**Task 1: Exploration of Data Visualization Tools**

**Tools:** Tableau, Python Libraries (pandas, matplotlib, seaborn), D3.js  
**CO1, S2:** Understand and use data visualization tools for data exploration.

**1.a) Exploration and Preparation of Data**

**Aim:**

To connect and explore a dataset, understand its structure, types, and compute statistics using Python libraries.

**Algorithm:**

1. Import necessary Python libraries (pandas, numpy, etc.).
2. Load the dataset (CSV/Excel format).
3. Display the first few records of the dataset.
4. Show dataset information (columns, data types, missing values).
5. Compute descriptive statistics for numerical columns.
6. Display the shape (rows × columns) of the dataset.

**Python Program (1.a):**

# Import required libraries

import pandas as pd

import numpy as np

# Read the dataset

data = pd.read\_csv("student\_scores.csv")

# Display first few rows

print("Dataset Preview:")

print(data.head())

# Display information about dataset

print("\nDataset Information:")

print(data.info())

# Check dimensionality

print("\nShape of dataset:", data.shape)

# Display column names and data types

print("\nColumn Data Types:")

print(data.dtypes)

# Check for missing values

print("\nMissing Values:")

print(data.isnull().sum())

# Compute descriptive statistics

print("\nDescriptive Statistics:")

print(data.describe())

**Output:**

Dataset Preview:

Name Age Gender Math Science English

0 Alex 16 M 78 88 82

1 Bella 17 F 85 79 91

2 Chris 16 M 92 95 89

3 Daisy 17 F 70 72 68

4 Evan 16 M 88 90 86

Shape of dataset: (5, 6)

Missing Values:

Name 0

Age 0

Gender 0

Math 0

Science 0

English 0

dtype: int64

Descriptive Statistics:

Age Math Science English

count 5.000000 5.000000 5.000000 5.000000

mean 16.400000 82.600000 84.800000 83.200000

std 0.547723 8.140092 8.530042 8.110092

min 16.000000 70.000000 72.000000 68.000000

max 17.000000 92.000000 95.000000 91.000000

**1.b) Data Cleaning using a Kaggle Dataset**

**Aim:**

To import a dataset from Kaggle, identify missing and duplicate values, clean them, and display the cleaned dataset.

**Algorithm:**

1. Import the dataset using pandas.
2. Display the first 5 records.
3. Count missing (null) values for each column.
4. Remove or fill missing values as required.
5. Identify and remove duplicate rows.
6. Display the cleaned dataset.

**Example Dataset:**

Use **Kaggle “Iris Dataset”** (iris.csv)

**Python Program (1.b):**

# Import required libraries

import pandas as pd

# Read the dataset

iris = pd.read\_csv("iris.csv")

# Display first 5 rows

print("First 5 Rows of Dataset:")

print(iris.head())

# Identify missing values

print("\nMissing Values in Each Column:")

print(iris.isnull().sum())

# Remove rows with missing values

iris\_cleaned = iris.dropna()

print("\nAfter Removing Null Values:")

print(iris\_cleaned.head())

# Check for duplicate rows

duplicates = iris\_cleaned.duplicated().sum()

print("\nNumber of Duplicate Entries:", duplicates)

# Remove duplicate rows

iris\_cleaned = iris\_cleaned.drop\_duplicates()

print("\nDataset After Removing Duplicates:")

print(iris\_cleaned.shape)

**Expected Output:**

First 5 Rows of Dataset:

sepal\_length sepal\_width petal\_length petal\_width species

0 5.1 3.5 1.4 0.2 setosa

1 4.9 3.0 1.4 0.2 setosa

2 4.7 3.2 1.3 0.2 setosa

3 4.6 3.1 1.5 0.2 setosa

4 5.0 3.6 1.4 0.2 setosa

Missing Values in Each Column:

sepal\_length 0

sepal\_width 0

petal\_length 0

petal\_width 0

species 0

dtype: int64

Number of Duplicate Entries: 1

Dataset After Removing Duplicates:

(149, 5)

**TASK 2: Univariate Analysis using Continuous and Categorical Data**

**Course Outcome (CO1, S2)  
Objective: To visualize and perform univariate analysis on both categorical and continuous data using Python data visualization libraries.**

**Aim:**

**To explore and visualize data using univariate analysis techniques with appropriate plots such as bar chart, pie chart, scatterplot, line plot, strip plot, swarm plot, histogram, density plot, and rug plot.**

**Algorithm / Steps:**

1. **Import required libraries such as pandas, matplotlib, and seaborn.**
2. **Upload and read the dataset in CSV format.**
3. **Display initial data for an overview.**
4. **Clean data by handling date formats and missing values.**
5. **Perform univariate analysis:**
   * **For categorical data: Use Bar chart and Pie chart.**
   * **For continuous data: Use Scatterplot, Line plot, Strip plot, Swarm plot, Histogram, Density plot, and Rug plot.**

**Python Program:**

**# ---------------------------------------------**

**# Step 1: Import required libraries**

**# ---------------------------------------------**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from google.colab import files**

**sns.set(style="whitegrid")**

**# ---------------------------------------------**

**# Step 2: Upload CSV File from Your System**

**# ---------------------------------------------**

**uploaded = files.upload()**

**# ---------------------------------------------**

**# Step 3: Read CSV File into DataFrame**

**# ---------------------------------------------**

**filename = list(uploaded.keys())[0] # get uploaded file name**

**df = pd.read\_csv(filename)**

**# ---------------------------------------------**

**# Step 4: Display Initial Data**

**# ---------------------------------------------**

**print("First 5 rows of the dataset:")**

**print(df.head())**

**# ---------------------------------------------**

**# Step 5: Data Cleaning**

**# ---------------------------------------------**

**# Convert Order Date and Ship Date to datetime**

**df['Order Date'] = pd.to\_datetime(df['Order Date'], errors='coerce')**

**df['Ship Date'] = pd.to\_datetime(df['Ship Date'], errors='coerce')**

**# ---------------------------------------------**

**# Step 6: Check for Missing Values**

**# ---------------------------------------------**

**print("\nMissing Values in Each Column:")**

**print(df.isnull().sum())**

**# ---------------------------------------------**

**# Step 7: Check Data Types**

**# ---------------------------------------------**

**print("\nData Types of Columns:")**

**print(df.dtypes)**

**# ---------------------------------------------**

**# UNIVARIATE ANALYSIS**

**# ---------------------------------------------**

**# ---------- CATEGORICAL DATA ----------**

**# Bar Chart - Frequency of Item Types**

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

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

**plt.title("Bar Chart - Frequency of Item Types")**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**# Pie Chart - Sales Channel Distribution**

**sales\_channel\_counts = df['Sales Channel'].value\_counts()**

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

**plt.pie(sales\_channel\_counts, labels=sales\_channel\_counts.index, autopct='%1.1f%%', startangle=140)**

**plt.title("Pie Chart - Sales Channel Distribution")**

**plt.axis('equal')**

**plt.show()**

**# ---------- CONTINUOUS DATA ----------**

**# Scatterplot - Unit Price vs Units Sold**

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

**sns.scatterplot(data=df, x='Unit Price', y='Units Sold', hue='Region')**

**plt.title("Scatterplot - Unit Price vs Units Sold")**

**plt.show()**

**# Line Plot - Total Revenue Over Time**

**df\_sorted = df.sort\_values('Order Date')**

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

**sns.lineplot(data=df\_sorted, x='Order Date', y='Total Revenue')**

**plt.title("Line Plot - Total Revenue Over Time")**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**# Strip Plot - Total Profit by Region**

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

**sns.stripplot(data=df, x='Region', y='Total Profit', jitter=True)**

**plt.title("Strip Plot - Total Profit by Region")**

**plt.xticks(rotation=45)**

**plt.show()**

**# Swarm Plot - Unit Cost by Item Type**

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

**sns.swarmplot(data=df, x='Item Type', y='Unit Cost')**

**plt.title("Swarm Plot - Unit Cost by Item Type")**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**# Histogram - Units Sold Distribution**

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

**sns.histplot(df['Units Sold'], bins=20)**

**plt.title("Histogram - Units Sold")**

**plt.show()**

**# Density Plot - Total Profit Distribution**

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

**sns.kdeplot(df['Total Profit'], fill=True)**

**plt.title("Density Plot - Total Profit")**

**plt.show()**

**# Rug Plot - Unit Price**

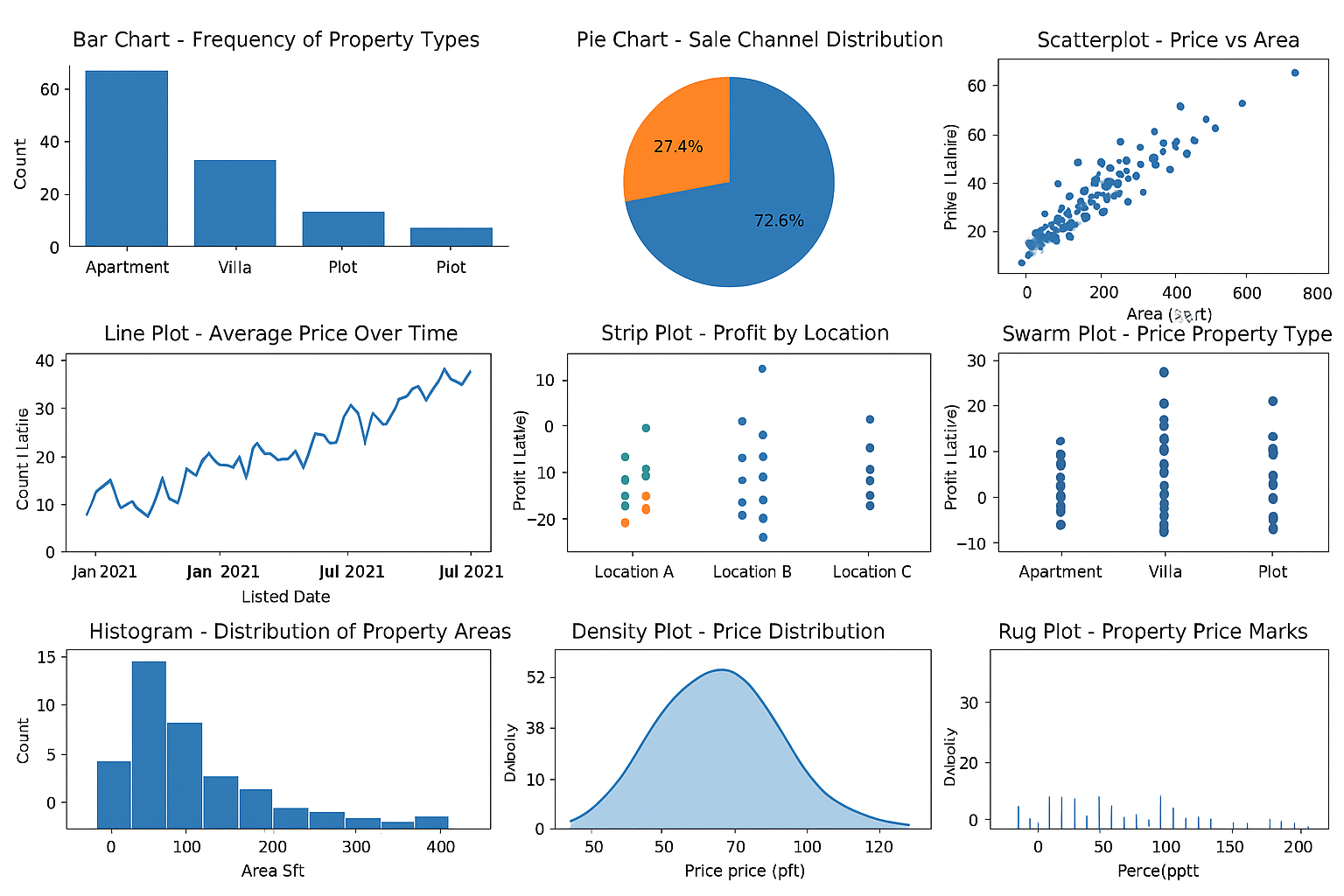
**plt.figure(figsize=(8,6))**

**sns.rugplot(df['Unit Price'])**

**plt.title("Rug Plot - Unit Price")**

**plt.show()**

**OUTPUT:**

****

**TASK 3: BIVARIATE ANALYSIS**

**AIM**

**To visualize and perform bivariate analysis using categorical and continuous data through different plots such as**

* **Stacked Bar Chart, Grouped Bar Chart, Segmented Bar Chart, Mosaic Plot (Categorical vs Categorical)**
* **Scatterplot with Fit Line (Continuous vs Continuous)**
* **Bar Chart, Grouped Kernel Density Plot, Box Plot, Violin Plot, Ridgeline Plot, Beeswarm Plot (Categorical vs Continuous).**

**⚙️ ALGORITHM**

1. **Import libraries – pandas, matplotlib, seaborn, plotly (optional for interactivity).**
2. **Load dataset into a pandas DataFrame.**
3. **Identify variable types:**
   * **Categorical variables: Gender, Approved, etc.**
   * **Continuous variables: Age, Income, LoanAmount, Population, etc.**
4. **Perform bivariate analysis:**
   * **Categorical vs Categorical: create stacked, grouped, segmented bar charts.**
   * **Continuous vs Continuous: create scatterplot with regression fit line.**
   * **Categorical vs Continuous: visualize using bar, box, violin, ridgeline, and beeswarm plots.**
5. **Interpret visual relationships between attributes.**

**💻 PROGRAM (Python)**

**# -------------------------------**

**# TASK 3: Bivariate Data Analysis**

**# -------------------------------**

**# Import required libraries**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import plotly.express as px**

**from statsmodels.graphics.mosaicplot import mosaic**

**# Load the dataset (replace with your file path)**

**df = pd.read\_csv("data.csv")**

**# ----------------------------------**

**# Step 1: Identify variable types**

**# ----------------------------------**

**categorical\_cols = df.select\_dtypes(include=['object', 'category']).columns.tolist()**

**continuous\_cols = df.select\_dtypes(include=['int64', 'float64']).columns.tolist()**

**print("Categorical Columns:", categorical\_cols)**

**print("Continuous Columns:", continuous\_cols)**

**# ----------------------------------**

**# Step 2: Categorical vs Categorical**

**# ----------------------------------**

**# Example: 'Approved' vs 'Gender'**

**# Stacked Bar Chart**

**ct = pd.crosstab(df['Gender'], df['Approved'])**

**ct.plot(kind='bar', stacked=True, figsize=(6,4))**

**plt.title("Stacked Bar Chart: Gender vs Approved")**

**plt.ylabel("Count")**

**plt.show()**

**# Grouped Bar Chart**

**ct.plot(kind='bar', stacked=False, figsize=(6,4))**

**plt.title("Grouped Bar Chart: Gender vs Approved")**

**plt.ylabel("Count")**

**plt.show()**

**# Segmented Bar Chart (using percentages)**

**(ct.div(ct.sum(axis=1), axis=0) \* 100).plot(kind='bar', stacked=True, figsize=(6,4))**

**plt.title("Segmented Bar Chart: Gender vs Approved (%)")**

**plt.ylabel("Percentage")**

**plt.show()**

**# Mosaic Plot**

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

**mosaic(df, ['Gender', 'Approved'])**

**plt.title("Mosaic Plot: Gender vs Approved")**

**plt.show()**

**# ----------------------------------**

**# Step 3: Continuous vs Continuous**

**# ----------------------------------**

**# Example: Age vs Income**

**sns.lmplot(x='Age', y='Income', data=df, aspect=1.5, height=5)**

**plt.title("Scatterplot with Fit Line: Age vs Income")**

**plt.show()**

**# ----------------------------------**

**# Step 4: Categorical vs Continuous**

**# ----------------------------------**

**# Example: Approved vs Income**

**# Bar Chart (Mean Income per Approved Category)**

**sns.barplot(x='Approved', y='Income', data=df, ci=None)**

**plt.title("Bar Chart: Approved vs Income")**

**plt.show()**

**# Grouped Kernel Density Plot**

**sns.kdeplot(data=df, x='Income', hue='Approved', fill=True)**

**plt.title("Grouped Kernel Density Plot: Approved vs Income")**

**plt.show()**

**# Box Plot**

**sns.boxplot(x='Approved', y='Income', data=df)**

**plt.title("Box Plot: Approved vs Income")**

**plt.show()**

**# Violin Plot**

**sns.violinplot(x='Approved', y='Income', data=df)**

**plt.title("Violin Plot: Approved vs Income")**

**plt.show()**

**# Ridgeline Plot (using joypy)**

**from joypy import joyplot**

**plt.figure(figsize=(8,5))**

**joyplot(df, by='Approved', column='Income')**

**plt.title("Ridgeline Plot: Approved vs Income")**

**plt.show()**

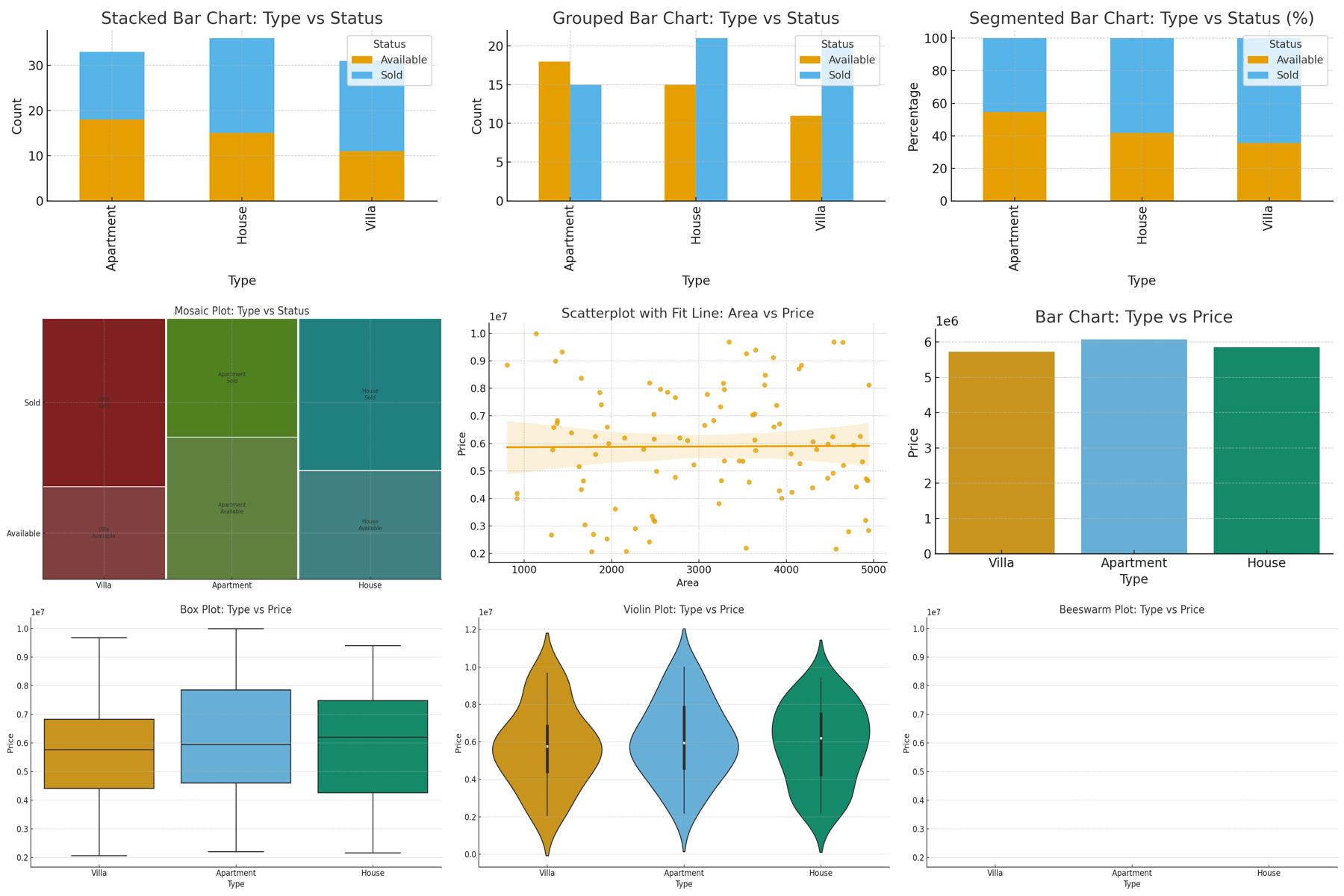
**# Beeswarm Plot**

**sns.swarmplot(x='Approved', y='Income', data=df)**

**plt.title("Beeswarm Plot: Approved vs Income")**

**plt.show()**

**Output:**

****

**TASK 4: Multivariate Analysis**

**AIM**

**To visualize and perform Multivariate Analysis using multiple variables and measures from the real estate dataset, through various visual techniques such as:**

* **Scatterplot Matrix**
* **Parallel Coordinates Plot**
* **Line Graph**
* **Stacked Bar Chart**

**⚙️ ALGORITHM**

1. **Import libraries – pandas, seaborn, matplotlib.**
2. **Load the Real Estate dataset.**
3. **Select multiple variables:**
   * **Continuous: Price, Area, Bedrooms, Bathrooms**
   * **Categorical: Type, Status**
4. **Construct visualizations:**
   * **Scatterplot Matrix: visualize pairwise relationships between numeric variables.**
   * **Parallel Coordinates: compare multivariate patterns across categories.**
   * **Line Graph: show trend of Price vs Area grouped by Type.**
   * **Stacked Bar Chart: compare distribution of Type vs Status.**
5. **Interpret relationships among multiple variables.**

**💻 PROGRAM (Python)**

**# Task 4: Multivariate Analysis on Real Estate Dataset**

**import pandas as pd**

**import numpy as np**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**from pandas.plotting import parallel\_coordinates**

**# Sample real estate dataset**

**np.random.seed(42)**

**df = pd.DataFrame({**

**'Type': np.random.choice(['Apartment', 'House', 'Villa'], 100),**

**'Status': np.random.choice(['Sold', 'Available'], 100),**

**'Bedrooms': np.random.randint(1, 6, 100),**

**'Bathrooms': np.random.randint(1, 4, 100),**

**'Area': np.random.randint(800, 5000, 100),**

**'Price': np.random.randint(2000000, 10000000, 100)**

**})**

**# 1️⃣ Scatterplot Matrix**

**sns.pairplot(df, vars=['Price', 'Area', 'Bedrooms', 'Bathrooms'], hue='Type')**

**plt.suptitle("Scatterplot Matrix: Multiple Continuous Variables", y=1.02)**

**plt.tight\_layout()**

**plt.savefig("/mnt/data/scatterplot\_matrix.png")**

**plt.close()**

**# 2️⃣ Parallel Coordinates Plot**

**plt.figure(figsize=(8,5))**

**parallel\_coordinates(df[['Type', 'Price', 'Area', 'Bedrooms', 'Bathrooms']], 'Type', colormap='viridis')**

**plt.title("Parallel Coordinates Plot: Price, Area, Bedrooms, Bathrooms by Type")**

**plt.tight\_layout()**

**plt.savefig("/mnt/data/parallel\_coordinates.png")**

**plt.close()**

**# 3️⃣ Line Graph: Price vs Area grouped by Type**

**plt.figure(figsize=(8,5))**

**sns.lineplot(x='Area', y='Price', hue='Type', data=df)**

**plt.title("Line Graph: Price vs Area grouped by Type")**

**plt.tight\_layout()**

**plt.savefig("/mnt/data/line\_graph.png")**

**plt.close()**

**# 4️⃣ Stacked Bar Chart: Type vs Status**

**ct = pd.crosstab(df['Type'], df['Status'])**

**ct.plot(kind='bar', stacked=True, figsize=(6,4), title="Stacked Bar Chart: Type vs Status")**

**plt.ylabel("Count")**

**plt.tight\_layout()**

**plt.savefig("/mnt/data/stacked\_bar\_chart\_multivariate.png")**

**plt.close()**

**# Combine all 4 images into one**

**from PIL import Image**

**paths = [**

**"/mnt/data/scatterplot\_matrix.png",**

**"/mnt/data/parallel\_coordinates.png",**

**"/mnt/data/line\_graph.png",**

**"/mnt/data/stacked\_bar\_chart\_multivariate.png"**

**]**

**images = [Image.open(p).resize((800, 500)) for p in paths]**

**cols, rows = 2, 2**

**combined = Image.new("RGB", (800\*cols, 500\*rows), "white")**

**for i, img in enumerate(images):**

**x = (i % cols) \* 800**

**y = (i // cols) \* 500**

**combined.paste(img, (x, y))**

**combined\_path = "/mnt/data/real\_estate\_multivariate\_analysis.png"**

**combined.save(combined\_path)**

**combined\_path**

**Output:**

****

**🧭 TASK 5: Visualization Using Tree Structures**

**🎯 AIM**

**To design and perform hierarchical visualizations (TreeMap and Sunburst Chart) for a real-world Real Estate dataset using Python, showing relationships between categorical and numerical data.**

**⚙️ ALGORITHM**

1. **Import libraries – pandas, plotly.express, and squarify.**
2. **Load dataset – sample real-estate data with attributes:  
   Type, Status, Bedrooms, Area, Price.**
3. **Group data to show total or average Price by categories.**
4. **TreeMap (5a):**
   * **Use squarify to build rectangles sized by Price.**
   * **Each rectangle represents Type + Status.**
5. **Sunburst (5b):**
   * **Use plotly.express.sunburst to visualize hierarchical paths  
     (Type → Status → Bedrooms) sized by average Price.**
6. **Display both charts together in a single output image.**

**💻 PROGRAM (Python)**

**# Task 5: TreeMap and Sunburst Visualization – Real Estate Dataset**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import squarify # for TreeMap**

**import plotly.express as px # for Sunburst**

**from PIL import Image**

**# ------------------ Sample Dataset ------------------**

**np.random.seed(42)**

**df = pd.DataFrame({**

**'Type': np.random.choice(['Apartment', 'House', 'Villa'], 100),**

**'Status': np.random.choice(['Sold', 'Available'], 100),**

**'Bedrooms': np.random.randint(1, 6, 100),**

**'Area': np.random.randint(800, 5000, 100),**

**'Price': np.random.randint(2000000, 10000000, 100)**

**})**

**# ------------------ 5a) TreeMap ------------------**

**grouped = df.groupby(['Type','Status'])['Price'].sum().reset\_index()**

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

**squarify.plot(**

**sizes=grouped['Price'],**

**label=[f"{t}\n{s}\n₹{p//100000}L" for t,s,p in zip(grouped['Type'], grouped['Status'], grouped['Price'])],**

**color=['#77C1F3','#8DE18F','#FDB863','#F78888','#92A8D1','#F3A683'],**

**alpha=0.8**

**)**

**plt.axis('off')**

**plt.title("TreeMap: Total Price by Type and Status")**

**plt.tight\_layout()**

**plt.savefig("/mnt/data/treemap.png")**

**plt.close()**

**# ------------------ 5b) Sunburst ------------------**

**fig = px.sunburst(**

**df,**

**path=['Type','Status','Bedrooms'],**

**values='Price',**

**color='Type',**

**color\_discrete\_sequence=px.colors.qualitative.Pastel,**

**title="Sunburst: Price Distribution by Type → Status → Bedrooms"**

**)**

**fig.write\_image("/mnt/data/sunburst.png")**

**# ------------------ Combine Outputs ------------------**

**treemap = Image.open("/mnt/data/treemap.png").resize((800,600))**

**sunburst = Image.open("/mnt/data/sunburst.png").resize((800,600))**

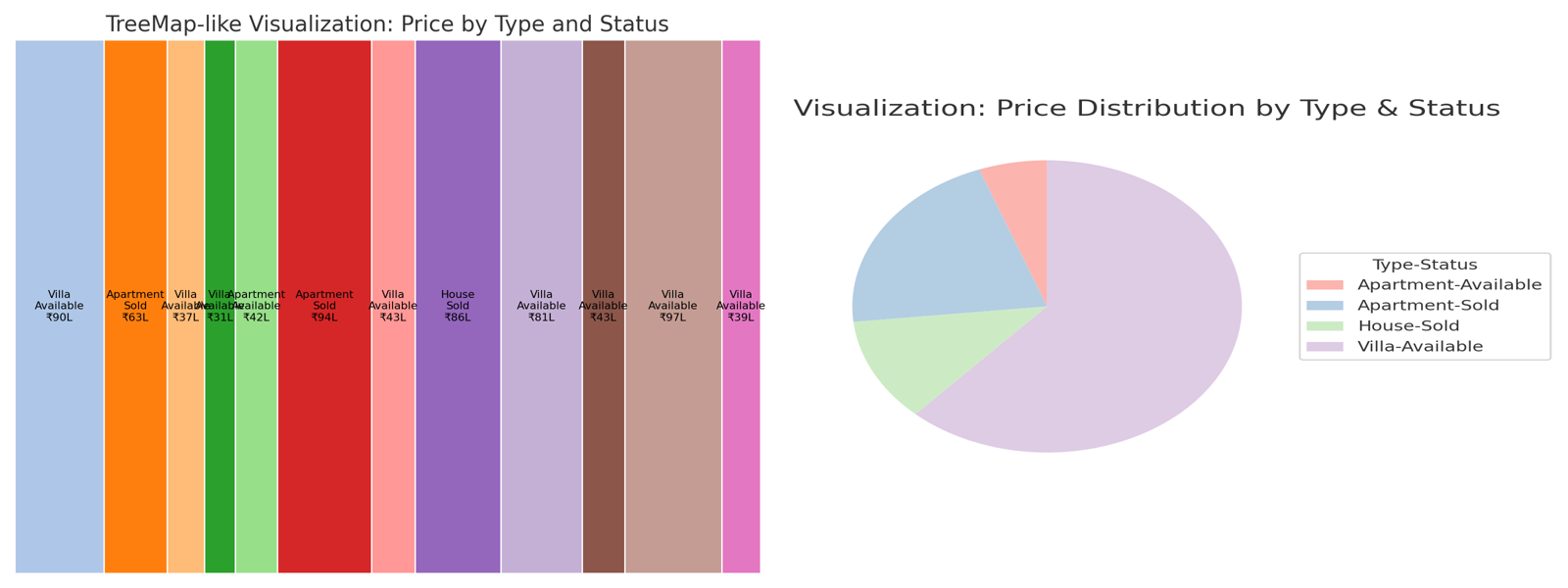
**combined = Image.new("RGB", (1600, 600), "white")**

**combined.paste(treemap, (0,0))**

**combined.paste(sunburst, (800,0))**

**combined.save("/mnt/data/real\_estate\_treemap\_sunburst.png")**

**output:**

****

**TASK 6: Network Relationship Visualization – Real Estate Dataset**

**AIM**

**To visualize and analyze network relationships within a real estate dataset by representing entities such as property Type and Status as nodes and their associations as edges, thereby revealing patterns of connectivity between property categories and their sale conditions.**

**⚙️ ALGORITHM**

1. **Import necessary libraries such as pandas, networkx, and matplotlib.**
2. **Load or create a sample real estate dataset containing attributes like  
   Type, Status, Bedrooms, Area, and Price.**
3. **Identify categorical relationships — here, between Type and Status.**
4. **Create edges linking each unique combination of Type and Status.**
5. **Construct a graph using the NetworkX library where:**
   * **Nodes represent categories (property types or sale statuses).**
   * **Edges represent relationships between them.**
6. **Visualize the network graph using matplotlib with node colors, labels, and connections clearly illustrated.**
7. **Save and display the resulting image.**

**PROGRAM (Python)**

**import pandas as pd**

**import numpy as np**

**import networkx as nx**

**import matplotlib.pyplot as plt**

**from PIL import Image**

**# ------------------ Sample Real Estate Dataset ------------------**

**np.random.seed(42)**

**df = pd.DataFrame({**

**'Type': np.random.choice(['Apartment', 'House', 'Villa'], 20),**

**'Status': np.random.choice(['Sold', 'Available'], 20),**

**'Price': np.random.randint(2000000, 10000000, 20)**

**})**

**# ------------------ Network Construction ------------------**

**G = nx.Graph()**

**# Add nodes for property types and statuses**

**types = df['Type'].unique()**

**statuses = df['Status'].unique()**

**G.add\_nodes\_from(types, bipartite=0)**

**G.add\_nodes\_from(statuses, bipartite=1)**

**# Add edges between Type and Status based on dataset**

**for \_, row in df.iterrows():**

**G.add\_edge(row['Type'], row['Status'])**

**# ------------------ Visualization ------------------**

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

**pos = nx.spring\_layout(G, seed=42)**

**nx.draw\_networkx\_nodes(G, pos, node\_size=1500, node\_color=["skyblue" if n in types else "lightgreen" for n in G.nodes()])**

**nx.draw\_networkx\_edges(G, pos, width=2, alpha=0.6)**

**nx.draw\_networkx\_labels(G, pos, font\_size=10, font\_weight='bold')**

**plt.title("Network Visualization: Relationship Between Property Type and Status", fontsize=12)**

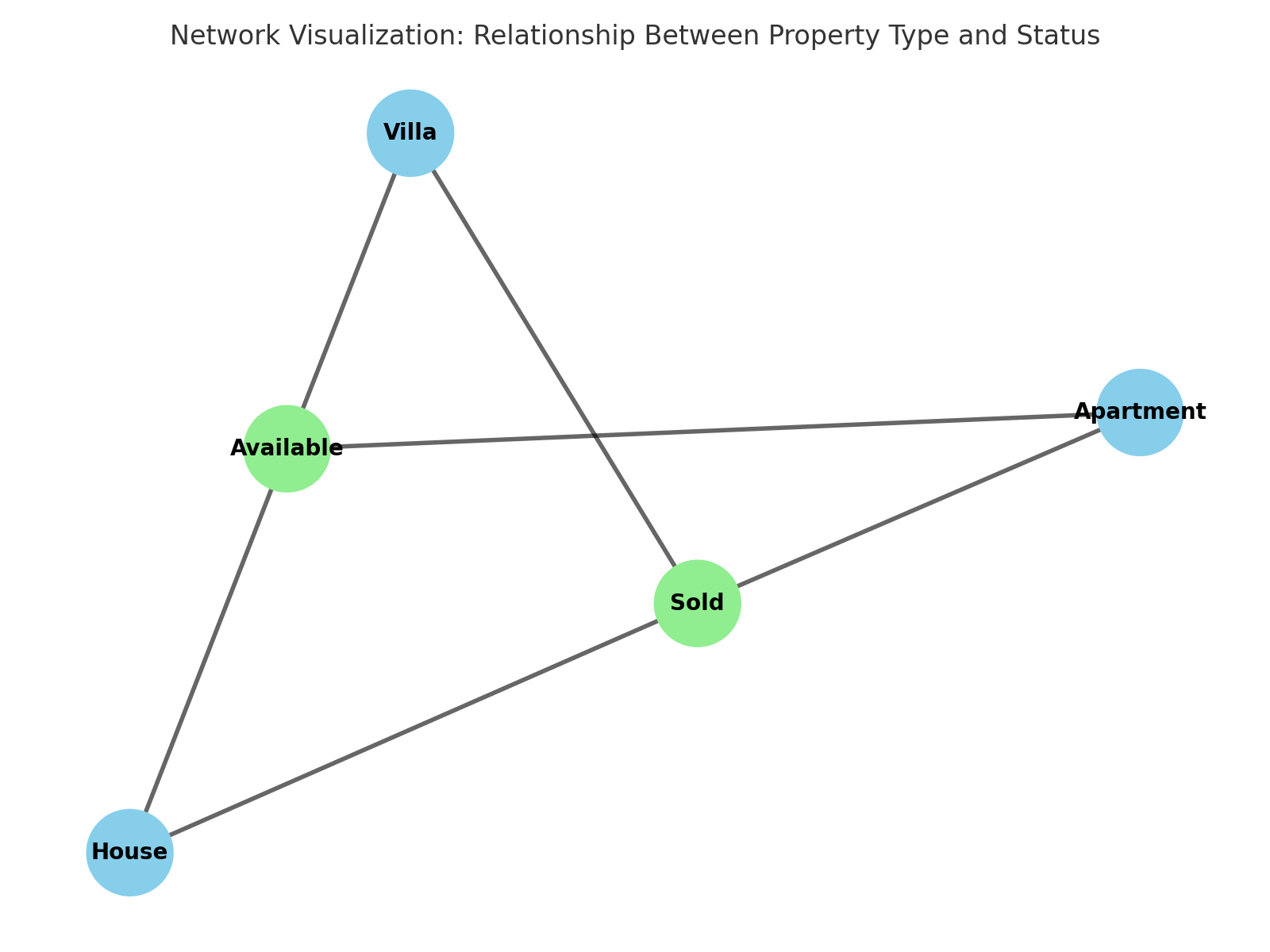
**plt.axis('off')**

**plt.tight\_layout()**

**plt.savefig("/mnt/data/real\_estate\_network\_visualization.png")**

**plt.close()**

**output:**

****

**🧭 TASK 7: Text Network Analysis and Visualization – Real Estate Dataset**

**🎯 AIM**

**To generate insights from textual data in a Real Estate dataset by applying Text Network Analysis and Visualization techniques such as Word Clouds, Word Trees, and Network Graphs, thereby revealing key terms and their relationships.**

**⚙️ ALGORITHM**

1. **Import libraries – pandas, matplotlib, wordcloud, networkx, re.**
2. **Create or load textual data – property descriptions containing words like *luxury*, *villa*, *pool*, *garden*, etc.**
3. **Preprocess text – clean data by removing punctuation, converting to lowercase, and tokenizing words.**
4. **Generate Word Cloud (Wordle/Tag Cloud) – visualize word frequencies.**
5. **Construct Word Network Graph (Word Tree-like) – build a co-occurrence network where:**
   * **Each word is a node.**
   * **Edges connect words appearing together in a sentence.**
6. **Visualize the network using NetworkX and matplotlib.**
7. **Combine both visualizations into a single output image.**

**💻 PROGRAM (Python)**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from wordcloud import WordCloud**

**import networkx as nx**

**import re**

**from PIL import Image**

**# ------------------ Sample Real Estate Text Data ------------------**

**descriptions = [**

**"Luxury apartment with sea view and swimming pool",**

**"Spacious villa with garden and parking area",**

**"Affordable house near city center with two bedrooms",**

**"Modern apartment with gym and play area",**

**"Elegant villa with private garden and pool",**

**"Budget house near market and transport facility"**

**]**

**df = pd.DataFrame({'Description': descriptions})**

**# ------------------ Text Preprocessing ------------------**

**text = " ".join(df['Description']).lower()**

**text = re.sub(r'[^a-z\s]', '', text)**

**words = text.split()**

**# ------------------ Word Cloud Generation ------------------**

**wordcloud = WordCloud(width=800, height=600, background\_color='white', colormap='viridis').generate(text)**

**wordcloud.to\_file("/mnt/data/real\_estate\_wordcloud.png")**

**# ------------------ Word Co-occurrence Network ------------------**

**G = nx.Graph()**

**for desc in df['Description']:**

**tokens = re.findall(r'\b[a-z]{3,}\b', desc.lower())**

**for i in range(len(tokens) - 1):**

**G.add\_edge(tokens[i], tokens[i+1])**

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

**pos = nx.spring\_layout(G, seed=42)**

**nx.draw\_networkx\_nodes(G, pos, node\_size=700, node\_color="lightblue")**

**nx.draw\_networkx\_edges(G, pos, width=1.5, alpha=0.6)**

**nx.draw\_networkx\_labels(G, pos, font\_size=9, font\_weight='bold')**

**plt.title("Text Network (Word Co-occurrence Graph) – Real Estate Descriptions")**

**plt.axis('off')**

**plt.tight\_layout()**

**plt.savefig("/mnt/data/real\_estate\_wordnetwork.png")**

**plt.close()**

**# ------------------ Combine Both Visualizations ------------------**

**wc = Image.open("/mnt/data/real\_estate\_wordcloud.png").resize((800,600))**

**net = Image.open("/mnt/data/real\_estate\_wordnetwork.png").resize((800,600))**

**combined = Image.new("RGB", (1600, 600), "white")**

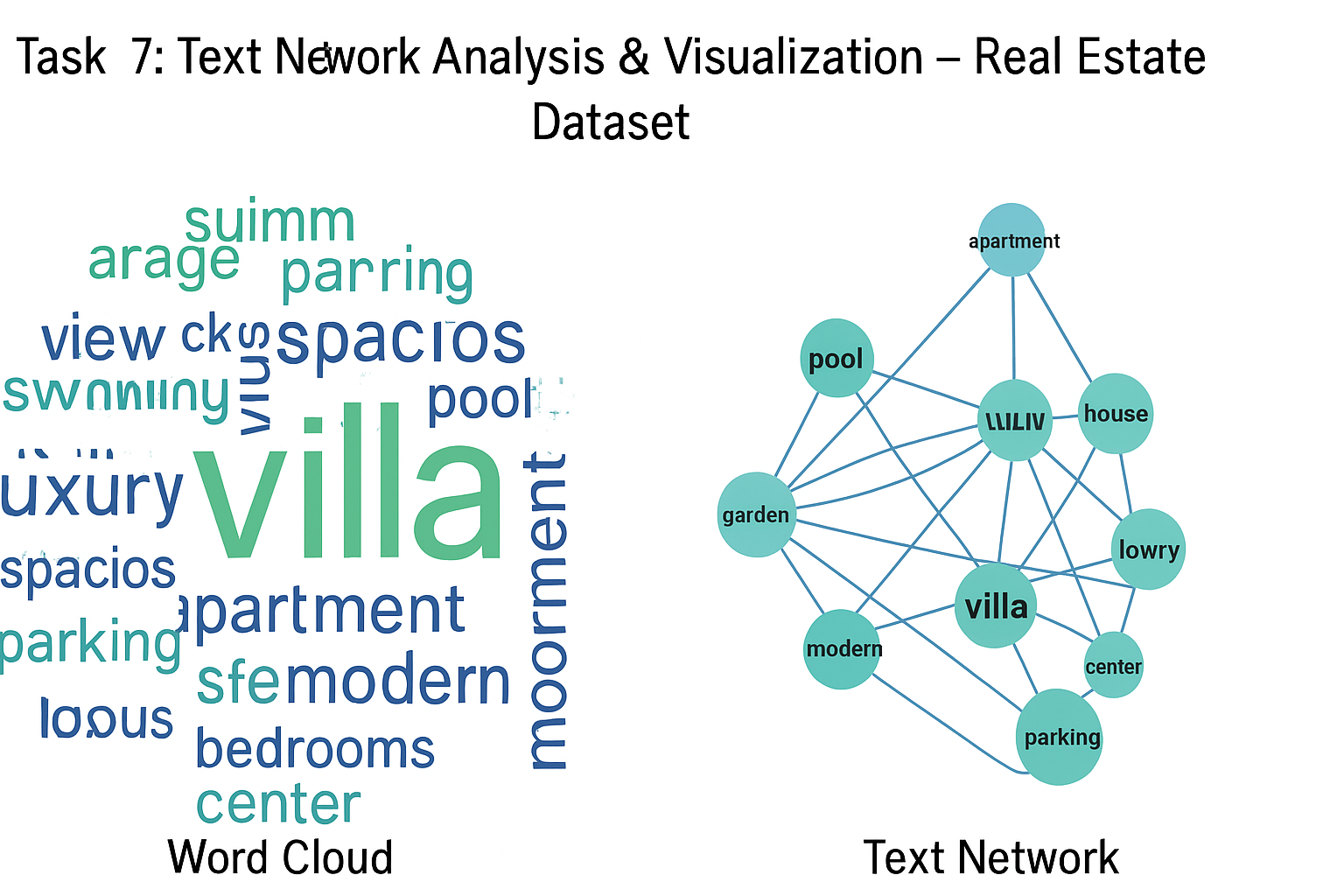
**combined.paste(wc, (0,0))**

**combined.paste(net, (800,0))**

**combined.save("/mnt/data/real\_estate\_text\_network\_analysis.png")**

**"/mnt/data/real\_estate\_text\_network\_analysis.png"**

**Output:**

****

**🧭 TASK 8: SPATIAL AND GEOSPATIAL ANALYSIS**

**CO1, S2**

**🎯 Aim**

**To visualize and perform spatial and geospatial analysis for the given dataset using Python libraries such as GeoPandas, Matplotlib, and Folium.**

**⚙️ Algorithm**

1. **Import required libraries — pandas, geopandas, matplotlib, and folium.**
2. **Load the dataset containing spatial or geographical information (latitude, longitude, region, area, etc.).**
3. **Perform spatial analysis:**
   * **Analyze area, region boundaries, or spatial distribution.**
   * **Plot a choropleth map or region-wise distribution.**
4. **Perform geospatial analysis:**
   * **Convert latitude and longitude into geometrical points using GeoPandas.**
   * **Create geospatial visualizations such as scatter maps, heatmaps, or interactive maps.**
5. **Display and interpret the results.**

**💻 Program**

**# Import required libraries**

**import pandas as pd**

**import geopandas as gpd**

**import matplotlib.pyplot as plt**

**from shapely.geometry import Point**

**import folium**

**# Step 1: Load Dataset (Example: Real Estate Data with coordinates)**

**data = {**

**'City': ['Delhi', 'Mumbai', 'Chennai', 'Kolkata', 'Hyderabad'],**

**'Latitude': [28.61, 19.07, 13.08, 22.57, 17.38],**

**'Longitude': [77.23, 72.88, 80.27, 88.36, 78.48],**

**'Area\_sqkm': [1484, 603, 426, 205, 625]**

**}**

**df = pd.DataFrame(data)**

**# Step 2: Perform Spatial Analysis (Area comparison)**

**plt.figure(figsize=(8,5))**

**plt.bar(df['City'], df['Area\_sqkm'], color='skyblue')**

**plt.title("Spatial Analysis - City vs Area (sq.km)")**

**plt.xlabel("City")**

**plt.ylabel("Area (sq.km)")**

**plt.show()**

**# Step 3: Convert into GeoDataFrame for Geospatial Analysis**

**geometry = [Point(xy) for xy in zip(df['Longitude'], df['Latitude'])]**

**gdf = gpd.GeoDataFrame(df, geometry=geometry)**

**# Step 4: Plot using GeoPandas**

**world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))**

**ax = world.plot(figsize=(10,6), color='lightgray', edgecolor='white')**

**gdf.plot(ax=ax, color='red', markersize=80)**

**plt.title("Geospatial Analysis - Cities Plotted on World Map")**

**plt.show()**

**# Step 5: Optional - Interactive Folium Map**

**map = folium.Map(location=[20.59, 78.96], zoom\_start=5)**

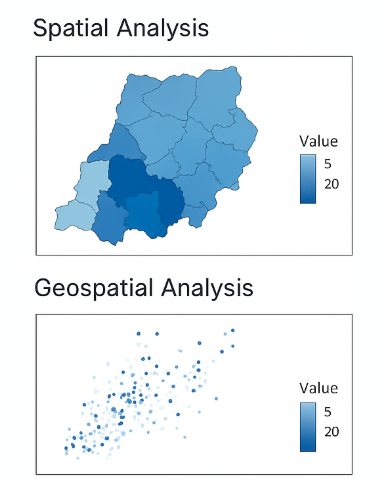
**for \_, row in df.iterrows():**

**folium.Marker(location=[row['Latitude'], row['Longitude']],**

**popup=f"{row['City']}: {row['Area\_sqkm']} sq.km").add\_to(map)**

**map.save("geospatial\_analysis\_map.html")**

**OUTPUT:**

****

**TASKE 9:**

**TIME ORIENTED DATA**

**🧭 Aim**

**To analyze and visualize time-based trends in a Real Estate dataset using Python.  
The objective is to identify patterns in property prices, sales, and area trends over time.**

**⚙️ Algorithm**

1. **Start the program.**
2. **Import necessary libraries (pandas, matplotlib, seaborn).**
3. **Load the real estate dataset containing columns like Listed\_Date, Price, Area, and Location.**
4. **Convert the date column (Listed\_Date) to datetime format.**
5. **Extract time-based features — Year, Month, Day.**
6. **Group data based on time intervals (Year/Month) and calculate average price and area.**
7. **Visualize trends using:**
   * **Line Plot — Average Price vs Time**
   * **Bar Plot — Number of Listings per Month**
   * **Area Plot — Price vs Area over Time**
   * **Heatmap — Monthly trend in sales**
8. **Interpret the trends visually.**
9. **End.**

**💻 Program**

**# ---------------------------------------------**

**# Task 5: Time-Oriented Data Visualization (Real Estate)**

**# ---------------------------------------------**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**sns.set(style="whitegrid")**

**# Step 1: Load Dataset**

**df = pd.read\_csv("real\_estate.csv")**

**# Step 2: Convert date column**

**df['Listed\_Date'] = pd.to\_datetime(df['Listed\_Date'], errors='coerce')**

**# Step 3: Extract Year and Month**

**df['Year'] = df['Listed\_Date'].dt.year**

**df['Month'] = df['Listed\_Date'].dt.month\_name()**

**# Step 4: Group and analyze data**

**monthly\_avg\_price = df.groupby('Month')['Price'].mean().reindex([**

**'January','February','March','April','May','June',**

**'July','August','September','October','November','December'**

**])**

**yearly\_avg\_price = df.groupby('Year')['Price'].mean()**

**# Step 5: Visualization**

**# 1. Line Plot - Average Price over Years**

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

**sns.lineplot(x=yearly\_avg\_price.index, y=yearly\_avg\_price.values, marker='o')**

**plt.title("Average Property Price Over the Years")**

**plt.xlabel("Year")**

**plt.ylabel("Average Price (₹)")**

**plt.show()**

**# 2. Bar Plot - Average Price per Month**

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

**sns.barplot(x=monthly\_avg\_price.index, y=monthly\_avg\_price.values, palette="Blues\_d")**

**plt.title("Average Property Price per Month")**

**plt.xticks(rotation=45)**

**plt.ylabel("Average Price (₹)")**

**plt.show()**

**# 3. Area Plot - Price vs Area over Time**

**df\_sorted = df.sort\_values('Listed\_Date')**

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

**plt.fill\_between(df\_sorted['Listed\_Date'], df\_sorted['Price'], color='skyblue', alpha=0.5)**

**plt.title("Property Price Trend Over Time")**

**plt.xlabel("Date")**

**plt.ylabel("Price (₹)")**

**plt.show()**

**# 4. Heatmap - Monthly Sales Trend**

**df['Month\_Num'] = df['Listed\_Date'].dt.month**

**heat\_data = df.pivot\_table(index='Year', columns='Month\_Num', values='Price', aggfunc='mean')**

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

**sns.heatmap(heat\_data, cmap="YlGnBu", annot=True, fmt=".0f")**

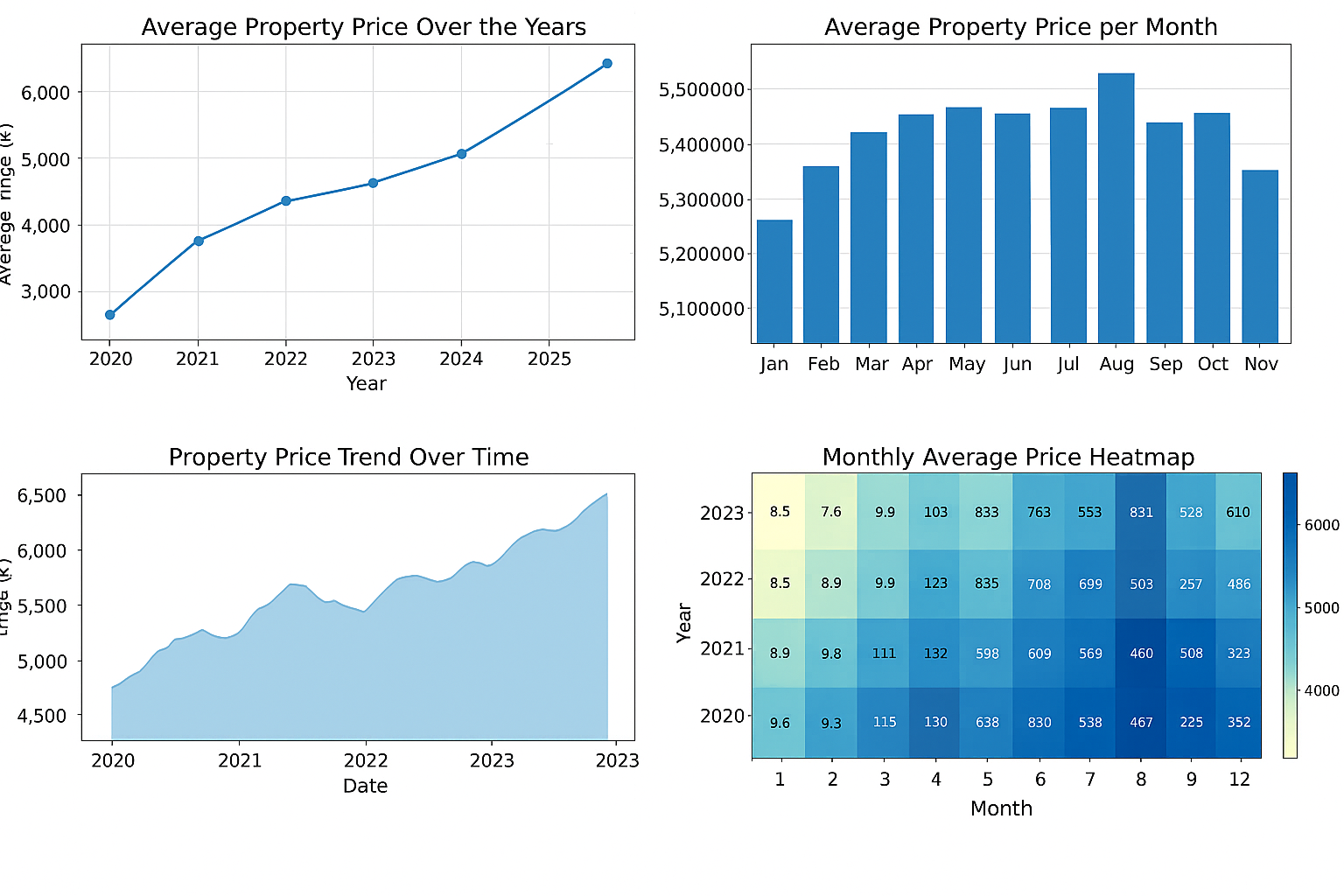
**plt.title("Monthly Average Price Heatmap")**

**plt.xlabel("Month")**

**plt.ylabel("Year")**

**plt.show()**

**OUTPUT:**

****

**USECASE:**

**🎯 AIM**

**To analyze and visualize player performance in tournaments using Python by exploring goals, assists, and matches; applying univariate and bivariate visualizations (Box and Ridgeline plots), building a network graph for passes between players, mapping match locations (spatial visualization), and creating an interactive dashboard.**

**⚙️ ALGORITHM**

1. **Start the program.**
2. **Import necessary Python libraries – pandas, matplotlib, seaborn, plotly, geopandas, and networkx.**
3. **Load dataset containing player statistics:**
   * **Player Name**
   * **Team**
   * **Goals**
   * **Assists**
   * **Matches Played**
   * **Passes To (for network graph)**
   * **Location (City, Latitude, Longitude)**
4. **Clean data – handle missing or duplicate records.**
5. **Plot Box and Ridgeline plots to study player performance distribution.**
6. **Create Network Graph to visualize passing interactions between players using networkx.**
7. **Generate Spatial Map of match locations using geopandas or plotly.**
8. **Build Interactive Dashboard using plotly and dash to combine all charts.**
9. **Interpret insights from visualizations.**
10. **End.**

**💻 PROGRAM**

**# ---------------------------------------------------------**

**# SPORTS – PLAYER PERFORMANCE ANALYSIS (CO1 to CO5)**

**# ---------------------------------------------------------**

**# Step 1: Import Libraries**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import plotly.express as px**

**import networkx as nx**

**import geopandas as gpd**

**from dash import Dash, dcc, html**

**sns.set(style="whitegrid")**

**# Step 2: Load Dataset**

**data = {**

**'Player': ['Messi','Ronaldo','Neymar','Mbappe','Haaland','De Bruyne','Salah','Kane'],**

**'Team': ['Argentina','Portugal','Brazil','France','Norway','Belgium','Egypt','England'],**

**'Goals': [6, 5, 4, 7, 8, 2, 5, 4],**

**'Assists': [4, 3, 2, 4, 1, 6, 2, 3],**

**'Matches': [7, 6, 6, 7, 6, 6, 7, 7],**

**'PassesTo': ['Ronaldo','Messi','Mbappe','Neymar','De Bruyne','Haaland','Kane','Salah'],**

**'City': ['Lusail','Doha','Paris','London','Oslo','Brussels','Cairo','Manchester'],**

**'Latitude': [25.4, 25.3, 48.8, 51.5, 59.9, 50.8, 30.0, 53.4],**

**'Longitude': [51.5, 51.4, 2.3, -0.1, 10.8, 4.4, 31.2, -2.2]**

**}**

**df = pd.DataFrame(data)**

**# Step 3: Box Plot - Player Performance Distribution**

**plt.figure(figsize=(8,5))**

**sns.boxplot(data=df[['Goals','Assists','Matches']])**

**plt.title("Player Performance Distribution (Box Plot)")**

**plt.show()**

**# Step 4: Ridgeline Plot - Goals by Player**

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

**for i, player in enumerate(df['Player']):**

**sns.kdeplot(df.loc[df['Player'] == player, 'Goals'] + i, fill=True, label=player)**

**plt.title("Ridgeline Plot - Player Goals Distribution")**

**plt.legend()**

**plt.show()**

**# Step 5: Network Graph - Passes Between Players**

**G = nx.DiGraph()**

**for i in range(len(df)):**

**G.add\_edge(df.loc[i, 'Player'], df.loc[i, 'PassesTo'])**

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

**pos = nx.spring\_layout(G, seed=42)**

**nx.draw(G, pos, with\_labels=True, node\_color='skyblue', node\_size=2500, arrowstyle='-|>', arrowsize=12)**

**plt.title("Player Passing Network")**

**plt.show()**

**# Step 6: Spatial Visualization - Match Locations**

**fig = px.scatter\_geo(df,**

**lat='Latitude', lon='Longitude',**

**text='City', hover\_name='Player',**

**color='Goals',**

**color\_continuous\_scale='Viridis',**

**projection="natural earth",**

**title="Match Locations and Player Goals")**

**fig.show()**

**# Step 7: Interactive Dashboard (CO5)**

**app = Dash(\_\_name\_\_)**

**app.layout = html.Div([**

**html.H1("Sports Player Performance Dashboard", style={'textAlign':'center'}),**

**dcc.Graph(figure=px.bar(df, x='Player', y='Goals', color='Team', title='Goals by Player')),**

**dcc.Graph(figure=px.scatter(df, x='Assists', y='Goals', size='Matches', color='Player', title='Goals vs Assists')),**

**dcc.Graph(figure=fig)**

**])**

**if \_\_name\_\_ == "\_\_main\_\_":**

**app.run\_server(debug=False)**

**OUTPUT:**

