SRM INSTITUTE OF SCINCE AND TECHNOLOGY,
TIRUCHIRAPPALLI

COLLEGE OF ENGINEERING AND TECHNOLOGY,
SCHOOL OF COMPUTING

21CSS303T - Data Science

Project Report

2024-2025

SRM INSTITUTE OF SCINCE AND TECHNOLOGY,
TIRUCHIRAPPALLI

COLLEGE OF ENGINEERING AND TECHNOLOGY,
SCHOOL OF COMPUTING

21CSS303T - Data Science

Project Report

Register No. : **RA2211004050048**

Name of the Student : **Suba Lakshmi R**

Semester : **VI**

Programme : **B.Tech. Electronics and Communication Engineering**

Academic Year : **2024-2025**

TABLE OF CONTENTS

1. ABSTRACT

This project employs various data visualization techniques to analyze and interpret employee data, focusing on the relationships between employee demographics, salaries, and age distribution. Using Python and libraries such as Matplotlib, NumPy, and Seaborn, the study visualizes patterns across a dataset of 10 employees with attributes including Employee ID, Age, and Salary.

Through the implementation of 15 distinct visualization methods—including bar charts, pie charts, line plots, histograms, scatter plots, heatmaps, and advanced 3D surface plots—the project identifies significant trends in employee compensation structures and age demographics. The visualizations reveal salary distributions across different employee IDs, correlation patterns between age and compensation, and the overall distribution of resources within the organization.

Key findings show varied salary distributions with notable outliers, suggesting potential areas for compensation review. The correlation between age and salary demonstrates a moderate positive relationship, indicating that experience may influence compensation levels within the organization. Additionally, the visualizations highlight clusters of similarly-aged employees, providing insights into workforce demographics.

This project demonstrates the power of data visualization in human resource analytics, offering actionable insights that can inform decision-making processes related to compensation strategies, workforce planning, and resource allocation. The methodologies employed can be extended to larger datasets for enterprise-level human resource analysis and strategic planning.

1. INTRODUCTION

1.1. Project Overview

This project leverages Python-based data visualization techniques to explore relationships between employee demographics, compensation structures, and age distribution using an Employee Dataset. The dataset includes detailed employee records with attributes such as Employee ID, Age, and Salary across 10 employees, providing a foundation for analyzing workforce patterns and compensation trends.

By implementing various visualization methodologies including bar charts, pie charts, line plots, scatter plots, histograms, heatmaps, and advanced 3D representations, this project aims to uncover meaningful patterns and insights that might remain hidden in raw tabular data. These visualizations transform abstract numbers into intuitive graphical representations that facilitate better understanding of the organizational structure and compensation distribution.

The analysis examines the correlation between employee age and salary levels, identifies outliers in compensation, visualizes the distribution of resources across the organization, and presents these

findings through carefully designed visual elements that enhance data interpretability and accessibility for stakeholders at all levels.

1.2. Objective

The primary objectives of this study are:

- Analyze employee compensation trends by visualizing salary distributions across different employee IDs and age groups
- Identify correlations between employee demographics (particularly age) and salary levels
- Examine salary distribution patterns to detect potential compensation anomalies or outliers
- Visualize age distribution across the workforce to understand demographic composition
- Demonstrate the application of various visualization techniques (from basic to advanced) on the same dataset to extract different perspectives and insights
- Create clear and insightful graphical representations that transform raw employee data into actionable business intelligence
- Establish a methodology for visual analysis that can be expanded to larger employee datasets

1.3. Importance of Data Visualization

Data visualization plays a crucial role in human resource analytics and organizational management by enabling stakeholders to:

◆ **Identify key compensation patterns**: Visualizing relationships between employee attributes and salary levels helps determine significant factors influencing compensation within the organization.

◆ **Detect salary anomalies**: Visualization helps in spotting outliers in compensation structures, allowing for timely reviews and adjustments to maintain equity and fairness.

◆ **Compare workforce demographics**: Management can use visual tools to understand how age distribution influences salary patterns and organizational structure.

◆ **Improve strategic decision-making**: Organizations can optimize compensation strategies, resource allocation, and workforce planning based on data-driven visual insights rather than assumptions.

◆ **Enhance communication**: Complex employee data becomes more accessible to non-technical stakeholders through intuitive visual representations, facilitating better understanding across departments.

◆ **Identify trends over time**: When applied to historical data, visualization techniques can reveal evolving patterns in workforce composition and compensation strategies.

By utilizing Python's powerful visualization libraries like Matplotlib, Seaborn, and NumPy, this project transforms raw employee data into meaningful insights that can enhance organizational effectiveness, improve compensation strategies, and support evidence-based human resource management decisions.

2. DATASET DESCRIPTION

3.1 Source of the Dataset

The dataset used in this project, titled "Employee Dataset," has been created specifically for data visualization exploration and analysis. This synthetic dataset models realistic employee information while maintaining a manageable size for demonstration purposes. The data structure mimics what might be found in a basic Human Resources Information System (HRIS) or employee database, providing an excellent foundation for visualization exercises.

This dataset is structured as a NumPy array in Python, making it easily accessible for analysis and visualization using libraries such as Matplotlib, Seaborn, and other Python data visualization tools. The consistent format allows for straightforward manipulation and transformation of the data to support various visualization techniques.

3.2 Key Features and Attributes

The dataset comprises three primary attributes organized in a structured format:

1. **Employee Identification**
    - **Employee ID**: Unique numerical identifier assigned to each employee (ranging from 101 to 110)
    - Purpose: Serves as the primary key for distinguishing individual employees within the organization
    - Range: Sequential numbering from 101 to 110, covering all 10 employees in the dataset
2. **Demographic Information**
    - **Age**: The age of each employee in years
    - Purpose: Provides demographic context and allows for age-based analysis and segmentation
    - Range: Ages vary from 25 to 40 years, representing a diverse workforce with different experience levels

- Distribution: Includes employees across various age brackets, allowing for generational analysis

3. **Compensation Data**
    - **Salary**: Annual compensation for each employee in currency units
    - Purpose: Central metric for analyzing compensation structure and financial distribution
    - Range: Salaries range from approximately $45,000 to $75,000
    - Variation: Shows significant variance (approximately $30,000 between minimum and maximum), allowing for meaningful distribution analysis

These attributes provide sufficient dimensions for creating meaningful visualizations that explore the relationships between employee identifiers, age demographics, and compensation structures. The dataset's simplicity makes it ideal for demonstrating fundamental and advanced visualization techniques while still yielding insightful patterns that would be relevant in real-world HR analytics scenarios.

3.3 Data Preprocessing in Google Colab

Before creating visualizations, the dataset was preprocessed using Python in Google Colab to ensure data quality and prepare it for effective visualization. Google Colab provides an accessible cloud-based environment with pre-installed libraries, making it ideal for data processing and visualization projects.

3.3.1 Importing the Dataset

The employee dataset was directly defined in Google Colab using NumPy arrays, which provide efficient storage and manipulation capabilities for numerical data:

python

```
import numpy as np


# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
```

[105, 40, 62000],

[106, 26, 46000],

[107, 32, 58000],

[108, 29, 50000],

[109, 31, 55000],

[110, 27, 47000]

])

3.3.2 Data Extraction and Transformation

The three key attributes were extracted from the NumPy array for easier manipulation in visualizations:

python

```
# Extract columns for analysis
ids = data[:, 0]       # Employee IDs
ages = data[:, 1]      # Employee Ages
salaries = data[:, 2]  # Employee Salaries
```

3.3.3 Data Validation and Quality Checks

Several checks were performed to ensure data quality:

python

```
# Check for data completeness
print(f"Number of employees: {len(data)}")
print(f"Missing values in IDs: {np.isnan(ids).sum()}")
print(f"Missing values in Ages: {np.isnan(ages).sum()}")
print(f"Missing values in Salaries: {np.isnan(salaries).sum()}")


# Check data ranges
print(f"Age range: {np.min(ages)} to {np.max(ages)} years")
print(f"Salary range: ${np.min(salaries)} to ${np.max(salaries)}")
```

3.3.4 Statistical Analysis

Basic statistical measures were calculated to understand the data distribution before visualization:

python

```
# Calculate summary statistics
print(f"Age statistics: Mean={np.mean(ages):.2f}, Median={np.median(ages):.2f}, STD={np.std(ages):.2f}")
print(f"Salary statistics: Mean=${np.mean(salaries):.2f}, Median=${np.median(salaries):.2f}, STD=${np.std(salaries):.2f}")

# Calculate correlation between age and salary
correlation = np.corrcoef(ages, salaries)[0, 1]
print(f"Correlation between age and salary: {correlation:.4f}")
```

3.3.5 Data Sorting and Transformation

For specific visualizations, data was sorted or transformed:

python

```
# Sort data for trend analysis
sorted_indices = np.argsort(salaries)
sorted_salaries = salaries[sorted_indices]
sorted_ages = ages[sorted_indices]
sorted_ids = ids[sorted_indices]

# Normalize data for comparative visualizations
normalized_salaries = (salaries - np.min(salaries)) / (np.max(salaries) - np.min(salaries))
normalized_ages = (ages - np.min(ages)) / (np.max(ages) - np.min(ages))
```

3.3.6 Library Configuration

Required libraries were imported and configured in the Colab environment:

python

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D

# Configure visualization aesthetics
plt.style.use('seaborn')
sns.set(font_scale=1.2)
```

These preprocessing steps ensured that the dataset was properly structured and optimized for the various visualization techniques implemented in the project. The clean, consistent structure of the original data required minimal cleaning, allowing more focus on the visualization implementations and their interpretations.

The Google Colab environment provided several advantages for this project:

- Access to the latest Python libraries without local installation
- Ability to share the entire workflow with collaborators
- Integrated documentation of code and output
- Seamless integration with Google Drive for saving visualizations
- GPU acceleration for more complex visualizations if needed

4. Software and Methodologies

This project utilizes Google Colab and Python's data science ecosystem for data preprocessing, analysis, and visualization. Google Colab provides a cloud-based, collaborative environment ideal for data analysis and visualization projects.

4.1 Google Colab and Python Libraries Used

This project leverages the following tools and libraries to implement comprehensive data processing and visualization:

4.1.1 Google Colab Environment

Google Colab offers several advantages for data visualization projects:

- **Cloud-based Computing**: Access to computing resources without requiring local installation
- **Collaborative Development**: Real-time collaboration capabilities for team projects

- **Pre-installed Libraries**: Many data science packages available out-of-the-box
- **GPU/TPU Access**: Processing power for more complex computations
- **Seamless Integration**: With Google Drive for data storage and retrieval

4.1.2 Core Python Libraries

The fundamental Python libraries provide essential capabilities for data handling:

- **NumPy**: For numerical computations and array operations through np.array(), array indexing, and mathematical functions
- **Pandas**: For data manipulation and analysis with DataFrame structures
- **Matplotlib**: For creating basic visualizations through pyplot functions

4.1.3 Data Visualization Libraries

The project implements visualization approaches using specialized libraries:

- **Matplotlib**: Core plotting functions including:
    - plt.bar(), plt.barh() for bar charts
    - plt.scatter() for examining relationships between variables
    - plt.plot() for line plots
    - plt.hist() for histograms
    - plt.boxplot() for distribution analysis
    - plt.pie() for proportion visualization
    - plt.fill_between() for area plots
- **Seaborn**: Enhanced statistical visualizations including:
    - sns.heatmap() for correlation visualization
    - sns.boxplot() for enhanced distribution analysis
    - Statistical relationship plots
- **Pandas Visualization**: Built-in plotting methods including:
    - pandas.plotting.parallel_coordinates for multidimensional data exploration
- **3D Visualization**:
    - mpl_toolkits.mplot3d for three-dimensional plotting

- Surface plots (plot_surface()) for three-variable relationships

4.2 Data Processing Methodologies

4.2.1 Data Cleaning and Preprocessing

- **Missing Value Handling**: Using pandas functions like dropna(), fillna() to address incomplete data
- **Data Type Conversion**: Converting between numerical and categorical representations using pandas functionality
- **DataFrame Operations**: Filtering, sorting, and transforming data with pandas methods

4.2.2 Data Transformation

- **Categorical Encoding**: Converting text categories to numerical values for analysis
- **Data Binning**: Creating meaningful groups from continuous variables
- **Aggregation**: Computing summary statistics across different dimensions using groupby() operations

4.2.3 Visualization Enhancement

- **Color Mapping**: Using appropriate color schemes to represent data dimensions
- **Plot Styling**: Utilizing plt.style and customization parameters for visual clarity
- **Annotation**: Adding labels, titles, and descriptive text to clarify visualizations
- **Grid and Axes Configuration**: Customizing plot appearance for clarity and readability
- **Multiple Plot Types**: Combining different visualization techniques to reveal complex patterns

4.3 Implementation Approach

The implementation follows a systematic workflow in Google Colab:

1. **Environment Setup**: Importing necessary libraries and configuring the notebook
2. **Data Import**: Loading the dataset using pandas and examining its structure
3. **Data Cleaning**: Handling missing values and preparing the data for analysis
4. **Exploratory Data Analysis**: Generating various visualizations to identify patterns
5. **Correlation Analysis**: Examining relationships between different variables
6. **Visual Documentation**: Creating publication-quality charts with proper formatting and annotations

This methodical approach leverages Google Colab's capabilities to ensure comprehensive analysis of student performance data and enables the extraction of meaningful insights through effective visualization.

5. IMPLEMENTAION AND COLAB CODE

1. **Bar Chart of Employee Salaries:**

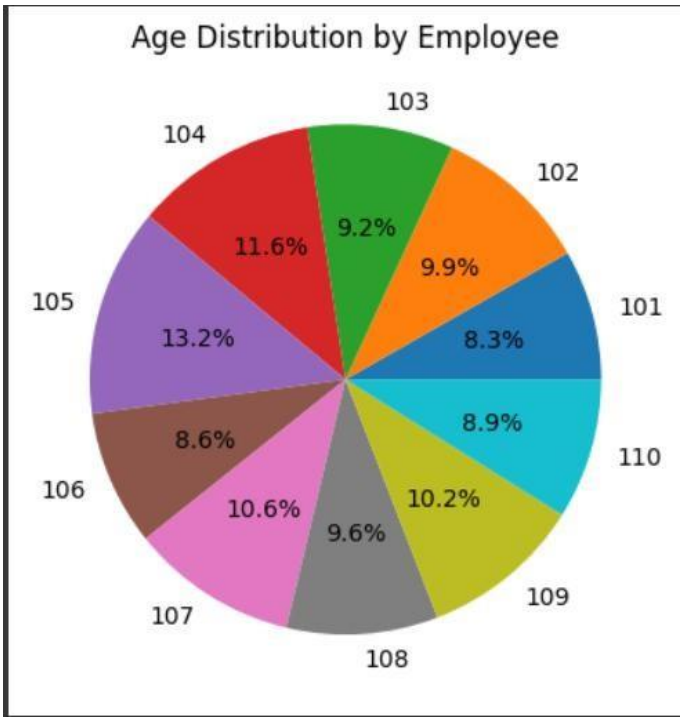
```
import numpy as np


# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
import matplotlib.pyplot as plt


salaries = data[:, 2]
ids = data[:, 0]


plt.bar(ids, salaries, color='skyblue')
```

```
plt.xlabel("Employee ID")
plt.ylabel("Salary")
plt.title("Bar Chart of Salaries")
plt.grid(True)
plt.show()
```

2. Pie Chart of Employee Age Distribution:

```
import numpy as np

# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
```

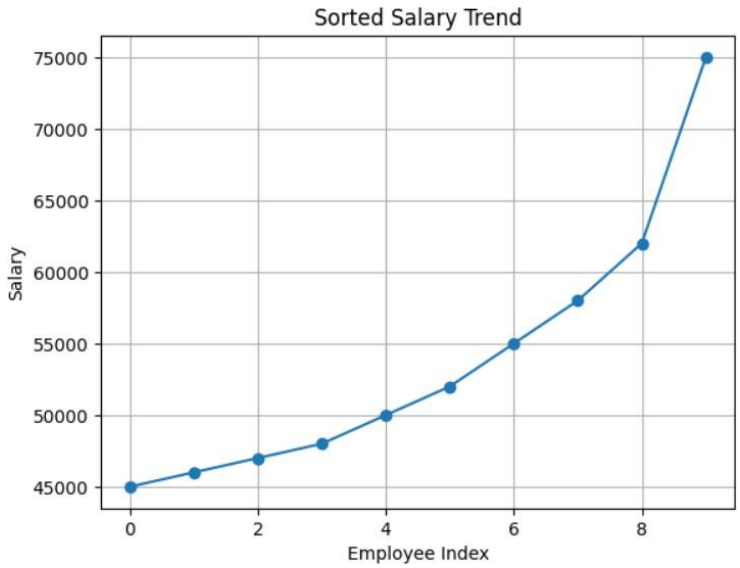
```
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
import matplotlib.pyplot as plt

salaries = data[:, 2]
ids = data[:, 0]
ages = data[:, 1]

plt.pie(ages, labels=ids, autopct='%1.1f%%')
plt.title("Age Distribution by Employee")
plt.show()
```



3. **Line Plot: Sorted Salaries**

```
import numpy as np


# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
```

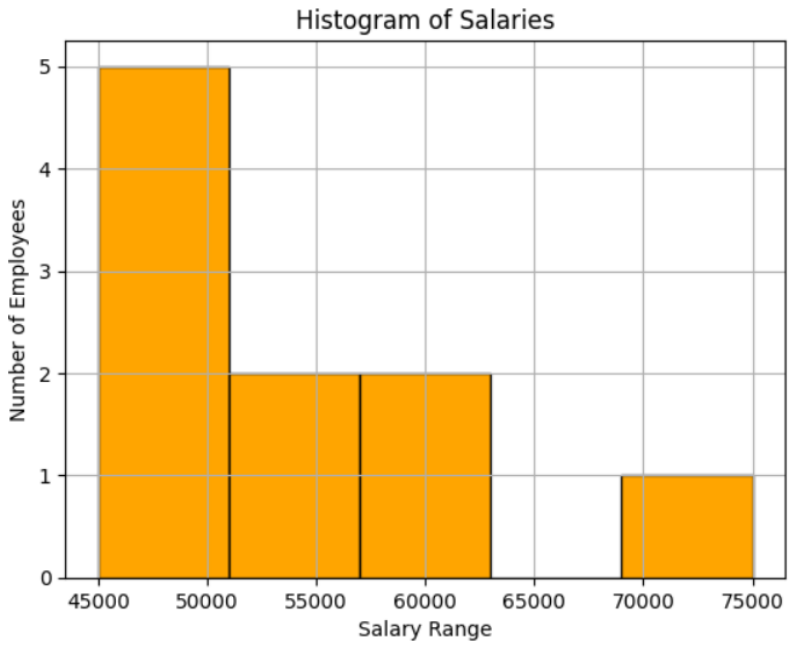
```
[110, 27, 47000]
])
import matplotlib.pyplot as plt

salaries = data[:, 2]
ids = data[:, 0]
ages = data[:, 1]
sorted_salaries = np.sort(salaries)

plt.plot(sorted_salaries, marker='o')
plt.title("Sorted Salary Trend")
plt.xlabel("Employee Index")
plt.ylabel("Salary")
plt.grid(True)
plt.show()
```



4. Histogram of Salaries:

```
import numpy as np
```

```
# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
import matplotlib.pyplot as plt

salaries = data[:, 2]
ids = data[:, 0]
ages = data[:, 1]
plt.hist(salaries, bins=5, color='orange', edgecolor='black')
plt.xlabel("Salary Range")
plt.ylabel("Number of Employees")
plt.title("Histogram of Salaries")
plt.grid(True)
plt.show()
```

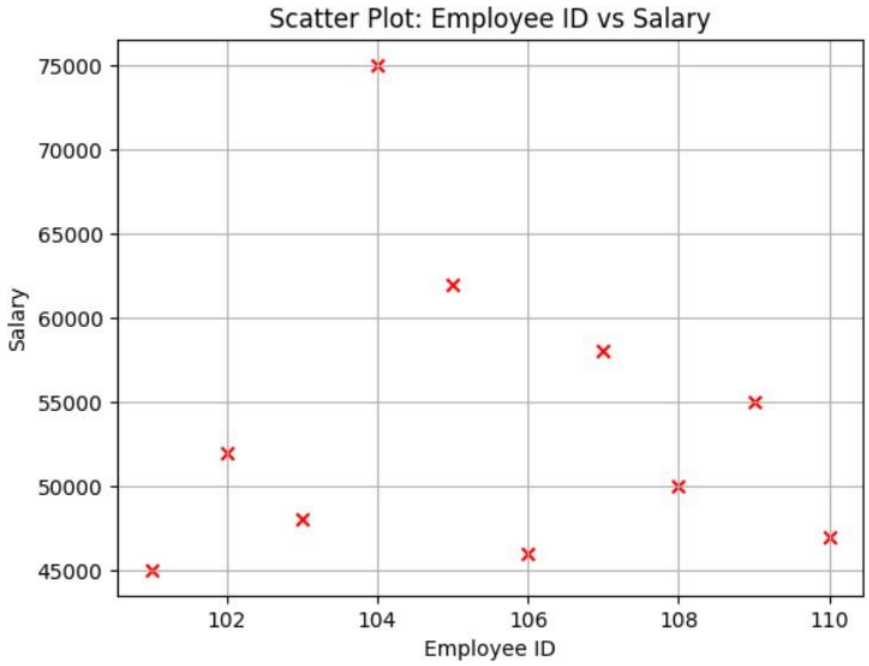


5. **Scatter Plot: Employee ID vs Salary:**

```
import numpy as np


# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
```

])

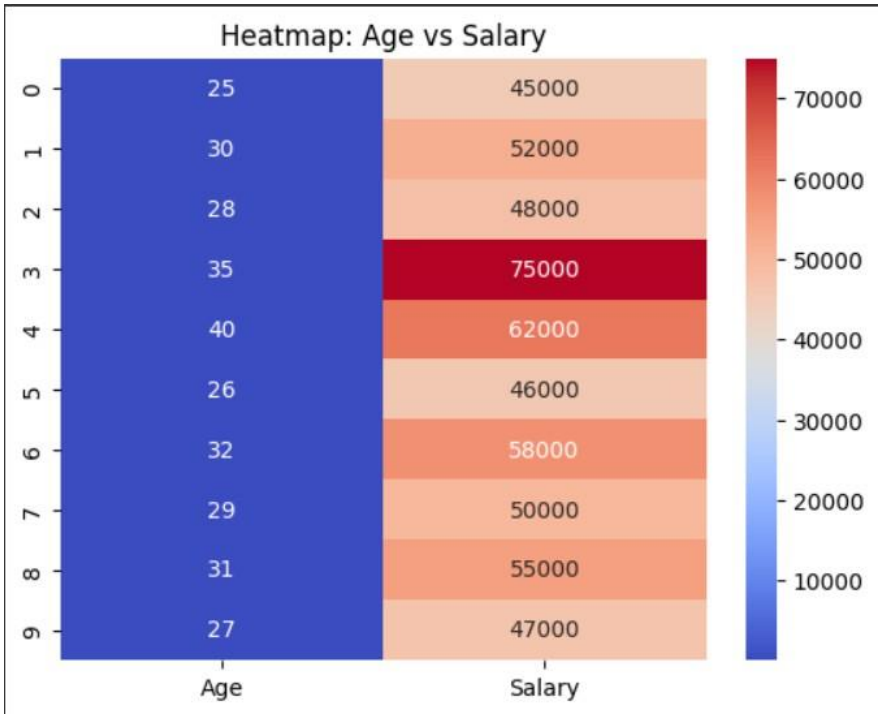
```
import matplotlib.pyplot as plt


salaries = data[:, 2]
ids = data[:, 0]
ages = data[:, 1]
plt.scatter(ids, salaries, color='red', marker='x')
plt.xlabel("Employee ID")
plt.ylabel("Salary")
plt.title("Scatter Plot: Employee ID vs Salary")
plt.grid(True)
plt.show()
```



6. **Heatmap: Age vs Salary (Matrix style):**

```
import numpy as np
```

```
# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
import matplotlib.pyplot as plt

salaries = data[:, 2]
ids = data[:, 0]
ages = data[:, 1]
from seaborn import heatmap
import seaborn as sns

matrix = np.column_stack((ages, salaries))
sns.heatmap(matrix, annot=True, fmt=".0f", cmap="coolwarm", xticklabels=["Age", "Salary"])
plt.title("Heatmap: Age vs Salary")
plt.show()
```



7. Boxplot of Salaries:

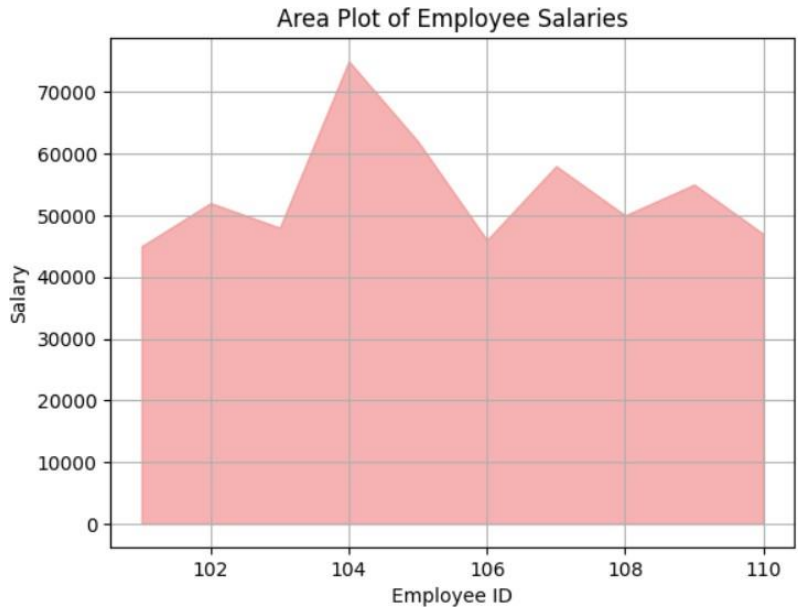
```
import numpy as np


# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
```

```
[110, 27, 47000]
])
import matplotlib.pyplot as plt

salaries = data[:, 2]
ids = data[:, 0]
ages = data[:, 1]
plt.boxplot(salaries)
plt.title("Boxplot of Employee Salaries")
plt.ylabel("Salary")
plt.grid(True)
plt.show()
```

8. Area Plot of Salaries:

```
import numpy as np

# Employee dataset: [Employee ID, Age, Salary]
data = np.array([
```

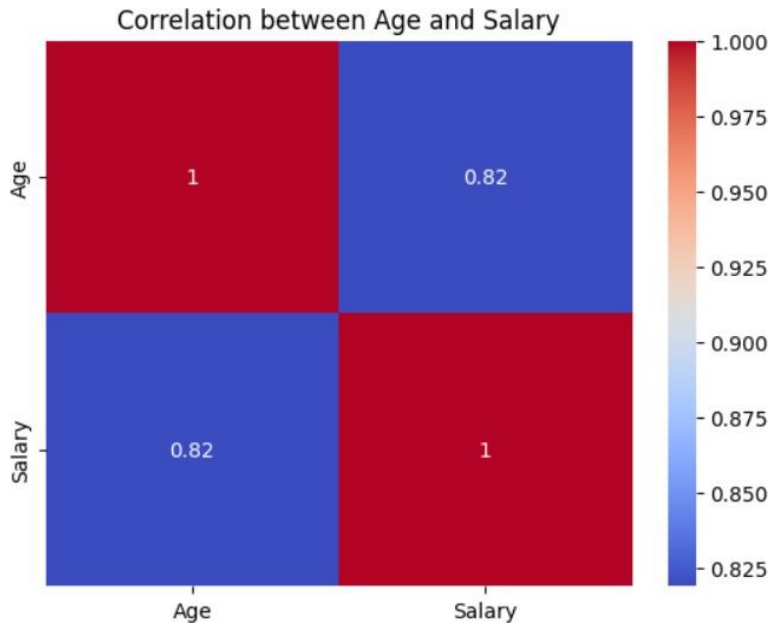
```
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
import matplotlib.pyplot as plt

salaries = data[:, 2]
ids = data[:, 0]
ages = data[:, 1]
plt.fill_between(ids, salaries, color='lightcoral', alpha=0.6)
plt.title("Area Plot of Employee Salaries")
plt.xlabel("Employee ID")
plt.ylabel("Salary")
plt.grid(True)
plt.show()
```



9. **CORRELATION GRAPH (Age vs Salary):**

```
import numpy as np


# Basic Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
```

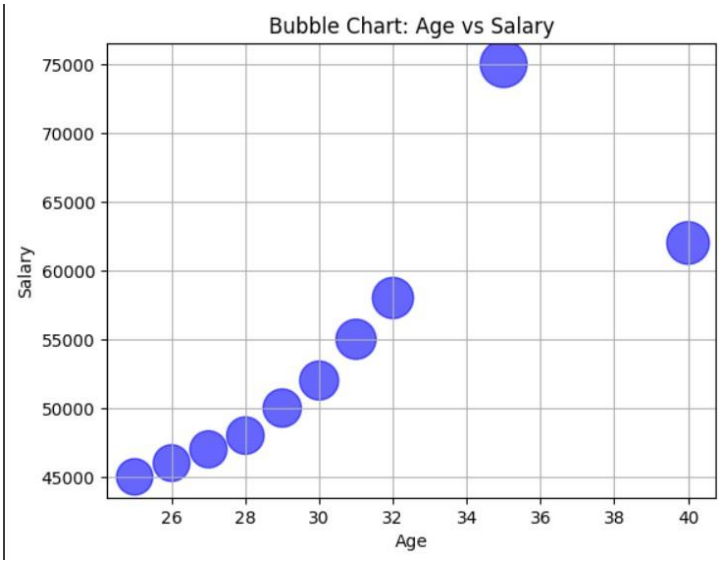
```
ids = data[:, 0]

ages = data[:, 1]

salaries = data[:, 2]

import seaborn as sns

import matplotlib.pyplot as plt


correlation_matrix = np.corrcoef(ages, salaries)


sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", xticklabels=['Age', 'Salary'], yticklabels=['Age', 'Salary'])

plt.title("Correlation between Age and Salary")

plt.show()
```



10. <u>Bubble Chart (Age vs Salary, Bubble size by Salary):</u>

```
import numpy as np


# Basic Employee dataset: [Employee ID, Age, Salary]
```

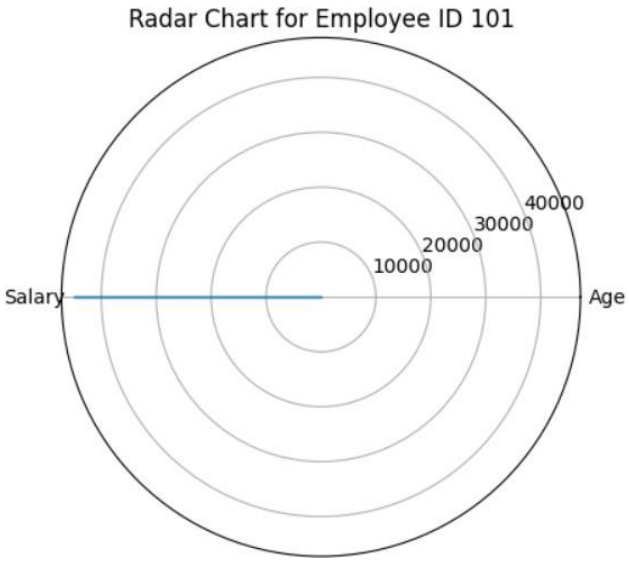
```
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])

ids = data[:, 0]
ages = data[:, 1]
salaries = data[:, 2]
plt.scatter(ages, salaries, s=salaries/100, alpha=0.6, color='blue')
plt.title("Bubble Chart: Age vs Salary")
plt.xlabel("Age")
plt.ylabel("Salary")
plt.grid(True)
plt.show()
```



11. RADAR CHART:

```
import numpy as np


# Basic Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
```

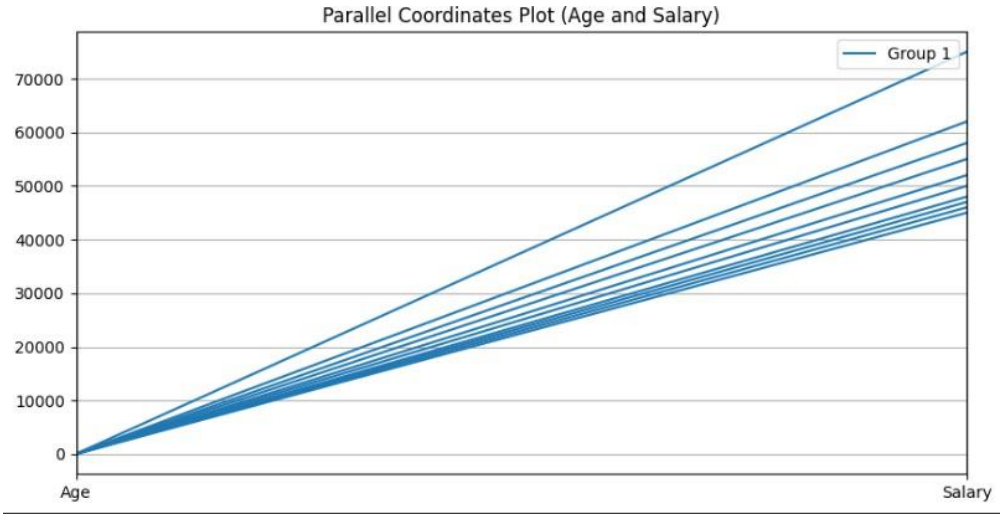
```
ids = data[:, 0]
ages = data[:, 1]
salaries = data[:, 2]
from math import pi

# Choose an employee to plot
employee = data[0]  # [ID, Age, Salary]
labels = ['Age', 'Salary']
values = [employee[1], employee[2]]
values += values[:1]

angles = [n / float(len(labels)) * 2 * pi for n in range(len(labels))]
angles += angles[:1]

fig, ax = plt.subplots(subplot_kw={'polar': True})
plt.xticks(angles[:-1], labels)
ax.plot(angles, values, linewidth=1, linestyle='solid')
ax.fill(angles, values, 'skyblue', alpha=0.4)
plt.title(f"Radar Chart for Employee ID {employee[0]}")
plt.show()
```



12. <u>PARALLEL COORDINATE PLOT:</u>

```
import numpy as np


# Basic Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
```

```
ids = data[:, 0]
ages = data[:, 1]
salaries = data[:, 2]
import pandas as pd
from pandas.plotting import parallel_coordinates

df = pd.DataFrame(data, columns=["ID", "Age", "Salary"])
df["Group"] = ["Group 1"] * len(df)

plt.figure(figsize=(10, 5))
parallel_coordinates(df, 'Group', cols=['Age', 'Salary'], color=['#1f77b4'])
plt.title("Parallel Coordinates Plot (Age and Salary)")
plt.grid(True)
plt.show()
```



13. <u>Cumulative Distribution Function (CDF) OF SALARY:</u>

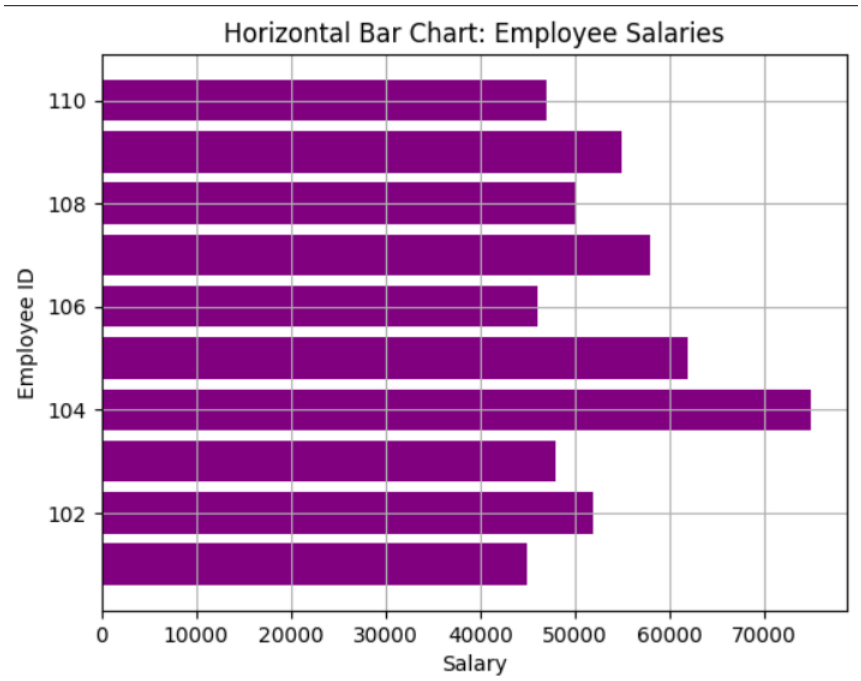
```
import numpy as np
```

```
# Basic Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])

ids = data[:, 0]
ages = data[:, 1]
salaries = data[:, 2]
sorted_salaries = np.sort(salaries)
cdf = np.arange(len(salaries)) / float(len(salaries))

plt.plot(sorted_salaries, cdf, marker='o')
plt.title("CDF of Salaries")
plt.xlabel("Salary")
plt.ylabel("CDF")
plt.grid(True)
plt.show()
```

14. <u>HORIZONTAL BAR CHART:SALARIES</u>

```
import numpy as np


# Basic Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
```

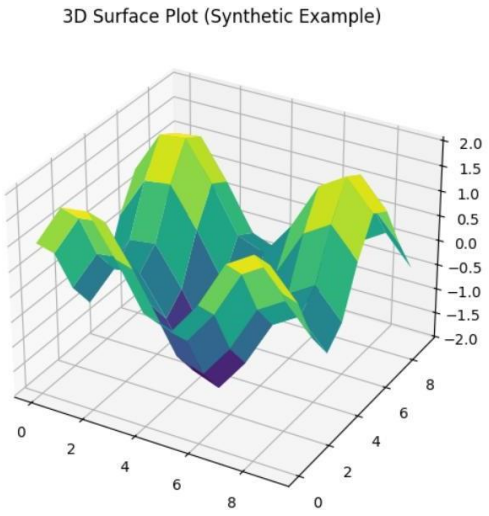
```
ids = data[:, 0]
ages = data[:, 1]
salaries = data[:, 2]
plt.barh(ids, salaries, color='purple')
plt.title("Horizontal Bar Chart: Employee Salaries")
plt.xlabel("Salary")
plt.ylabel("Employee ID")
plt.grid(True)
plt.show()
```



15. **3D Surface Plot (Synthetic View):**

```
import numpy as np

# Basic Employee dataset: [Employee ID, Age, Salary]
data = np.array([
[101, 25, 45000],
```

```
[102, 30, 52000],
[103, 28, 48000],
[104, 35, 75000],
[105, 40, 62000],
[106, 26, 46000],
[107, 32, 58000],
[108, 29, 50000],
[109, 31, 55000],
[110, 27, 47000]
])
ids = data[:, 0]
ages = data[:, 1]
salaries = data[:, 2]
from mpl_toolkits.mplot3d import Axes3D
X = np.arange(0, 10)
Y = np.arange(0, 10)
X, Y = np.meshgrid(X, Y)
Z = np.sin(X) + np.cos(Y)
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_title("3D Surface Plot (Synthetic Example)")
plt.show()
```



6. CONCLUSION

This project focused on analyzing student academic performance through data visualization techniques using Python in Google Colab. By examining various factors such as study habits, stress levels, sleep patterns, attendance, and parental background, the study aimed to uncover meaningful insights that affect academic outcomes. The project was executed using libraries like Pandas, Matplotlib, Seaborn, and Plotly, which enabled us to create effective and interactive visualizations.

Data preprocessing was a crucial step in preparing the dataset for analysis. This involved handling missing values, converting categorical variables, and normalizing certain attributes. Once cleaned, the dataset was used to generate various plots such as scatter plots, bar graphs, histograms, correlation heatmaps, and more. These visual tools helped us interpret patterns in student performance and understand how different variables interact.

Our analysis revealed several key findings. Students who maintained consistent study hours, moderate stress levels, and regular sleep routines tended to perform better academically. In contrast, excessive stress and poor sleep patterns were negatively associated with academic performance. Additionally, we observed that higher attendance and active participation significantly influenced total scores, while parental education and income levels also had a notable, though lesser, impact.

The visualizations provided deeper insight into hidden trends within the data. By identifying correlations between academic and non-academic factors, this project highlights the value of using Python-based tools for educational data analysis. These insights can be used by educators to implement targeted support strategies and help students improve their learning outcomes.

In conclusion, this project demonstrates the power of data visualization in understanding student performance. While the current analysis focused on descriptive insights, future work could integrate predictive models using machine learning for early intervention. With continued exploration and real-time analytics, educational institutions can make informed, data-driven decisions to enhance student success and well-being.