they're looking to address for key stakeholders.

Data visualization for idea illustration assists in conveying an idea, such as a tactic or process. It is commonly used in learning settings, such as tutorials, certification courses, centers of excellence, but it can also be used to represent organization structures or processes, facilitating communication between the right individuals for specific tasks. Project managers frequently use Gantt charts and waterfall charts to illustrate workflows. Data modeling also uses abstraction to represent and better understand data flow within an enterprise's information system, making it easier for developers, business analysts, data architects, and others to understand the relationships in a database or data warehouse.

Data Reporting

Data reporting is a structured process that involves the transformation of raw data into understandable and actionable information. At its core, it is about presenting data in a comprehensible format, allowing individuals, businesses, and organizations to draw insights, monitor performance, and make informed decisions.

This systematic approach to reporting goes beyond the mere presentation of data; it aims to convey a narrative, highlighting trends, patterns, and key metrics within the information. As a result, data reporting stands as a cornerstone in the landscape of data analysis, providing a structured mechanism for transforming raw data into comprehensible insights.

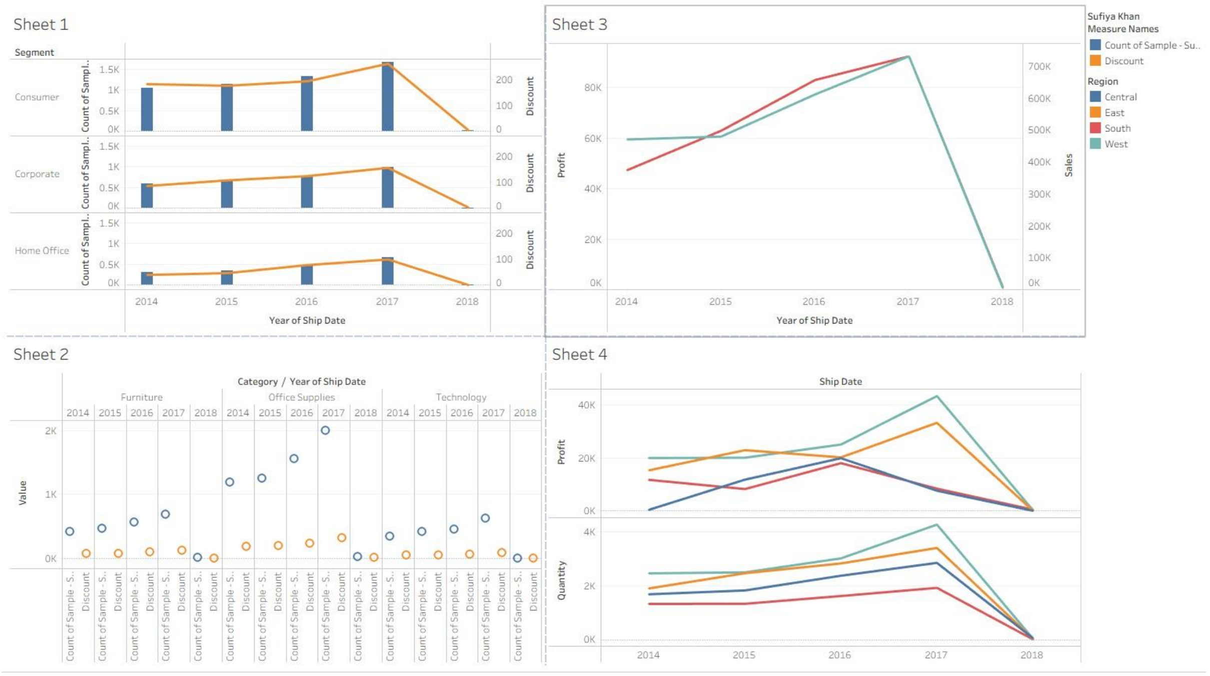
At its core, data reporting is the systematic process of presenting information in a meaningful and accessible format. It involves the organization and presentation of data points to communicate trends, patterns, and critical metrics. In the ever-expanding world of data-driven decision-making, data reporting plays a critical role in distilling complex datasets into actionable intelligence.

This page explores the fundamental aspects of data reporting, from its definition and working principles to its applications and the diverse types of reports it encompasses. Understanding data reporting is integral to harnessing the full potential of data in making informed decisions across various domains.

Data reporting is the process of collecting and submitting data. The effective management of any organization relies on accurate data. Inaccurate data reporting can lead to poor decision-making based on erroneous evidence. Data reporting is different from data analysis which transforms data and information into insights. Data reporting is the previous step that translates raw data into information. When data is not reported, the problem is known as underreporting; the opposite problem leads to false positives.

Data reporting can be difficult. Census bureaus may hire perhaps hundreds of thousands of workers to achieve the task of counting all of the residents of a country. Teachers use data from student assessments to determine grades; manufacturers rely on sales data from retailers to indicate which products should have increased production, and which should be curtailed or discontinued.

Output:



Practical No: 10

Name: Hammad Ansari

Roll No: 15

Date:

Aim: Create a data story using Tableau or Power BI.

Theory:

Story Telling

A data story is a narrative constructed around a set of data that puts it into context and frames the broader implications. Unlike business intelligence or data science that emphasizes the technical task of turning data into insights, a data story brings these insights together with qualitative analysis and domain expertise to better understand a relevant business goal or objective.

Data storytelling is the skill to craft the narrative by leveraging data, which is then contextualized, and finally presented to an audience. It utilizes not only data analysis and statistics, but also data visualization, qualitative and contextual analysis, and presentation.

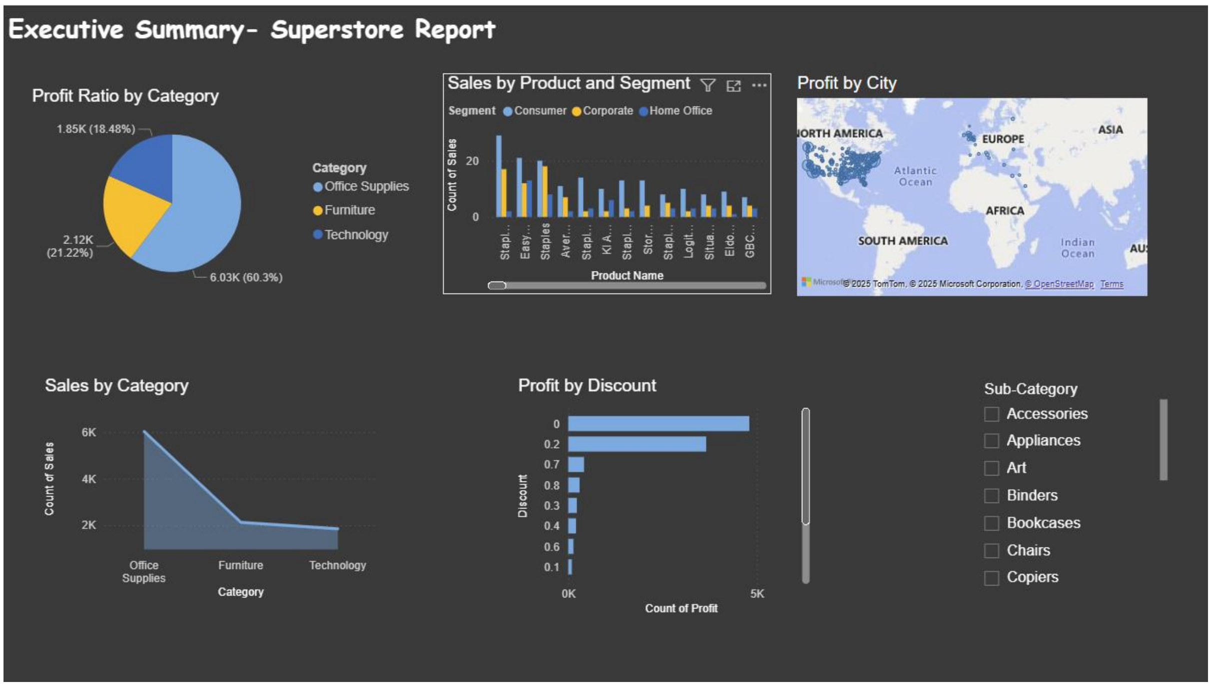
Telling a story with data is important because it allows the narrator to put the data into context of a broader objective and use tools such as visual aids to help break down the results so that the audience, regardless of their background, domain expertise, or technical sophistication, can easily understand them and their implications.

Data storytelling also helps to explain data to people of different learning styles and allows the narrator to craft the communication methods to the respective audience. For instance, an auditory learner may respond better to a spoken presentation while a visual learner may require more data visualizations and other visual aids. In general, a good data story will include a combination of these various components so a diverse audience can stay engaged.

Data storytelling is a way to convey and interpret complicated metrics for business executives and customers. A good data story uses interactive data visualizations to present a compelling narrative that makes the entire process and message easy to understand.

Telling stories frames information in a way that's clear and memorable. A story can engage people and present the data in a way they can process, comprehend and empathize with any effects the data shows.

Output:


Executive Summary- Superstore Report
Profit Ratio by Category
1.85K (18.48%)
2.12K (21.22%)
6.03K (60.3%)
Category
● Office Supplies
● Furniture
● Technology
Sales by Product and Segment
Segment ● Consumer ● Corporate ● Home Office
Count of Sales
20
0
Stapl...
Easy...
Staples
Aver...
Stapl...
KI A...
Stapl...
Stor...
Stapl...
Logit...
Situa...
Eldo...
GBC...
Product Name
Profit by City
NORTH AMERICA
EUROPE
ASIA
Atlantic Ocean
AFRICA
SOUTH AMERICA
Indian Ocean
© 2025 TomTom, © 2025 Microsoft Corporation, © OpenStreetMap Terms
Sales by Category
Count of Sales
6K
4K
2K
Office Supplies
Furniture
Technology
Category
Profit by Discount
Discount
0
0.2
0.7
0.8
0.3
0.4
0.6
0.1
0K
5K
Count of Profit
Sub-Category
Accessories
Appliances
Art
Binders
Bookcases
Chairs
Copiers