# DSDMS LAB Topics Highlighted

**1 Aggregate values**

import pandas as pd

data = {

'RepName': ['Alice', 'Bob', 'Charlie', 'Alice', 'Bob', 'Charlie'],

'Leads': [45, 30, 25, 50, 40, 35],

'Sales': [20, 15, 10, 25, 20, 18],

'WorkHours': [160, 150, 145, 170, 155, 148]

}

df = pd.DataFrame(data)

print("Sales Team Data:\n", df, "\n")

print("Total Leads & Sales:\n", df[['Leads', 'Sales']].sum(), "\n")

print("Average Sales & WorkHours:\n", df[['Sales', 'WorkHours']].mean(), "\n")

print("EX.NO:1(a) DEMONSTRATE AGGREGATION DATE:\n")

print("Min & Max Leads by Rep:\n", df.groupby('RepName')['Leads'].agg(['min', 'max']), "\n")

print("Avg WorkHours by Rep:\n", df.groupby('RepName')['WorkHours'].mean())

**1 Education**

import pandas as pd

data = {

'Name': ["Alice", "Bob", "Charlie", "David", "Eva"],

'Math': [85, 78, 92, 88, 76],

'Science': [90, 82, 88, 76, 95],

'English': [78, 85, 80, 90, 88]

}

df = pd.DataFrame(data)

print("Student Exam Scores:\n", df, "\n")

print("EX.NO:1(b) EDUCATIONAL STUDENT PERFORMANCE DATE:\n")

print("Average Scores:\n", df[['Math', 'Science', 'English']].mean(), "\n")

print("Highest Scores:\n", df[['Math', 'Science', 'English']].max(), "\n")

print("Lowest Scores:\n", df[['Math', 'Science', 'English']].min())

**1 Health data:**

import pandas as pd

df = pd.DataFrame({

'Name': ["Jaya","Kalai","Mani","Surya","Kumar","Jaya","Mani","Kavi","Kani"],

'Age': [25,45,35,60,52,28,33,64,36],

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

'BP': [125,135,135,145,138,118,132,122,147],

'AgeGrp': ['20-30','40-50','30-40','60-70','50-60','20-30','30-40','60-70','30-40']

})

print("Patient Data:\n", df, "\n")

print("BP Analysis by Age Group:\n", df.groupby('AgeGrp')['BP'].agg(['mean','min','max']), "\n")

print("Total BP & Age:\n", df[['BP','Age']].sum())

**1 Real estate**

import pandas as pd

data = {

'City': ['Chennai', 'Bangalore', 'Chennai', 'Delhi', 'Delhi', 'Bangalore'],

'Price': [7500000, 8200000, 6900000, 12000000, 9500000, 8700000],

'Area': [1500, 1600, 1400, 2000, 1800, 1750],

'Bedrooms': [3, 4, 2, 4, 3, 5]

}

df = pd.DataFrame(data)

print("Real Estate Property Analysis:\n")

print(df.groupby('City').agg({'Price':'mean','Area':'min','Bedrooms':'max'}))

**1 Online store**

import pandas as pd

df = pd.read\_csv('retail.csv')

print("Customer Purchase Analysis:\n")

print(df.groupby('Customer\_ID')['Purchase\_Amount'].agg(['sum', 'min', 'max', 'mean']))

**2 shopping Indexing & sorting**

import pandas as pd

data = {

'Product': ['Laptop', 'Headphones', 'Keyboard', 'Mouse', 'Monitor', 'Charger'],

'Price': [60000, 1500, 1200, 800, 10000, 900],

'Category': ['Electronics', 'Accessories', 'Accessories', 'Accessories', 'Electronics', 'Accessories'],

'Discount': [5000, 200, 150, 100, 1200, 100]

}

df = pd.DataFrame(data)

df.index.name = 'Index'

print("EX.NO:2(a) DEMONSTRATE INDEXING AND SORTING DATE:\n")

print("Indexed Shopping Data:\n", df, "\n")

print("Sorted by Product:\n", df.sort\_values(by='Product'))

**2 Astronomical analysis**

import pandas as pd

data = {

'ObservationID': ['OBS001', 'OBS002', 'OBS003', 'OBS004', 'OBS005'],

'StarName': ['Sirius', 'Betelgeuse', 'Rigel', 'Vega', 'Polaris'],

'Magnitude': [1.46, 0.42, 0.18, 0.03, 1.97],

'ObservationDate': ['2023-12-10', '2023-11-20', '2023-12-01', '2023-10-05', '2023-12-15']

}

df = pd.DataFrame(data)

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

df.set\_index('ObservationID', inplace=True)

print("EX.NO:2(b) DATAFRAME OPERATIONS FOR ASTRONOMICAL DATA ANALYSIS DATE:\n")

sorted\_df = df.sort\_values(by=['Magnitude', 'ObservationDate'])

print("Astronomical Observation Data (Sorted):\n", sorted\_df)

**2 Movie database management**

import pandas as pd

df = pd.DataFrame({

'Title': ["Inception","Titanic","Avatar","The Matrix","Interstellar","Good Will Hunting","Sherlock Holmes"],

'Genre': ["Sci-Fi","Romance","Sci-Fi","Action","Sci-Fi","Drama","Mystery"],

'Year': [2010,1997,2009,1999,2014,1997,2009],

'Rating': [8.8,7.8,7.9,8.7,8.6,8.3,7.6]

})

df.index.name = "Index"

print("Movies:\n", df, "\n")

print("Sorted by Year & Rating:\n", df.sort\_values(['Year','Rating']))

**2 Library management**

import pandas as pd

data = {

'Book\_ID': [101, 102, 103, 104],

'Title': ['AI Basics', 'Python Guide', 'ML Concepts', 'Data Science'],

'Author': ['John', 'Sara', 'Ravi', 'Anu'],

'Copies': [4, 10, 6, 8]

}

df = pd.DataFrame(data)

print("Original Data:\n", df, "\n")

df.set\_index('Book\_ID', inplace=True)

print("After Setting Index:\n", df, "\n")

print("Sorted by Copies (Descending):\n", df.sort\_values('Copies', ascending=False))

**2 Sports team**

import pandas as pd

data = {

'Player': ['Rohit', 'Kohli', 'Dhoni', 'Hardik'],

'Matches': [250, 270, 350, 150],

'Runs': [9800, 12000, 10700, 4500]

}

df = pd.DataFrame(data)

print("Original Data:\n", df, "\n")

df.set\_index('Player', inplace=True)

print("After Setting Index:\n", df, "\n")

print("Sorted by Runs (Descending):\n", df.sort\_values('Runs', ascending=False))

**2 Customer feedback**

import pandas as pd

data = {

'Customer\_ID': [201, 202, 203, 204],

'Name': ['Alice', 'Bob', 'Charlie', 'David'],

'Rating': [4, 5, 3, 4],

'Feedback': ['Good', 'Excellent', 'Average', 'Good']

}

df = pd.DataFrame(data)

print("Original Data:\n", df, "\n")

df.set\_index('Customer\_ID', inplace=True)

print("After Setting Index:\n", df, "\n")

print("Sorted by Rating (Descending):\n", df.sort\_values('Rating', ascending=False))

**2 sort\_index(),values()**

import pandas as pd

df = pd.DataFrame({

'ID': [103, 101, 102, 104],

'Name': ['David', 'Alice', 'Bob', 'Charlie'],

'Marks': [85, 92, 78, 88]

})

print("Original Data:\n", df, "\n")

df\_sorted\_index = df.sort\_index()

print("Sorted by Index:\n", df\_sorted\_index, "\n")

df\_sorted\_values = df.sort\_values(by='Marks')

print("Sorted by Marks:\n", df\_sorted\_values)

**3 patient Missing data**

import pandas as pd

df = pd.read\_csv('patient\_record.csv')

missing\_data = df.isnull()

print("Original Dataset Information:")

df.info()

df = df.fillna(df.mean(numeric\_only=True))

print("\nCleaned Patient Dataset:")

print(df)

print("\nMissing Data Map:")

print(missing\_data)

**3 Missing crop yield**

import pandas as pd

df = pd.read\_csv('agriculture\_data.csv')

print("Original Data:\n", df, "\n")

df['CropYield'] = df['CropYield'].fillna(df['CropYield'].mean())

print("EX.NO:3(b) HANDLING MISSING CROP YIELD INFO\n")

print("Updated Data:\n", df)

**3 missing in humidity,temp**

import pandas as pd

df = pd.read\_csv('weather\_data.csv')

print("Original Data:\n", df, "\n")

df['Temperature'] = df['Temperature'].fillna(df['Temperature'].mean())

df['Humidity'] = df['Humidity'].fillna(df['Humidity'].mean())

print("EX.NO:3(c) HANDLING MISSING TEMPERATURE AND HUMIDITY DATA\n")

print("Updated Data:\n", df)

**3 sports missing data**

import pandas as pd

import numpy as np

data = {

'Player': ['Rohit', 'Kohli', 'Dhoni', 'Hardik', 'Raina'],

'Matches': [250, np.nan, 350, 150, np.nan],

'Runs': [9800, 12000, np.nan, 4500, 7000]

}

df = pd.DataFrame(data)

print("Original Data:\n", df, "\n")

df['Matches'] = df['Matches'].fillna(df['Matches'].mean())

df['Runs'] = df['Runs'].fillna(df['Runs'].median())

print("After Handling Missing Data:\n", df)

**3 E commerce missing data**

import pandas as pd

import numpy as np

df = pd.DataFrame({

'Order\_ID': [1, 2, 3, 4, 5],

'Customer': ['Alice', 'Bob', 'Charlie', np.nan, 'Eve'],

'Amount': [2500, np.nan, 1800, 3000, np.nan],

'Category': ['Electronics', 'Fashion', 'Grocery', 'Fashion', np.nan]

})

df['Amount'] = df['Amount'].fillna(df['Amount'].mean())

df = df.fillna({'Customer': 'Unknown', 'Category': 'Unknown'})

print(df)

**3 general data frame(dropna,fillna,interpolate)**

import pandas as pd

import numpy as np

df = pd.DataFrame({

'Name': ['A', 'B', np.nan, 'D', 'E'],

'Age': [20, np.nan, 25, np.nan, 22],

'City': ['Delhi', 'Mumbai', 'Chennai', np.nan, 'Pune']

})

df = df.dropna(subset=['Name'])

df['Age'] = df['Age'].interpolate().round().astype(int)

df['City'] = df['City'].fillna('Unknown')

print(df)

**4 student Hierarachical indexing**

import pandas as pd

df = pd.read\_csv('students.csv')

print("Original Data:\n", df.head(), "\n")

indexed\_df = df.set\_index(['Class', 'Section', 'Name'])

print("DataFrame with Hierarchical Indexing:\n", indexed\_df.head())

**4 prof dept hierarchy**

import pandas as pd

df = pd.DataFrame({

'Department': ['CSE', 'CSE', 'ECE', 'ECE', 'MECH', 'MECH'],

'Course': ['AI', 'ML', 'VLSI', 'Embedded', 'Thermo', 'Design'],

'Professor': ['Dr. Rao', 'Dr. Meena', 'Dr. Kumar', 'Dr. Priya', 'Dr. Ravi', 'Dr. Rekha'],

'Students': [120, 80, 90, 100, 70, 110],

'Avg\_Grade': ['A', 'B+', 'A-', 'B', 'A', 'B+']

})

print("Original Data:\n", df, "\n")

indexed\_df = df.set\_index(['Department', 'Course', 'Professor'])

print("DataFrame with Hierarchical Indexing:\n", indexed\_df)

**4 supplychain management**

import pandas as pd

df = pd.DataFrame({

'Warehouse': ['W1', 'W2', 'W1', 'W3', 'W2', 'W3'],

'Product': ['Laptop', 'Laptop', 'Mobile', 'Mobile', 'TV', 'TV'],

'Stock': [100, 80, 120, 90, 60, 70]

})

print("Original DataFrame:\n", df, "\n")

df.set\_index(['Warehouse', 'Product'], inplace=True)

print("DataFrame with Hierarchical Indexing:\n", df)

**4 network,service,customer**

import pandas as pd

df = pd.DataFrame({

'Network': ['5G', '5G', '4G', '4G', 'Fiber', 'Fiber'],

'Service': ['Streaming', 'Streaming', 'Browsing', 'Browsing', 'Gaming', 'Gaming'],

'Customer\_Segment': ['Premium', 'Basic', 'Premium', 'Basic', 'Premium', 'Basic'],

'Active\_Users': [5000, 3200, 4500, 2800, 4000, 3700],

'Avg\_Usage\_GB': [50, 45, 60, 30, 55, 40]

})

print("Original DataFrame:\n", df, "\n")

df.set\_index(['Network', 'Service', 'Customer\_Segment'], inplace=True)

print("DataFrame with Hierarchical Indexing:\n", df)

**4 Indexing & display**

import pandas as pd

df = pd.read\_csv('energy\_data.csv', header=[0, 1, 2])

print("Original DataFrame with Hierarchical Columns:\n")

print(df)

print("\nColumn Index Levels:", df.columns.nlevels)

**5 Pivot table**

import pandas as pd

df = pd.DataFrame({

"Student": ["Alice", "Alice", "Alice", "Bob", "Bob", "Charlie", "Charlie", "Charlie", "Charlie"],

"Section": ["A", "A", "A", "B", "B", "A", "A", "B", "B"],

"Subject": ["Math", "Science", "Science", "Math", "Math", "Science", "Math", "Math", "Science"],

"Test1\_Score": [85, 78, 88, 90, 95, 70, 80, 60, 75],

"Test2\_Score": [80, 85, 90, 92, 88, 75, 85, 65, 78]

})

print("EX.NO:5(a) DEMONSTRATE USAGE OF PIVOT TABLE DATE:\n")

print("Original DataFrame:\n", df, "\n")

print("Basic Pivot Table:\n", pd.pivot\_table(df, index=['Student', 'Section'],

values=['Test1\_Score', 'Test2\_Score']), "\n")

print("Aggregate (Sum):\n", pd.pivot\_table(df, index=['Student', 'Section'],

values=['Test1\_Score', 'Test2\_Score'],

aggfunc='sum'), "\n")

print("Mean by Student & Subject:\n", pd.pivot\_table(df, index=['Student', 'Subject'],

values=['Test1\_Score', 'Test2\_Score'],

aggfunc='mean'), "\n")

print("Multiple Aggregations:\n", pd.pivot\_table(df, index=['Student', 'Subject'],

values=['Test1\_Score', 'Test2\_Score'],

aggfunc={'Test1\_Score': 'mean',

'Test2\_Score': ['min', 'max', 'mean']}))

**5 Small business pivot table**

import pandas as pd

# Sample data

data = {

'Month': ['Jan', 'Jan', 'Feb', 'Feb', 'Mar', 'Mar', 'Jan', 'Feb', 'Mar'],

'Product': ['Shoes', 'Bags', 'Shoes', 'Bags', 'Shoes', 'Bags', 'Shoes',

'Shoes', 'Bags'],

'Sales': [5000, 3000, 6000, 3500, 7000, 4000, 4500, 5500, 4200],

'Returns': [200, 150, 250, 100, 300, 200, 180, 220, 170]

}

# Create DataFrame

df = pd.DataFrame(data)

# Create a pivot table

pivot\_table = pd.pivot\_table(

df,

values=['Sales', 'Returns'],

index='Product',

columns='Month',

aggfunc='sum',

fill\_value=0

)

# Display pivot table

print("Pivot Table - Monthly Sales and Returns by Product Type:\n")

print(pivot\_table)

**5 stock price,moving avg**

import pandas as pd

df = pd.DataFrame({

'Date': ['2025-10-01', '2025-10-01', '2025-10-02', '2025-10-02', '2025-10-03', '2025-10-03'],

'Company': ['AAPL', 'MSFT', 'AAPL', 'MSFT', 'AAPL', 'MSFT'],

'Stock\_Price': [180, 320, 185, 325, 190, 330],

'Volume': [2000, 2500, 2100, 2400, 2300, 2600]

})

print("Original Data:\n", df, "\n")

pivot = pd.pivot\_table(df, index='Date', columns='Company', values=['Stock\_Price', 'Volume'])

print("Pivot Table:\n", pivot, "\n")

aapl = df[df['Company'] == 'AAPL'].copy()

aapl['Moving\_Avg'] = aapl['Stock\_Price'].rolling(3).mean()

print("AAPL Moving Average:\n", aapl[['Date', 'Stock\_Price', 'Moving\_Avg']])

**5 manufac**

import pandas as pd

data = {

'Factory': ['F1', 'F1', 'F2', 'F2', 'F3', 'F3'],

'Product': ['A', 'B', 'A', 'B', 'A', 'B'],

'Units\_Produced': [500, 400, 550, 420, 600, 450],

'Defective\_Units': [10, 15, 12, 18, 8, 20]

}

df = pd.DataFrame(data)

print("Original Data:\n", df, "\n")

pivot = pd.pivot\_table(df, index='Factory', columns='Product',

values=['Units\_Produced', 'Defective\_Units'],

aggfunc=['sum', 'mean'])

print("Pivot Table:\n", pivot)

**6 Eval()**

expr = input("Enter a math expression (in terms of x): ")

x = float(input("Enter the value of x: "))

result = eval(expr)

print("Result =", result)

**6 eval math op**

import numpy as np

import pandas as pd

expr = input("Enter the function (in terms of x, y, and z): ")

x = int(input("Enter the value of x: "))

y = int(input("Enter the value of y: "))

z = int(input("Enter the value of z: "))

result = eval(expr)

print("Result", result)

**6 Currency conversion**

exchange = {'USD': 1, 'INR': 83.2, 'EUR': 0.85, 'JPY': 140.3}

amt = float(input("Enter amount: "))

f = input("From: ").upper()

t = input("To: ").upper()

if f in exchange and t in exchange:

result = amt / exchange[f] \* exchange[t]

print(f"{amt} {f} = {round(result, 2)} {t}")

else:

print("Invalid currency")

**6 query()**

import pandas as pd

df = pd.DataFrame({

'Name': ['A', 'B', 'C', 'D'],

'Age': [25, 30, 35, 28],

'Salary': [40000, 50000, 60000, 45000]

})

print(df.query('Age > 25'))

df['Bonus'] = pd.eval('df.Salary \* 0.1')

print(df)

**6 multi level sales data analysis**

import pandas as pd

df = pd.DataFrame({

'Product': ['Laptop', 'Mobile', 'TV'],

'Sales': [50000, 30000, 60000],

'Profit': [8000, 5000, 9000]

})

print(df.query('Sales > 40000'))

df['Profit\_%'] = pd.eval('(df.Profit / df.Sales) \* 100')

print(df)

**7 Scatter plot**

import matplotlib.pyplot as plt

hours = [2, 3, 4, 5, 6, 7, 8, 9, 10]

scores = [50, 55, 60, 65, 70, 75, 80, 85, 90]

plt.scatter(hours, scores, color='red', marker='o')

plt.title("Study Hours vs Exam Scores")

plt.xlabel("Hours of Study")

plt.ylabel("Exam Score (%)")

plt.show()

**7 3D Scatter plot**

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

x = [10, 20, 30]

y = [15, 25, 35]

z = [5, 10, 15]

ax = plt.axes(projection='3d')

ax.scatter(x, y, z, color='purple')

plt.show()

**7 3D plot**

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

x = [1, 2, 3, 4]

y = [2, 3, 4, 5]

z = [3, 4, 5, 6]

ax = plt.axes(projection='3d')

ax.plot3D(x, y, z)

plt.show()

**7 3d pollution scatter plot**

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

PM2\_5 = [180, 120, 90, 150, 70]

PM10 = [250, 180, 140, 200, 100]

NO2 = [80, 60, 40, 55, 35]

ax = plt.axes(projection='3d')

ax.scatter(PM2\_5, PM10, NO2, color='green')

ax.set\_xlabel('PM2.5')

ax.set\_ylabel('PM10')

ax.set\_zlabel('NO2')

plt.show()

**7 3D pie,bar chart**

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

x = [1, 2, 3, 4]

y = [30, 20, 40, 10]

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.bar(x, y, zs=0, zdir='y', color='orange', alpha=0.8)

ax.set\_xlabel('Category')

ax.set\_ylabel('Y')

ax.set\_zlabel('Value')

plt.title('3D Bar Chart')

plt.show()

**7 2D line plot**

import matplotlib.pyplot as plt

years = [2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]

temp\_change = [0.62, 0.74, 0.90, 1.01, 0.93, 0.85, 0.98, 1.02, 0.85, 0.89]

print("EX.NO:7(b) DEMONSTRATE 2D LINE PLOT FOR GLOBAL TEMPERATURE CHANGE DATE:\n")

plt.plot(years, temp\_change, marker='o')

plt.title("Global Temperature Change (2013-2022)")

plt.xlabel("Year")

plt.ylabel("Temperature Change (°C)")

plt.grid(True)

plt.show()

**7 2d pie,bar chart**

import matplotlib.pyplot as plt

x = ['A', 'B', 'C', 'D']

y = [30, 20, 40, 10]

plt.bar(x, y, color='skyblue')

plt.title('Bar Chart')

plt.show()

plt.pie(y, labels=x, autopct='%1.1f%%')

plt.title('Pie Chart')

plt.show()

**8 Facebook**

import pandas as pd

import matplotlib.pyplot as plt

data = {

'Post\_ID': [101, 102, 103, 104, 105],

'Likes': [250, 400, 300, 500, 450],

'Love': [100, 120, 80, 150, 130],

'Haha': [40, 60, 30, 50, 70],

'Angry': [10, 15, 5, 8, 12]

}

df = pd.DataFrame(data)

df['Total'] = df[['Likes', 'Love', 'Haha', 'Angry']].sum(axis=1)

print(df)

df.plot(x='Post\_ID', y=['Likes', 'Love', 'Haha', 'Angry'], kind='bar')

plt.title('Facebook Reactions')

plt.show()

**8 Traffic**

import pandas as pd

import matplotlib.pyplot as plt

data = {'Time': ['Morning', 'Noon', 'Evening', 'Night'],

'Vehicles': [1200, 800, 1500, 500]}

df = pd.DataFrame(data)

print("Peak Traffic:", df.loc[df['Vehicles'].idxmax(), 'Time'])

plt.plot(df['Time'], df['Vehicles'], marker='o')

plt.title('Vehicle Traffic Analysis')

plt.show()

**8 Spotify**

import matplotlib.pyplot as plt

artists = ['Arijit', 'Shreya', 'Darshan', 'Armaan', 'Neha']

streams = [450, 380, 300, 270, 250]

print("Most Streamed Artist:", artists[streams.index(max(streams))])

plt.bar(artists, streams, color='purple')

plt.title('Spotify Music Streams')

plt.xlabel('Artist')

plt.ylabel('Streams (Millions)')

plt.show()

**8 Smart home**

import matplotlib.pyplot as plt

rooms = ['Living Room', 'Bedroom', 'Kitchen', 'Bathroom']

energy = [120, 90, 150, 60]

# Bar chart

plt.bar(rooms, energy, color='orange')

plt.title('Energy Consumption')

plt.show()

# Pie chart

plt.pie(energy, labels=rooms, autopct='%1.1f%%')

plt.title('Energy Consumption Share')

plt.show()

**9 twitter:**

import matplotlib.pyplot as plt

sentiments = ['Positive', 'Negative', 'Neutral', 'Positive', 'Negative', 'Positive']

counts = {'Positive': 3, 'Negative': 2, 'Neutral': 1}

plt.bar(counts.keys(), counts.values(), color=['green', 'red', 'gray'])

plt.title('Tweet Sentiment Analysis')

plt.show()

**9 Traveller sntiment data analysis**

from textblob import TextBlob

import pandas as pd

data = ['Good service', 'Bad flight', 'Amazing stay', 'Average food']

df = pd.DataFrame(data, columns=['Review'])

df['Sentiment'] = df['Review'].apply(lambda x: 'Positive' if TextBlob(x).sentiment.polarity > 0 else 'Negative')

print(df)

**9 news sentiment**

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk

nltk.download('vader\_lexicon')

news = [

"Stock market gained this week.",

"Floods caused major damage.",

"New reforms improved education."

]

sia = SentimentIntensityAnalyzer()

for n in news:

score = sia.polarity\_scores(n)['compound']

sentiment = 'Positive' if score > 0.05 else ('Negative' if score < -0.05 else 'Neutral')

print(n, "→", sentiment)

**9 semantic analysis**

import pandas as pd

from textblob import TextBlob

import matplotlib.pyplot as plt

data = {

'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'],

'Sales': [12000, 15000, 17000, 14000, 19000],

'Review': [

"Loved the product!",

"Slow delivery.",

"Excellent quality.",

"Average service.",

"Great support!"

]

}

df = pd.DataFrame(data)

df['Sentiment'] = df['Review'].apply(lambda x: 'Positive' if TextBlob(x).sentiment.polarity > 0 else 'Negative')

print(df)

plt.plot(df['Month'], df['Sales'], marker='o')

plt.title("Monthly Sales Trend")

plt.show()

**9 entity recognition**

import re

# Sample text

text = "In 2023, India recorded a birth rate of 17.4 and a death rate of 7.3 per 1000 people."

# Find numbers (birth/death rates, years)

numbers = re.findall(r'\d+\.?\d\*', text)

# Find capitalized words (possible countries or names)

entities = re.findall(r'\b[A-Z][a-z]+\b', text)

print("Numbers found:", numbers)

print("Possible entities:", entities)

**10 Google tools:**

import pandas as pd

import matplotlib.pyplot as plt

data = {

'Region': ['USA', 'India', 'UK', 'Canada', 'Australia'],

'Facebook': [90, 85, 70, 60, 75],

'Apple': [80, 65, 60, 55, 70],

'Amazon': [95, 80, 75, 65, 85],

'Netflix': [70, 60, 55, 50, 65],

'Google': [100, 95, 80, 75, 90]

}

df = pd.DataFrame(data).set\_index('Region')

print(df)

df.plot(kind='bar')

plt.title("Interest by Region")

plt.show()

**10 Universities data**

import pandas as pd

import matplotlib.pyplot as plt

df = pd.DataFrame({

'name': ['MIT', 'Harvard', 'Stanford', 'Yale', 'Princeton'],

'state': ['MA', 'MA', 'CA', 'CT', 'NJ']

})

print("Universities:\n", df['name'].tolist())

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

print("\nUniversities by State:\n", state\_counts)

state\_counts.plot(kind='bar', color='skyblue')

plt.title("Universities by State")

plt.xlabel("State")

plt.ylabel("Count")

plt.show()

extra:

**1 ROI**

import pandas as pd

df = pd.read\_csv('digital\_campaign.csv')

print("Original Dataset:")

print(df)

df['ROI (%)'] = ((df['Revenue'] - df['Investment']) / df['Investment']) \* 100

print("\nDataset with ROI:")

print(df)

**2 KNNImpute**

import pandas as pd

import numpy as np

from sklearn.impute import KNNImputer

data = {

'Math': [85, np.nan, 78, 92, np.nan],

'Science': [90, 88, np.nan, 95, 70]

}

df = pd.DataFrame(data)

print("Before:\n", df)

imputer = KNNImputer(n\_neighbors=2)

df[:] = imputer.fit\_transform(df)

print("\nAfter KNN Imputation:\n", df)

**3 merged data of multiple platforms**

import pandas as pd

# Sample data from two platforms

fb = pd.DataFrame({'User': ['A', 'B'], 'Likes': [120, 80]})

tw = pd.DataFrame({'User': ['A', 'C'], 'Likes': [100, 90]})

# Merge both

merged = pd.concat([fb, tw], ignore\_index=True)

print("Merged Data:")

print(merged)

**4 daily,monthly sales**

sales = [200, 250, 180, 300, 270]

total = 0

for s in sales:

total += s

daily\_avg = total / len(sales)

monthly\_avg = daily\_avg \* 4

print("Daily:", daily\_avg, "Monthly:", monthly\_avg)

**5 Heatmap**

import pandas as pd

import matplotlib.pyplot as plt

# Sample marketing dataset

data = {

'Campaign': ['A', 'B', 'C', 'D'],

'Clicks': [1200, 900, 1500, 700],

'Impressions': [10000, 8500, 12000, 6000],

'Conversions': [120, 90, 160, 60],

'Cost': [500, 400, 550, 300]

}

df = pd.DataFrame(data)

print(df)

# Create heatmap

corr = df[['Clicks', 'Impressions', 'Conversions', 'Cost']].corr()

plt.imshow(corr, cmap='coolwarm', interpolation='none')

plt.colorbar()

plt.title("Campaign Performance Heatmap")

plt.show()

**6 simple marketing data set**

import pandas as pd

# Sample marketing dataset

data = {

'Campaign': ['A', 'B', 'C', 'D'],

'Clicks': [1200, 900, 1500, 700],

'Impressions': [10000, 8500, 12000, 6000],

'Conversions': [120, 90, 160, 60]

}

df = pd.DataFrame(data)

# Explore dataset

print(df)

print("\nDataset Info:")

print(df.info())

**7 built in functions**

import pandas as pd

data = {

'Sales': [200, 350, 150, 400, 300]

}

df = pd.DataFrame(data)

print("Length:", len(df['Sales']))

print("Maximum:", df['Sales'].max())

print("Minimum:", df['Sales'].min())

print("Sum:", df['Sales'].sum())

**8 pd.to\_datetime**

import pandas as pd

data = {'Date': ['2025-01-01', '2025-02-15', '2025-03-10']}

df = pd.DataFrame(data)

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

print(df)

print("Month:", df['Date'].dt.month)

print("Day:", df['Date'].dt.day)

**9 add campaign(agg func)**

import pandas as pd

data = {

'Campaign': ['A', 'B', 'C', 'D'],

'Impressions': [1200, 1500, 1800, 1100],

'Reach': [800, 1000, 1200, 700]

}

df = pd.DataFrame(data)

total\_impressions = df['Impressions'].sum()

avg\_reach = df['Reach'].mean()

print("Total Ad Impressions:", total\_impressions)

print("Average Reach:", avg\_reach)

**10 platform engament**

import pandas as pd

import matplotlib.pyplot as plt

data = {

'Platform': ['Facebook', 'Instagram', 'Twitter', 'LinkedIn'],

'Engagements': [1200, 1800, 900, 1100]

}

df = pd.DataFrame(data)

avg = df['Engagements'].mean()

print("Average Engagement:", avg)

plt.bar(df['Platform'], df['Engagements'], color='skyblue')

plt.title('Average Engagement per Platform')

plt.xlabel('Platform')

plt.ylabel('Engagements')

plt.show()

**11 np.where()**

import pandas as pd

import numpy as np

data = {'Sales': [400, 800, 300, 1000]}

df = pd.DataFrame(data)

df['Performance'] = np.where(df['Sales'] > 500, 'Good', 'Average')

print(df)

**12 linear regression**

import pandas as pd

from sklearn.linear\_model import LinearRegression

data = pd.DataFrame({

'Ad\_Spend': [10, 20, 30, 40, 50],

'Revenue': [25, 45, 65, 85, 105]

})

X = data[['Ad\_Spend']]

y = data['Revenue']

model = LinearRegression()

model.fit(X, y)

pred = model.predict(pd.DataFrame({'Ad\_Spend': [60]}))

print("Predicted Revenue for Ad\_Spend=60:", pred[0])

**13 time of day analysis**

import pandas as pd

import matplotlib.pyplot as plt

data = {'Time': ['Morning','Noon','Evening','Night'],

'Engagements': [120, 80, 150, 60]}

df = pd.DataFrame(data)

plt.plot(df['Time'], df['Engagements'], marker='o')

plt.title('Post Performance by Time of Day')

plt.xlabel('Time')

plt.ylabel('Engagements')

plt.show()

**14 ctr**

import pandas as pd

data = {

'Ad\_ID': [1, 2, 3, 4],

'Clicks': [200, 150, 300, 100],

'Impressions': [4000, 3000, 5000, 2500]

}

df = pd.DataFrame(data)

df['CTR (%)'] = (df['Clicks'] / df['Impressions']) \* 100

print(df)

**15 merge datasets**

import pandas as pd

# Sample data (you can replace these with your CSV file paths)

df1 = pd.DataFrame({'ID': [1, 2], 'Name': ['Alice', 'Bob']})

df2 = pd.DataFrame({'ID': [3, 4], 'Name': ['Charlie', 'David']})

# Merge the data

merged\_df = pd.concat([df1, df2])

print("Merged Dataset:")

print(merged\_df)

**16 stacked/unstacked**

import pandas as pd

# Sample data

data = {'Month': ['Jan', 'Jan', 'Feb', 'Feb'],

'Product': ['A', 'B', 'A', 'B'],

'Sales': [100, 150, 200, 250]}

df = pd.DataFrame(data)

print("Original DataFrame:")

print(df)

# Pivot the data

pivot = df.pivot(index='Month', columns='Product', values='Sales')

print("\nPivoted DataFrame:")

print(pivot)

# Stack and Unstack

stacked = pivot.stack()

print("\nStacked DataFrame:")

print(stacked)

unstacked = stacked.unstack()

print("\nUnstacked DataFrame:")

print(unstacked)

**17 rolling window analysis**

import pandas as pd

# Sample data

data = {'Sales': [10, 20, 30, 40, 50, 60]}

df = pd.DataFrame(data)

# Rolling mean (window size = 3)

df['Rolling\_Avg'] = df['Sales'].rolling(3).mean()

print(df)