**Framework for Data and Visual Analytics**

**Lab Experiments**

**Name: DEVESH D  
Roll No:231501034  
Department: Artificial Intelligence and Machine Learning  
Subject Name:** **Framework for Data and Visual Analytics  
Subject Code:** **AD23632**

**Experiment 1**

Setting up the Python environment and libraries-Juypter Notebook: code:

1. print("Hello, Jupyter Notebook!")

O/P:Hello, Jupyter Notebook!

1. import ipywidgets as widgets

display(slider)

button.on\_click(on\_click)

def on\_click(b):

print(f"Slider value is: {slider.value}")

button = widgets.Button(description="Click Me")

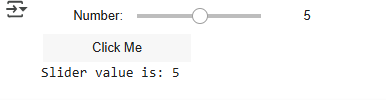
display(button)

slider = widgets.IntSlider(value=5, min=0, max=10, step=1,

description='Number:')

from IPython.display import display

# O/P:

****

**Experiment - 2**

**EDA-Data Import and Export:**

**Code:**

import pandas as pd

from sqlalchemy import create\_engine import sqlite3

import requests

import matplotlib.pyplot as plt

# Load CSV

df\_csv = pd.read\_csv("/content/gender\_submission.csv") print("CSV Data:\n", df\_csv.head())

# Export to Excel

excel\_path = "titanic\_full.xlsx" df\_csv.to\_excel(excel\_path, index=False)

print(f"Entire CSV exported to: {excel\_path}")

# Read Excel file and preview

df\_excel = pd.read\_excel("/content/titanic\_full.xlsx") print("Excel Data Preview:")

display(df\_excel.head()) # ✅ Now appears in a nice box

# SQL setup and query

engine = create\_engine("sqlite:///:memory:") df\_csv.to\_sql("titanic", engine, if\_exists="replace", index=False)

query = """

SELECT Survived, COUNT(\*) AS count FROM titanic

GROUP BY Survived """

df\_sql = pd.read\_sql\_query(query, engine) print("SQL Query Result:\n", df\_sql)

# Web scraping

url = "https://en.wikipedia.org/wiki/Titanic" try:

tables = pd.read\_html(url)

print("\nWeb Scraped Table Example:\n", tables[0].head()) except:

print("\nUnable to scrape table from the web.")

# Plot survival count

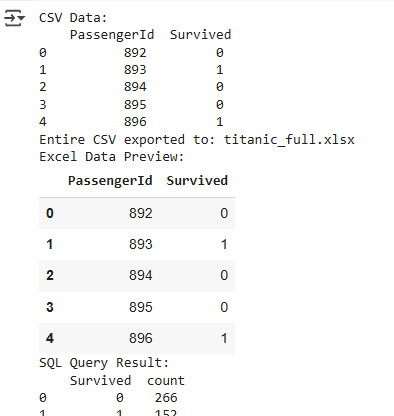
df\_sql.plot(kind='bar', x='Survived', y='count', legend=False, color=['red', 'green'])

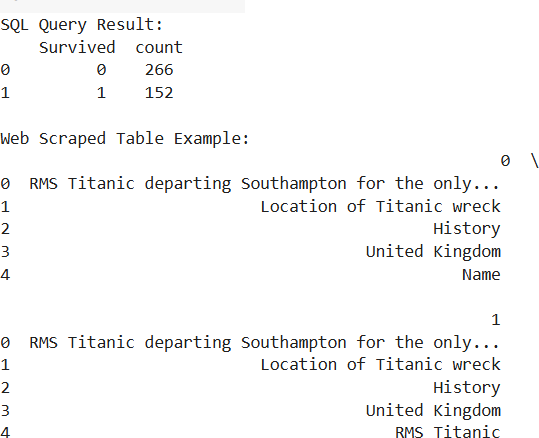
plt.title("Titanic Survival Count") plt.xlabel("Survived (0 = No, 1 = Yes)") plt.ylabel("Number of Passengers") plt.xticks(rotation=0) plt.grid(axis='y')

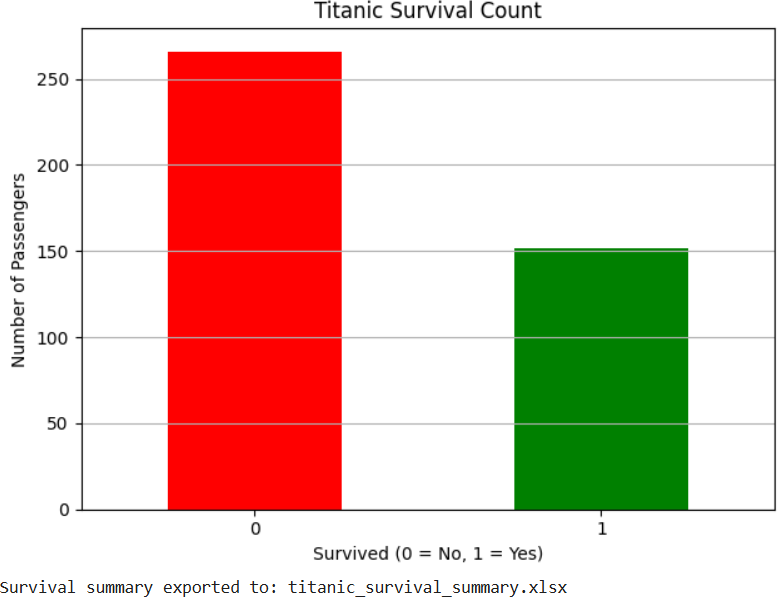
plt.tight\_layout() plt.show()

# Export SQL result to Excel df\_sql.to\_excel("titanic\_survival\_summary.xlsx", index=False)

print("Survival summary exported to: titanic\_survival\_summary.xlsx")

**Output:**



****

# EXPERIMENT - 3

**EDA-Data Cleaning:**

# CODE:

# 📌 Step 1: Import required libraries import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler, MinMaxScaler

# 📌 Step 2: Create a sample DataFrame data = {

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

'Age': [25, np.nan, 30, 22, 29, 25],

'Salary': [50000, 60000, 55000, 52000, np.nan, 50000],

'Department': ['HR', 'IT', 'IT', 'HR', 'Finance', 'HR']

}

df = pd.DataFrame(data)

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

# 📌 Step 3: Detect missing values

print("\n🔍 Missing values before cleaning:") print(df.isnull().sum())

# 📌 Step 4: Fill missing values using forward fill (new syntax) df.ffill(inplace=True)

# 📌 Step 5: Drop any remaining rows with missing values df.dropna(inplace=True)

# 📌 Step 6: Show missing values after cleaning

print("\n✅ Missing values after filling and dropping:") print(df.isnull().sum())

# 📌 Step 7: Remove duplicate rows

print("\n🔁 Duplicate rows before removal:", df.duplicated().sum())

df.drop\_duplicates(inplace=True)

print("✅ Duplicate rows after removal:", df.duplicated().sum())

# 📌 Step 8: Remove unnecessary columns df.drop(columns=['Name'], inplace=True)

# 📌 Step 9: Show data types before conversion print("\n📋 Data types before conversion:")

print(df.dtypes)

# 📌 Step 10: Convert data types for consistency df['Age'] = df['Age'].astype(int)

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

# 📌 Step 11: Show data types after conversion print("\n✅ Data types after conversion:")

print(df.dtypes)

# 📌 Step 12: Normalize numerical columns

numeric\_cols = df.select\_dtypes(include=[np.number]).columns.tolist() print("\n🔢 Numeric columns to normalize:", numeric\_cols)

# Standardization (Z-score) scaler\_std = StandardScaler() df\_standardized = df.copy() df\_standardized[numeric\_cols] =

scaler\_std.fit\_transform(df\_standardized[numeric\_cols])

# Min-Max Scaling (0–1 range) scaler\_mm = MinMaxScaler() df\_minmax = df.copy()

df\_minmax[numeric\_cols] = scaler\_mm.fit\_transform(df\_minmax[numeric\_cols])

# 📌 Step 13: Save cleaned data to CSV df.to\_csv("cleaned\_data.csv", index=False) df\_standardized.to\_csv("standardized\_data.csv", index=False) df\_minmax.to\_csv("minmax\_scaled\_data.csv", index=False)

# 📌 Step 14: Final Outputs

print("\n✅ Final Cleaned DataFrame:\n")

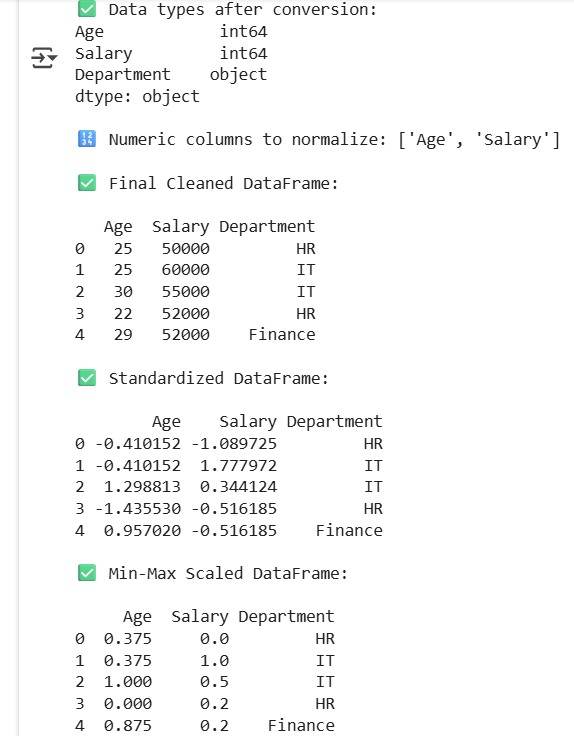
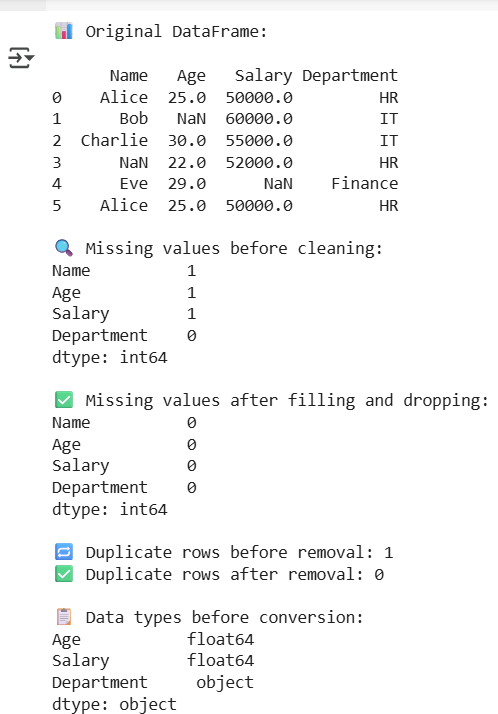
print(df)

print("\n✅ Standardized DataFrame:\n") print(df\_standardized)

print("\n✅ Min-Max Scaled DataFrame:\n") print(df\_minmax)

print("\n📁 Files saved: cleaned\_data.csv, standardized\_data.csv, minmax\_scaled\_data.csv")

# OUTPUT:

****

**EXPERIMENT - 4**

**EDA-Data Inspection and Analysis:**

# CODE:

# 📌 Step 1: Import necessary libraries import pandas as pd

import numpy as np

# 📌 Step 2: Create a sample DataFrame data = {

'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'], 'Age': [25, 30, 22, 28, 35],

'Salary': [50000, 60000, 52000, 58000, 65000],

'Department': ['HR', 'IT', 'HR', 'Finance', 'IT']

}

df = pd.DataFrame(data)

# 📌 Step 3: View and inspect the DataFrame print("🔍 First 5 rows of the DataFrame:\n") print(df.head())

print("\n📋 Info about DataFrame:") print(df.info())

print("\n📏 Shape of DataFrame:", df.shape)

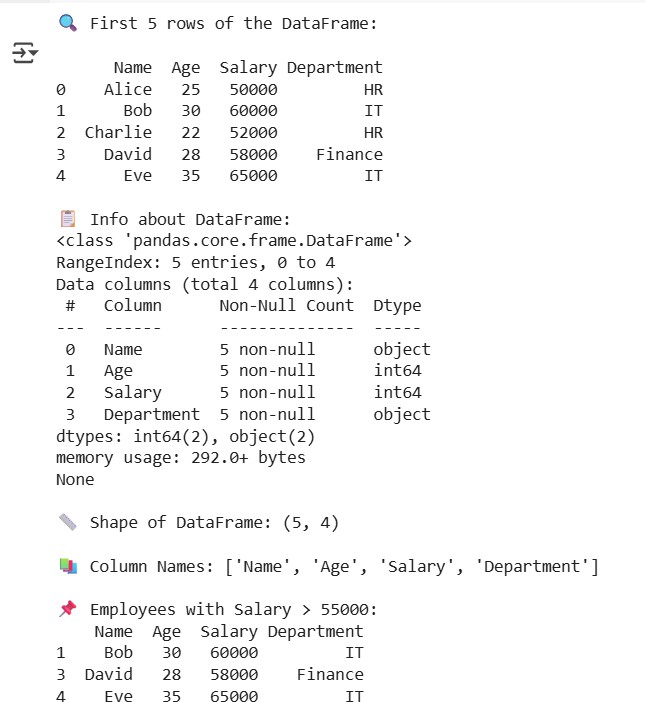
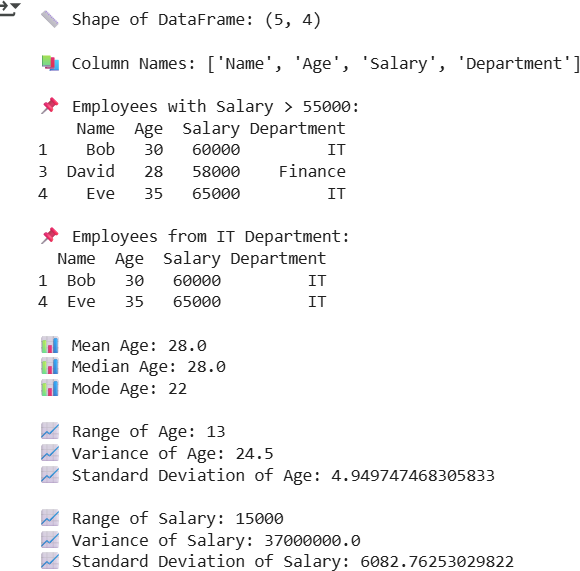
print("\n📚 Column Names:", df.columns.tolist())

# 📌 Step 4: Filter and subset data using conditions print("\n📌 Employees with Salary > 55000:") high\_salary = df[df['Salary'] > 55000]

print(high\_salary)

print("\n📌 Employees from IT Department:") it\_employees = df[df['Department'] == 'IT'] print(it\_employees)

|  |  |  |
| --- | --- | --- |
| # 📌 Step 5: Descriptive statistics - Central Tendency print("\n📊 Mean Age:", df['Age'].mean())  print("📊 Median Age:", df['Age'].median()) | |  |
| print("📊 Mode Age:", df['Age'].mode()[0]) |  |
| # 📌 Step 6: Descriptive statistics - Dispersion range\_age = df['Age'].max() - df['Age'].min()  print("\n📈 Range of Age:", range\_age)  print("📈 Variance of Age:", df['Age'].var())  print("📈 Standard Deviation of Age:", df['Age'].std()) | |
| range\_salary = df['Salary'].max() - df['Salary'].min() print("\n📈 Range of Salary:", range\_salary)  print("📈 Variance of Salary:", df['Salary'].var())  print("📈 Standard Deviation of Salary:", df['Salary'].std()) | | |

**OUTPUT:**

**Experiment 5**

**Aim:** EDA-Data Visualization with Matplotlib

Basic plotting: line charts, bar charts, histograms

**Code:**

# Importing libraries

import pandas as pd

import matplotlib.pyplot as plt

# --- Step 1: Load dataset ---

df = pd.read\_csv("Sample - Superstore.csv", encoding='latin1')

print("Dataset loaded successfully!")

print(df.head())

# --- Step 2: Line Chart (Sales Trend over Time) ---

# Convert 'Order Date' to datetime

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

# Group by Order Date and sum sales

sales\_trend = df.groupby('Order Date')['Sales'].sum()

plt.figure(figsize=(10, 5))

plt.plot(sales\_trend.index, sales\_trend.values, color='b', linewidth=2)

plt.title("Daily Sales Trend")

plt.xlabel("Date")

plt.ylabel("Total Sales ($)")

plt.grid(True, linestyle='--', alpha=0.6)

plt.show()

# --- Step 3: Bar Chart (Category-wise Average Profit) ---

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

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

plt.bar(category\_profit.index, category\_profit.values, color=['skyblue', 'orange', 'lightgreen'])

plt.title("Average Profit by Category")

plt.xlabel("Product Category")

plt.ylabel("Average Profit ($)")

plt.grid(axis='y', linestyle='--', alpha=0.6)

plt.show()

# --- Step 4: Histogram (Sales Distribution) ---

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

plt.hist(df['Sales'], bins=30, color='purple', edgecolor='black', alpha=0.7)

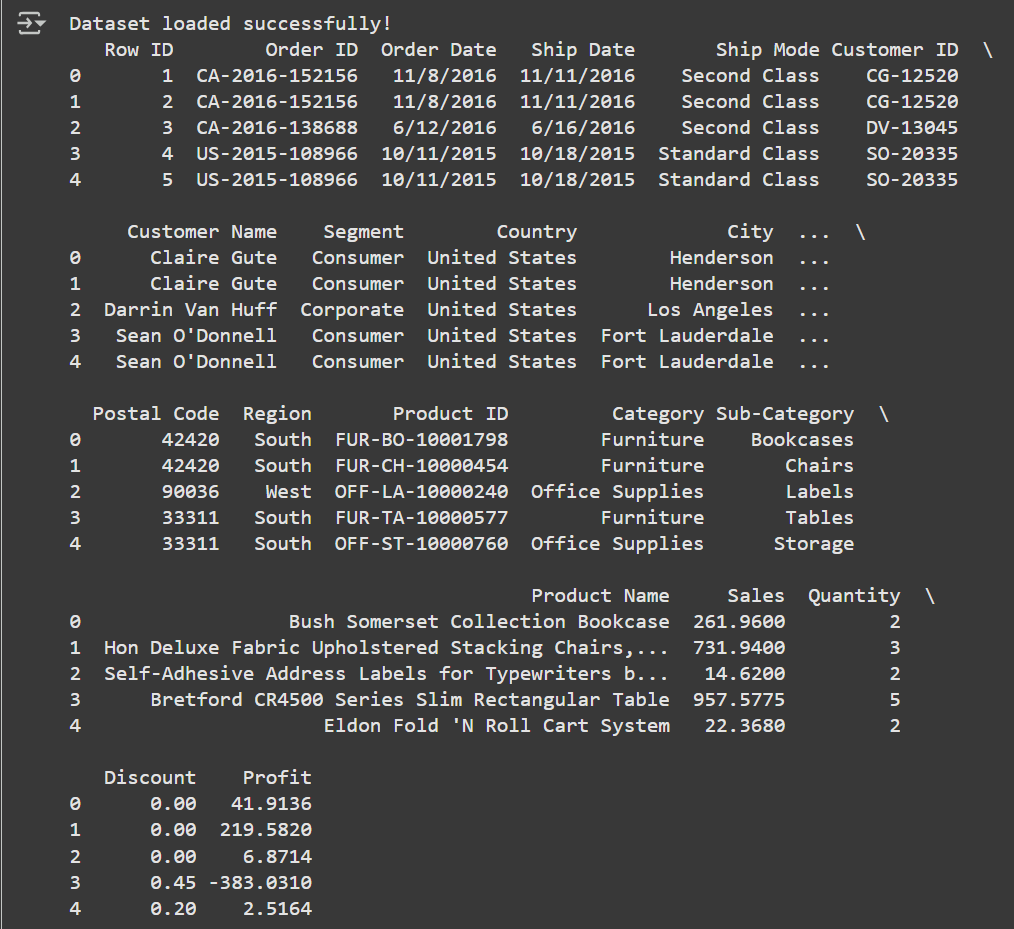
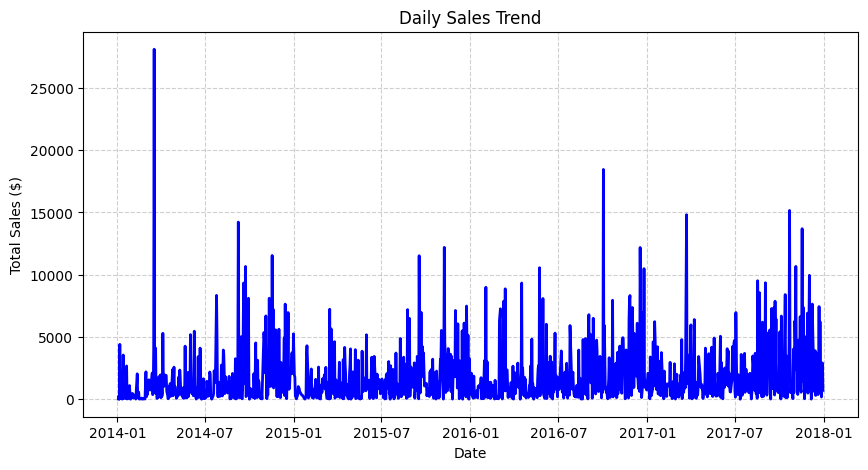
plt.title("Distribution of Sales")

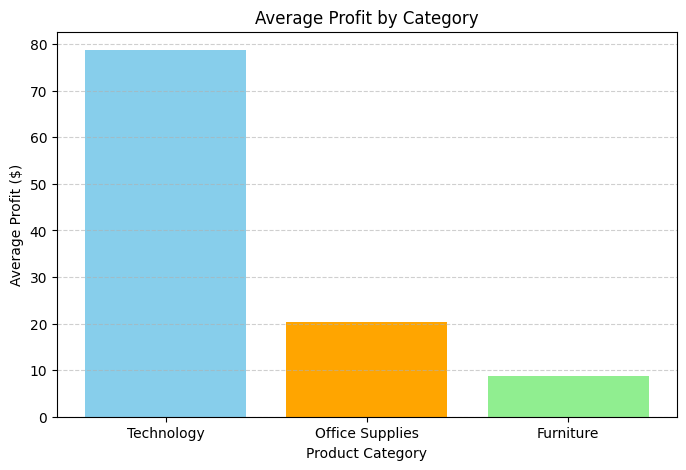
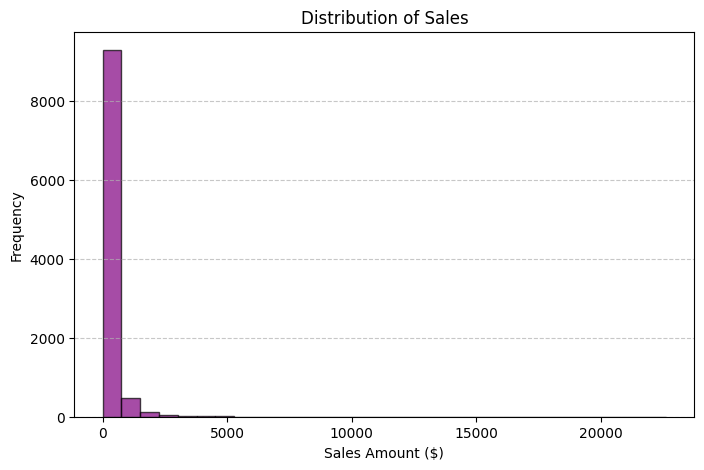
plt.xlabel("Sales Amount ($)")

plt.ylabel("Frequency")

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

Result:  
  
 

**Experiment 6**

**Data Visualization Using Power BI**

**Objective:** To visualize and analyze employee data using Power BI by connecting data sources, transforming data, and creating an interactive dashboard with key insights.

**Dataset Description:**

The dataset contains records of employees, covering demographic, job, and performance details.  
Key columns include:

* Employee\_ID: Unique identifier
* Department, Job\_Title: Work and role details
* Gender, Age, Education\_Level: Demographic info
* Monthly\_Salary, Years\_At\_Company: Employment details
* Performance\_Score, Employee\_Satisfaction\_Score: Evaluation metrics
* Remote\_Work\_Frequency, Promotions, Resigned: Work behavior indicators

Total attributes: 19 columns, each providing insights into workforce trends, performance, and satisfaction.

**Process:**

1. Data Preparation

* Imported the CSV dataset into Power BI Desktop.
* Used Power Query Editor to:
  + Correct data types (e.g., dates, numeric fields).
  + Remove unnecessary columns.
  + Promote headers and ensure data consistency.

2. Measures and Calculations

Created calculated metrics:

* Average Salary
* Average Satisfaction Score
* Resignation Rate

Used these as KPIs in the dashboard.

3. Visualizations Created

Six visuals were built to display employee insights:

1. Bar Chart – Employees by Monthly Salary
2. Line Chart – Employee Count by Year
3. Scatter Plot – Satisfaction vs. Performance
4. Column Chart – Average Salary by Job Title
5. Donut Chart – Remote Work Frequency
6. Column Chart – Department by Gender

4. Dashboard Design

* Professional dark background with gradient style.
* Consistent color palette for clarity.
* Added slicers and filters for interactivity.
* Combined visuals to highlight workforce diversity, job trends, and performance patterns.

**Output:**

****

**Observation:**

The Power BI dashboard effectively visualizes employee data, revealing trends in salary, job roles, satisfaction, and gender distribution. It demonstrates how visual analytics enhances HR insights and decision-making.

**Result:**

A complete Power BI dashboard was created using an employee dataset, including six visuals and key KPIs, providing meaningful insights into workforce structure and performance.

**Experiment 7**

**Data Visualization Using Tableau**

**Aim:**

To create interactive dashboards and visualizations using Tableau for better data understanding and insights.

**Objective:**

To explore data visualization techniques in Tableau by connecting to different data sources, applying calculated fields, and developing insightful dashboards.

**Tools Used:**

Tableau Desktop

**Dataset Description:**

A small HR dataset containing details such as job titles, departments, gender, education level, work hours, projects handled, and performance scores.  
The dataset is used to analyze workforce distribution, performance trends, and overtime patterns.

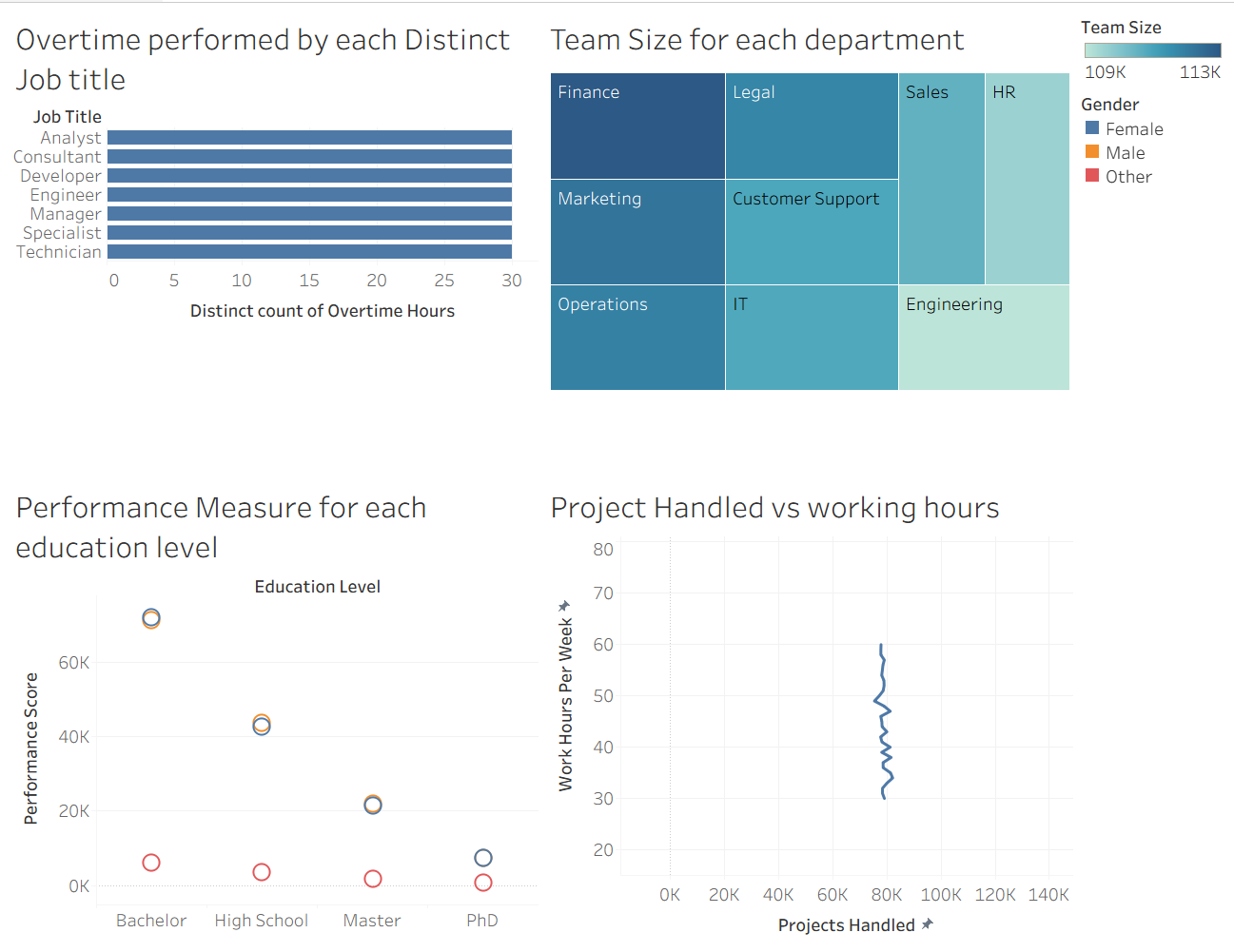
**Procedure:**

1. Open Tableau and connect to the dataset (Excel/CSV format).
2. Load the data and explore available fields.
3. Create visualizations including bar charts, scatter plots, and tree maps.
4. Use calculated fields for custom metrics (e.g., total overtime or average performance).
5. Combine all visuals into a single dashboard with filters and legends for interactivity.
6. Format the dashboard for clarity and visual appeal.

**Dashboard Design:**

The Tableau dashboard presents HR insights with clear and interactive visuals.  
It highlights overtime trends by job title, department team size, performance by education level, and the relationship between projects handled and working hours.  
The clean design, color consistency, and layout ensure easy interpretation of data.

**Output:**

****

**Result:**

An interactive Tableau dashboard was successfully created to visualize employee-related metrics, making it easier to analyze workforce trends and performance.

Conclusion:

This experiment demonstrates how Tableau simplifies complex data analysis through dynamic visualizations and dashboards, aiding data-driven decision-making**.**