# Staff Appraisal System

**MCA(Master in Computer Application)**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA, PUNJAB**


**Submitted By :**

**Sahil Rana – 12216166**
**Roshan Nayak – 12216172**
**Abhishek Rana – 12216226**

**Submitted To :**

**Dr. Tarandeep Singh Walia (25153)**

## Table of Contents

# Introduction

## Project Overview :

A Staff Appraisal System is a crucial component of effective human resource management within an organization. It serves as a structured and systematic process for evaluating and assessing the performance and development of employees. The primary goal of a staff appraisal system is to ensure that employees are contributing positively to the organization's objectives, and it provides a mechanism for feedback, recognition, and improvement.

## Scope :

The scope of a staff appraisal system encompasses a wide range of activities and considerations aimed at evaluating and managing employee performance and development within an organization. The scope can vary depending on the organization's size, industry, and specific objectives. Here are the key aspects that fall within the scope of a staff appraisal system:

1. Performance Evaluation
2. Goal Setting and Expectation Management
3. Data Collection and Analysis

## Project Description :

In the description, we will explore the key elements and benefits of a staff appraisal system, highlighting its significance in modern workplace management.

1. **Purpose and Objectives**:
    - The staff appraisal system is designed to achieve several key objectives, including:
        - Evaluating individual employee performance.
        - Identifying strengths and weaknesses.
        - Setting performance goals and expectations.
        - Facilitating career development and growth.
        - Recognizing and rewarding outstanding contributions.
        - Enhancing communication between employees and managers.
        - Ensuring alignment with organizational goals and values.
2. **Benefits**:
    - Implementing an effective staff appraisal system offers numerous advantages to both employees and the organization:
        - Improved employee performance and productivity.
        - Increased job satisfaction and motivation.
        - Enhanced communication and transparency.
        - Better alignment of individual and organizational goals.
        - Identification of training and development needs.
        - Fair and consistent recognition and reward systems.
        - Data for talent management and succession planning.
3. **Key Components**:
    - A typical staff appraisal system consists of the following essential components:
        - Performance Standards and Criteria: Clear and measurable expectations for employee performance.
        - Performance Appraisal Forms: Documents or software used to record evaluations and feedback.
        - Self-Assessment: Employees' self-evaluation of their performance.
        - Managerial Assessment: Evaluations provided by supervisors or managers.
        - Goal Setting: Setting performance goals and development plans.
        - Feedback and Communication: Constructive feedback and discussion between employees and managers.
        - Performance Ratings: Assigning scores or ratings to various aspects of performance.
        - Development Plans: Identifying areas for improvement and plans for growth.

**Project Modules**

**1. Add Employees**
**2. Conduct Appraisal**
**3. Generate Performance report**
**4. Plot Performance Graph**
**5. Import Data from CSV**
**6. Export Data to CSV**
**7. Delete Employee**
**8. Save Data to CSV**
**9. View CSV Data**
**10. Plot Pie Chart**

**1. Add Employees – This Module help to add employee in dataset. We have to add his name , id and have to fill other attributes.**

**2. Conduct Appraisal – This Module is used to conduct appraisal on any dataset.**

**3. Generate Performance Report - This Module is used to Generate Performance report in the PDF format.**

**4. Plot Performance Graph - This Module is used to Generate the Plot Performance Graph.**

**5. Import Data from CSV - This Module is used to Import the data from CSV file. We have to import from same directory.**

**6. Export Data to CSV - This Module is used to Export the data to CSV file. The exported file will saved in same directory.**

**7. Delete Employee – This Module is used to delete the employee from dataset.**

**8. Save data to Dataset – This module help us to save the the data in**

**dataset. It will save in same directory.**

**9. View CSV data – This will help in to view the data from CSV file in the interface.**

**10. Plot Pie Chart – This Module is used to plot the pie chart on the basis of score, behavior, attitude, etc.**

## Coding:

```
4. import matplotlib.pyplot as plt
   from reportlab.lib.units import inch
   from reportlab.lib import colors
   from reportlab.lib.pagesizes import letter, landscape
   from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
   from reportlab.lib.styles import getSampleStyleSheet
   import csv
   import numpy as np
   import seaborn as sns


   class Employee:
       def __init__(self, emp_id, name, salary=0):
           self.emp_id = emp_id
           self.name = name
           self.scores = {
               'goals': [],
               'objectives': [],
               'responsibilities': [],
               'work_quality': [],
               'productivity': [],
               'attitude': []
           }
           self.salary = salary  # New attribute: salary

       def add_score(self, category, score):
           self.scores[category].append(score)

       def calculate_average_score(self, category):
           if not self.scores[category]:
               return 0
           return sum(self.scores[category]) / len(self.scores[category])


   class StaffAppraisalSystem:
       def __init__(self):
           self.employees = []

       def plot_performance_pie_chart(self, emp_id):
           employee = next((emp for emp in self.employees if emp.emp_id ==
   emp_id), None)
           if employee:
               categories = ['goals', 'objectives', 'responsibilities',
   'work_quality', 'productivity', 'attitude']
               scores = [employee.calculate_average_score(category) for
   category in categories]

               # Create a pie chart with labels for each category
               plt.figure(figsize=(8, 8))
               plt.pie(scores, labels=categories, autopct='%1.1f%%',
   startangle=140)
               plt.title(f'Appraisal Scores Distribution for
   {employee.name}')
               plt.show()
           else:
```

```
            print("Employee not found!")

    def add_employee(self, emp_id, name, salary=0):
        if any(emp.emp_id == emp_id for emp in self.employees):
            print(f"Employee with ID {emp_id} already exists. Employee IDs
must be unique.")
        else:
            employee = Employee(emp_id, name, salary)
            self.employees.append(employee)

    def conduct_appraisal(self, emp_id, category, score):
        employee = next((emp for emp in self.employees if emp.emp_id ==
emp_id), None)
        if employee:
            employee.add_score(category, score)
        else:
            print("Employee not found!")

    def generate_performance_report(self, pdf_filename):
        doc = SimpleDocTemplate(pdf_filename, pagesize=landscape(letter))
        story = []

        categories = ['goals', 'objectives', 'responsibilities',
'work_quality', 'productivity', 'attitude']

        data = [['Employee ID', 'Name', 'Salary'] + categories + ['Average
Score']]

        max_avg_score = -1  # Initialize with a low value
        max_avg_score_employee = None

        for employee in self.employees:
            scores = [employee.calculate_average_score(category) for
category in categories]
            average_score = sum(scores) / len(scores)
            data.append([employee.emp_id, employee.name, employee.salary]
+ scores + [f'{average_score:.2f}'])

            # Update the max average score and employee
            if average_score > max_avg_score:
                max_avg_score = average_score
                max_avg_score_employee = employee

        # Calculate bonus for the employee with the highest average score
        if max_avg_score_employee:
            bonus = max_avg_score_employee.salary * 0.10
            max_avg_score_employee.salary += bonus

        table = Table(data, colWidths=[1.5 * inch] * (len(categories) +
4))
        table.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0),
colors.grey),
                                   ('TEXTCOLOR', (0, 0), (-1, 0),
colors.whitesmoke),
                                   ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
                                   ('FONTNAME', (0, 0), (-1, 0),
'Helvetica-Bold'),
                                   ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
```

```
                                        ('BACKGROUND', (0, 1), (-1, -1),
colors.beige),
                                        ('GRID', (0, 0), (-1, -1), 1,
colors.black)]))

        story.append(table)
        doc.build(story)

    def plot_performance_graph(self, category, records_per_page=50):
        employee_names = [employee.name for employee in self.employees]
        average_scores = [employee.calculate_average_score(category) for
employee in self.employees]

        # Split the data into multiple pages
        num_records = len(employee_names)
        num_pages = (num_records + records_per_page - 1) //
records_per_page

        for page in range(num_pages):
            start_idx = page * records_per_page
            end_idx = (page + 1) * records_per_page
            page_employee_names = employee_names[start_idx:end_idx]
            page_average_scores = average_scores[start_idx:end_idx]
            c = np.array(["blue", "green", "red", "yellow", "black",
"green"])
            plt.figure(figsize=(10, 6))
            plt.barh(page_employee_names, page_average_scores, color=c)
            plt.xlabel('Average Score')
            plt.ylabel('Employee Names')
            plt.title(f'Employee Performance in {category.capitalize()}
(Page {page + 1}/{num_pages})')

            plt.tight_layout()
            plt.show()

    def import_data_from_csv(self, csv_filename):
        categories = ['goals', 'objectives', 'responsibilities',
'work_quality', 'productivity', 'attitude']
        new_employees = []

        try:
            with open(csv_filename, 'r') as csv_file:
                csv_reader = csv.DictReader(csv_file)
                for row in csv_reader:
                    emp_id = row['Employee ID']
                    name = row.get('Name', 'Name Missing')
                    salary = float(row.get('Salary', 0))  # Provide a
default salary if missing

                    existing_employee = next((emp for emp in
self.employees if emp.emp_id == emp_id), None)

                    if existing_employee:
                        print(f"Employee with ID {emp_id} already exists.
Updating information.")
                        for category in categories:
                            if category in row:
                                score = float(row[category])
```

```
                                existing_employee.add_score(category,
score)
                            else:
                                print(
                                    f"Missing data for category {category}
for Employee ID {emp_id}. Please re-enter.")
                                return
                    else:
                        employee = Employee(emp_id, name, salary)
                        for category in categories:
                            if category in row:
                                score = float(row[category])
                                employee.add_score(category, score)
                            else:
                                print(
                                    f"Missing data for category {category}
for Employee ID {emp_id}. Please re-enter.")
                                return
                        new_employees.append(employee)

                self.employees.extend(new_employees)
                print(f'Data imported from {csv_filename}')
        except FileNotFoundError:
            print(f'File not found: {csv_filename}')
        except Exception as e:
            print(f'Error importing data from {csv_filename}: {str(e)}')

    def export_data_to_csv(self, csv_filename):
        try:
            with open(csv_filename, 'w', newline='') as csv_file:
                fieldnames = ['Employee ID', 'Name', 'Salary'] +
categories
                writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
                writer.writeheader()
                for employee in self.employees:
                    row = {'Employee ID': employee.emp_id, 'Name':
employee.name, 'Salary': employee.salary}
                    for category in categories:
                        row[category] =
employee.calculate_average_score(category)
                    writer.writerow(row)
                print(f'Data exported to {csv_filename}')
        except Exception as e:
            print(f'Error exporting data to {csv_filename}: {str(e)}')

    def add_employee_from_input(self):
        emp_id = input("Enter Employee ID: ")
        name = input("Enter Employee Name: ")
        salary = float(input("Enter Employee Salary: "))
        self.add_employee(emp_id, name, salary)

    def delete_employee(self, emp_id):
        employee = next((emp for emp in self.employees if emp.emp_id ==
emp_id), None)
        if employee:
            self.employees.remove(employee)
            print(f'Employee with ID {emp_id} deleted.')
        else:
```

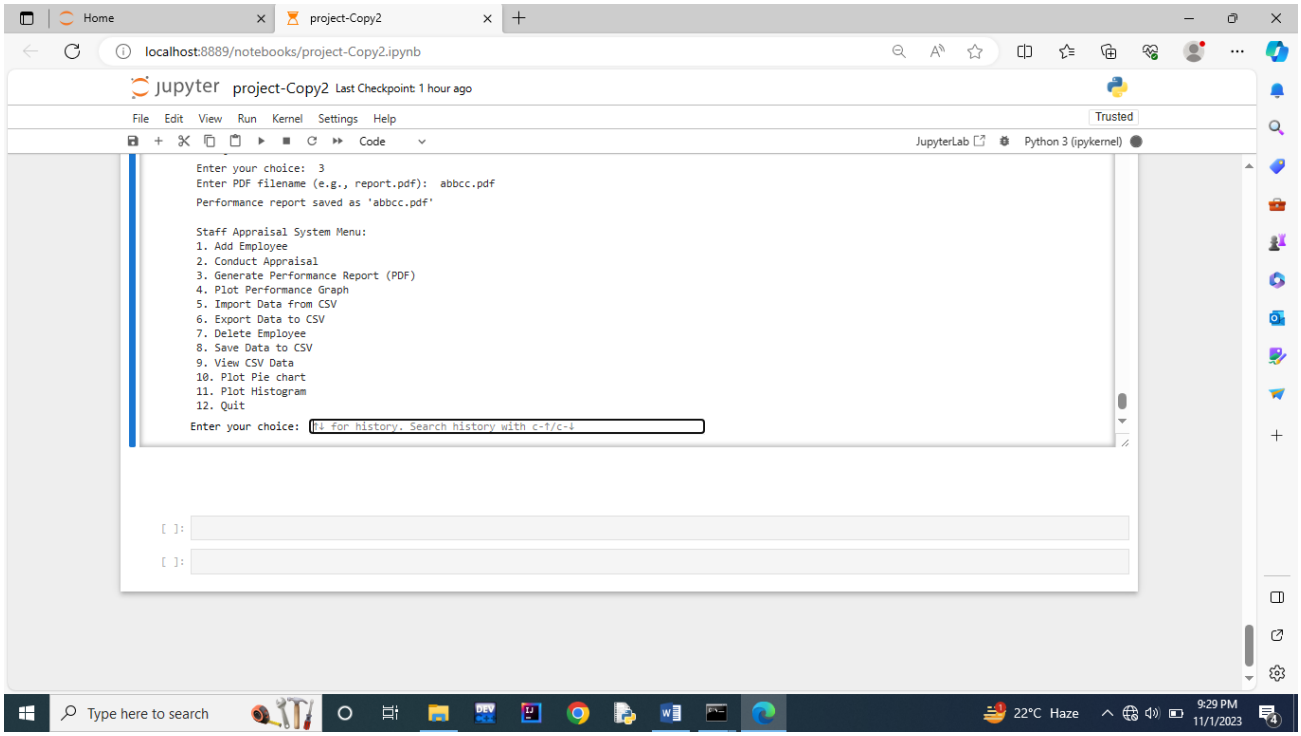
```
            print("Employee not found!")

    def save_data_to_csv(self, csv_filename):
        try:
            with open(csv_filename, 'w', newline='') as csv_file:
                fieldnames = ['Employee ID', 'Name', 'Salary'] +
categories
                writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
                writer.writeheader()
                for employee in self.employees:
                    row = {'Employee ID': employee.emp_id, 'Name':
employee.name, 'Salary': employee.salary}
                    for category in categories:
                        row[category] =
employee.calculate_average_score(category)
                    writer.writerow(row)
                print(f'Data saved to {csv_filename}')
        except Exception as e:
            print(f'Error saving data to {csv_filename}: {str(e)}')

    def view_csv_data(self, csv_filename, categories):
        try:
            with open(csv_filename, 'r') as csv_file:
                csv_reader = csv.DictReader(csv_file)
                for row in csv_reader:
                    data = ', '.join([f'{category}: {row[category]}' for
category in categories])
                    print(f"Employee ID: {row['Employee ID']}, Name:
{row['Name']}, Salary: {row['Salary']}, {data}")
                    print()
            print()
        except FileNotFoundError:
            print(f'File not found: {csv_filename}')
        except Exception as e:
            print(f'Error viewing data from {csv_filename}: {str(e)}')

    def plot_performance_scatter(self, category_x, category_y):
        valid_categories = set(categories)

        if category_x not in valid_categories or category_y not in
valid_categories:
            print("Invalid categories. Please enter valid categories.")

        x_values = [employee.calculate_average_score(category_x) for
employee in self.employees]
        y_values = [employee.calculate_average_score(category_y) for
employee in self.employees]

        plt.figure(figsize=(8, 6))
        plt.scatter(x_values, y_values, c='b', marker='o')
        plt.xlabel(f'Average Score in {category_x}')
        plt.ylabel(f'Average Score in {category_y}')
        plt.title(f'Scatter Plot: {category_x.capitalize()} vs.
{category_y.capitalize()}')
        plt.show()

    def plot_performance_histogram(self, category):
        scores = [employee.calculate_average_score(category) for employee
```

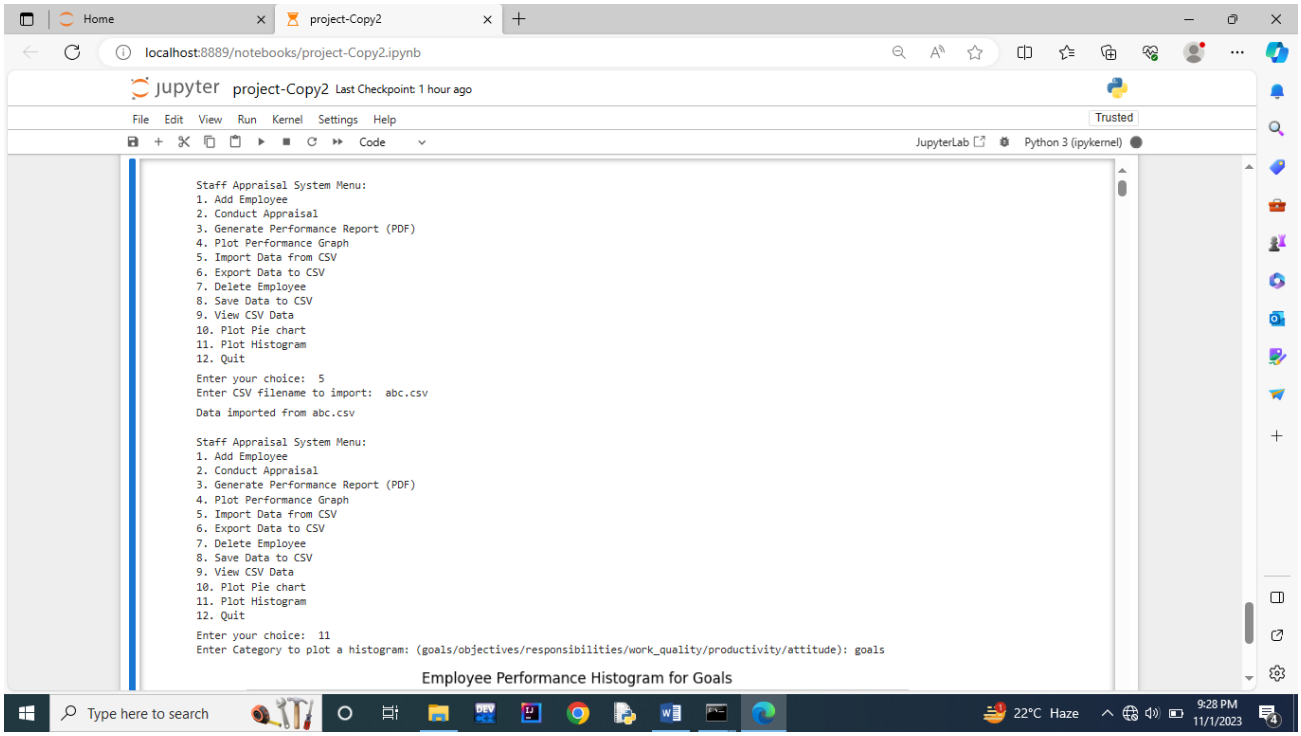
```
in self.employees]

        plt.figure(figsize=(10, 6))
        plt.hist(scores, bins=10, color='blue', edgecolor='black')
        plt.xlabel(f'Average Score in {category.capitalize()}')
        plt.ylabel('Number of Employees')
        plt.title(f'Employee Performance Histogram for
{category.capitalize()}')
        plt.show()

    def plot_heatmap(self):
        categories = ['goals', 'objectives', 'responsibilities',
'work_quality', 'productivity', 'attitude']
        data = [[employee.calculate_average_score(category) for category
in categories] for employee in self.employees]

        # Create a heatmap using matplotlib
        fig, ax = plt.subplots(figsize=(10, 6))
        im = ax.imshow(data, cmap='coolwarm')

        # Show all ticks and labels
        ax.set_xticks(np.arange(len(categories)))
        ax.set_yticks(np.arange(len(self.employees)))
        ax.set_xticklabels(categories)
        ax.set_yticklabels([employee.name for employee in self.employees])

        # Rotate the tick labels and set their alignment
        plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
rotation_mode="anchor")

        # Create colorbar
        cbar = ax.figure.colorbar(im, ax=ax, cmap='coolwarm')
        cbar.ax.set_ylabel('Average Score', rotation=-90, va="bottom")

        # Display the heatmap
        plt.title('Average Appraisal Scores Heatmap')
        plt.tight_layout()
        plt.show()


def main():
    appraisal_system = StaffAppraisalSystem()
    categories = ['goals', 'objectives', 'responsibilities',
'work_quality', 'productivity', 'attitude']

    while True:
        print("\nStaff Appraisal System Menu:")
        print("1. Add Employee")
        print("2. Conduct Appraisal")
        print("3. Generate Performance Report (PDF)")
        print("4. Plot Performance Graph")
        print("5. Import Data from CSV")
        print("6. Export Data to CSV")
        print("7. Delete Employee")
        print("8. Save Data to CSV")
        print("9. View CSV Data")
        print("10. Plot Pie chart")
        print("11. Plot Histogram ")
```

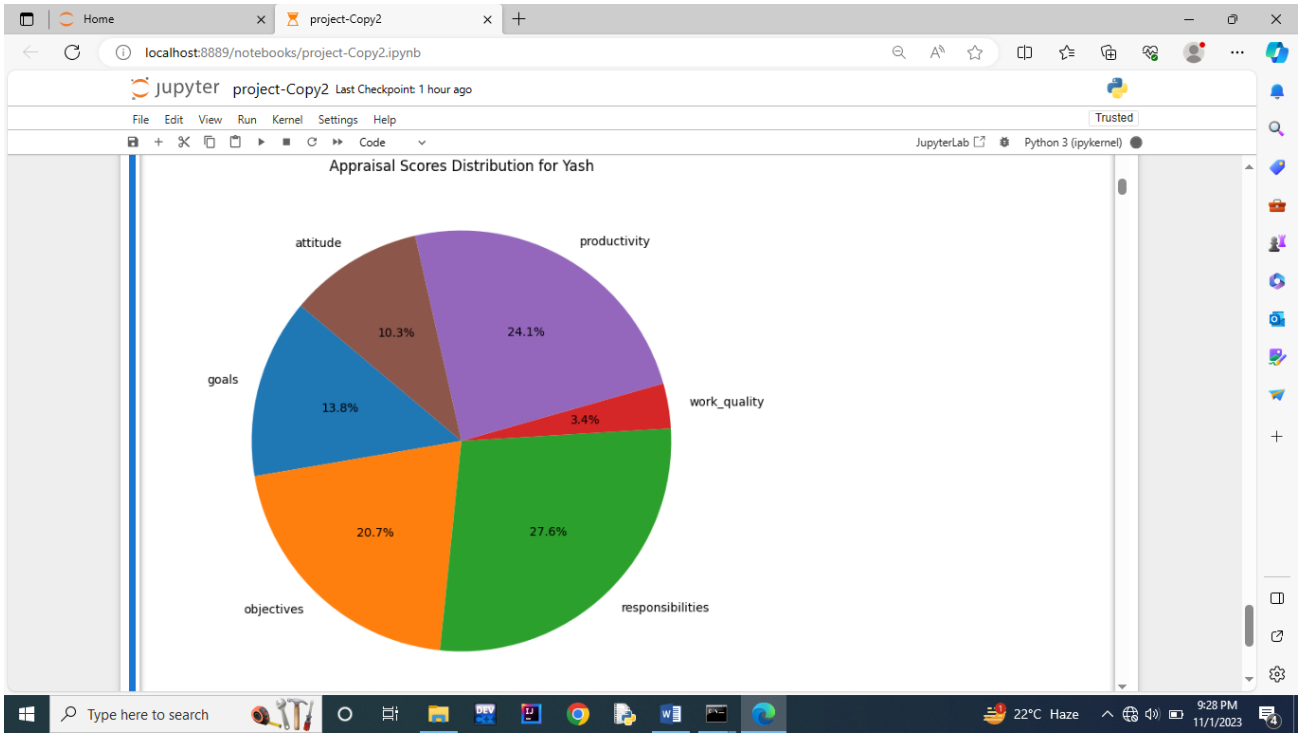
```
        print("12. Quit")

        choice = input("Enter your choice: ")

        if choice == "1":
            appraisal_system.add_employee_from_input()
        elif choice == "2":
            emp_id = input("Enter Employee ID: ")
            category = input(
                "Enter Category
(goals/objectives/responsibilities/work_quality/productivity/attitude):
").lower()
            if category in categories:
                score = float(input("Enter Appraisal Score: "))
                appraisal_system.conduct_appraisal(emp_id, category,
score)
            else:
                print("Invalid category. Please try again.")
        elif choice == "3":
            pdf_filename = input("Enter PDF filename (e.g., report.pdf):
")
            appraisal_system.generate_performance_report(pdf_filename)
            print(f"Performance report saved as '{pdf_filename}'")
        elif choice == "4":
            category = input(
                "Enter Category to plot
(goals/objectives/responsibilities/work_quality/productivity/attitude):
").lower()
            if category in categories:
                appraisal_system.plot_performance_graph(category)
            else:
                print("Invalid category. Please try again.")
        elif choice == "5":
            csv_filename = input("Enter CSV filename to import: ")
            appraisal_system.import_data_from_csv(csv_filename)
        elif choice == "6":
            csv_filename = input("Enter CSV filename to export: ")
            appraisal_system.export_data_to_csv(csv_filename)
        elif choice == "7":
            emp_id = input("Enter Employee ID to delete: ")
            appraisal_system.delete_employee(emp_id)
        elif choice == "8":
            csv_filename = input("Enter CSV filename to save: ")
            appraisal_system.save_data_to_csv(csv_filename)
        elif choice == "9":
            csv_filename = input("Enter CSV filename to view: ")
            appraisal_system.view_csv_data(csv_filename, categories)
        elif choice == "10":
            emp_id = input("Enter Employee ID to plot a pie chart: ")
            appraisal_system.plot_performance_pie_chart(emp_id)  # Option
to plot pie chart
        elif choice == "11":
            category = input(
                "Enter Category to plot a histogram:
(goals/objectives/responsibilities/work_quality/productivity/attitude):").
lower()
            if category in categories:
                appraisal_system.plot_performance_histogram(category)  #
```

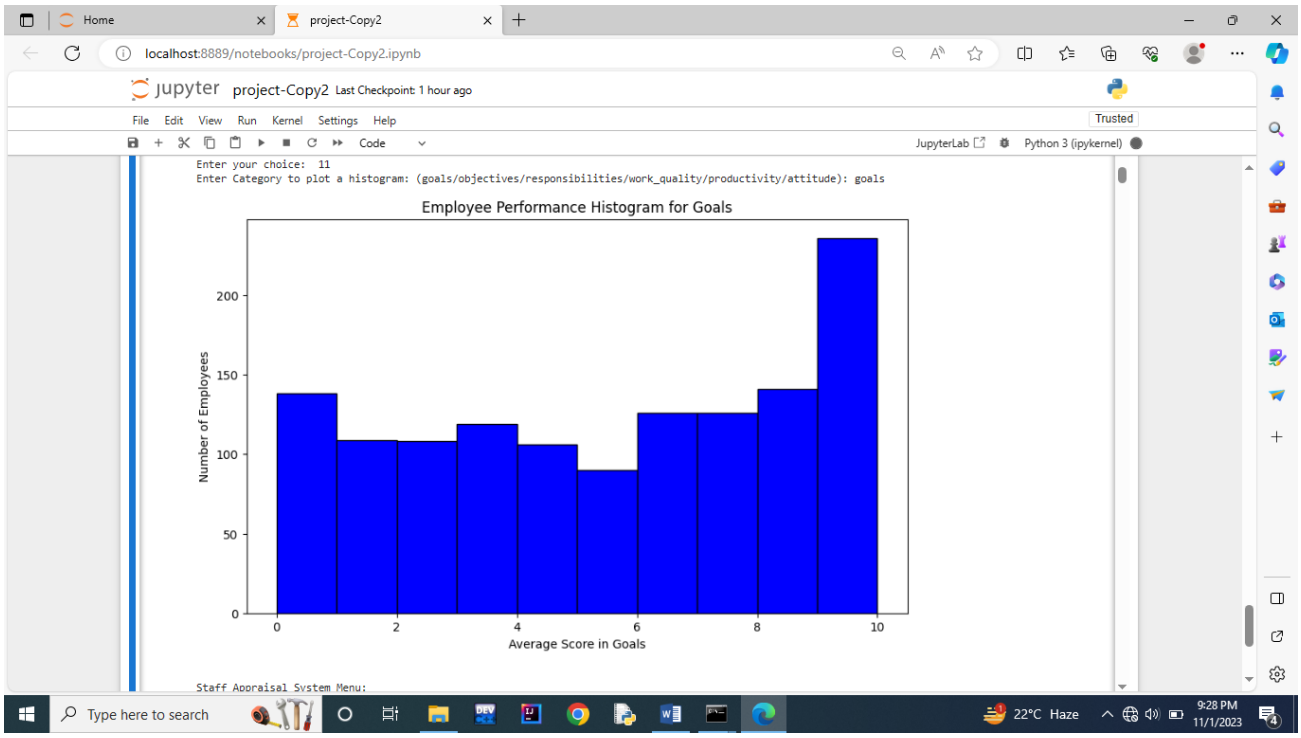
```
Option to plot histogram
            else:
                print("Invalid category. Please try again.")
        elif choice == "12":
            break
        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    main()
```

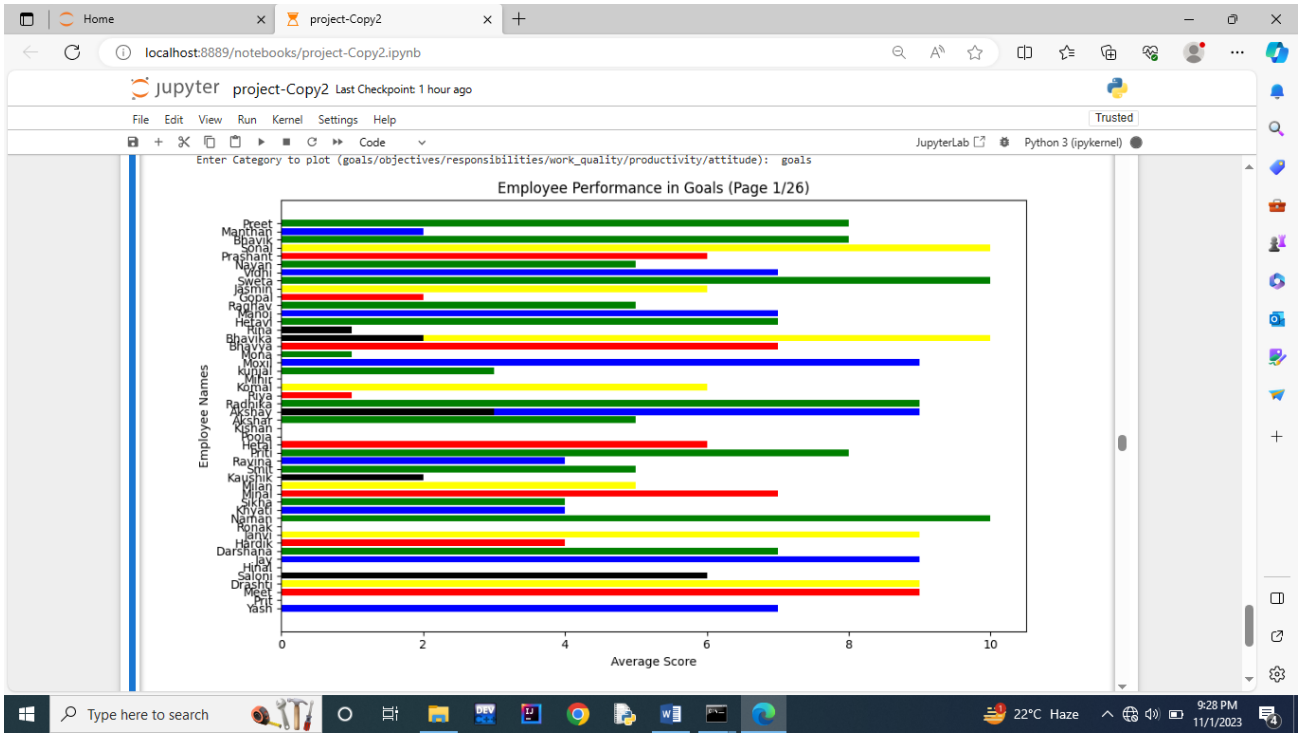
# Screenshot Output:

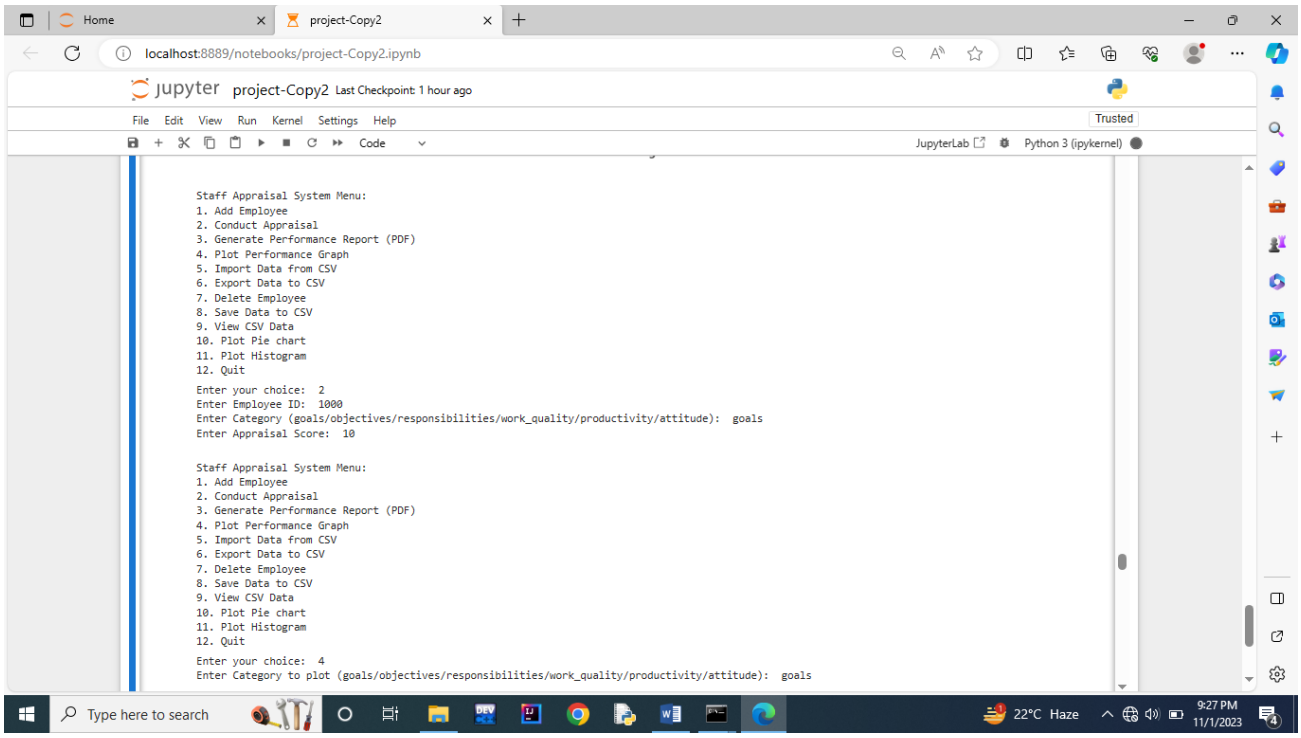


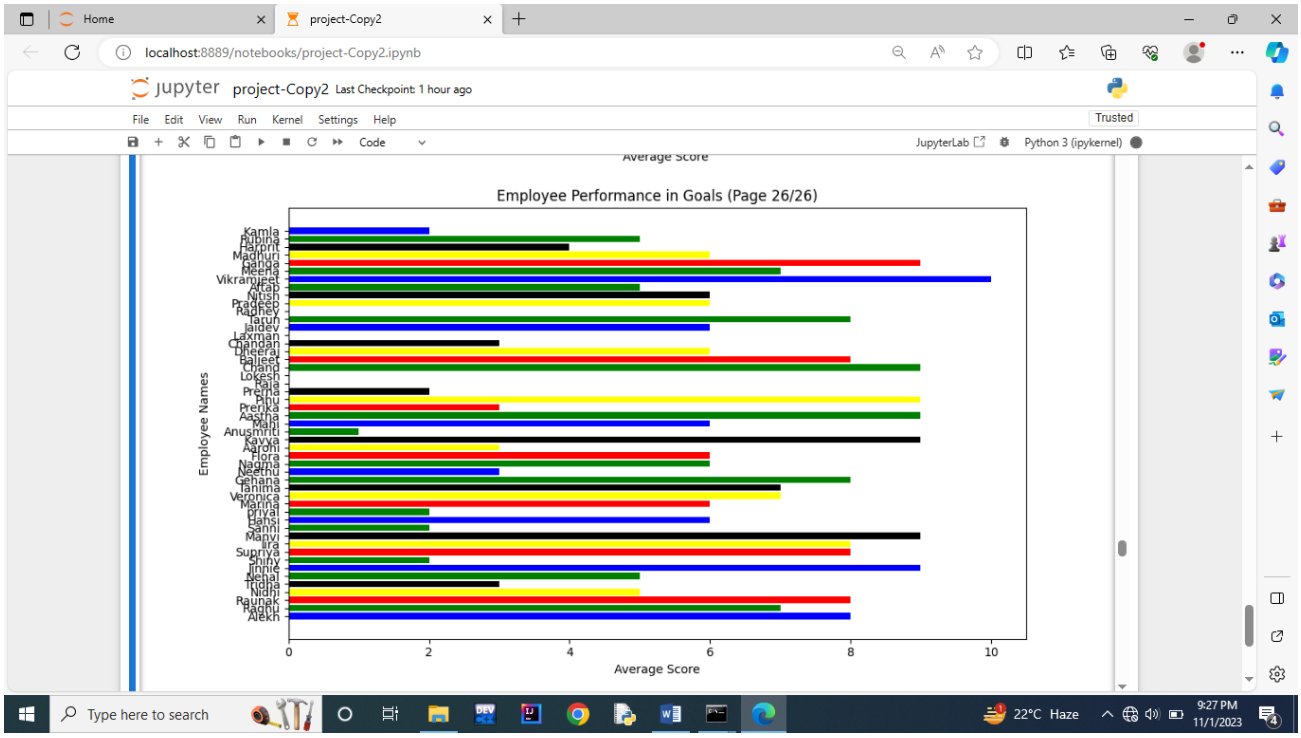




Appraisal Scores Distribution for Yash
attitude
productivity
goals
work_quality
objectives
responsibilities
10.3%
24.1%
13.8%
3.4%
20.7%
27.6%


Enter your choice: 11
Enter Category to plot a histogram: (goals/objectives/responsibilities/work_quality/productivity/attitude): goals
Employee Performance Histogram for Goals
Number of Employees
Average Score in Goals
Staff Appraisal System Menu:


Home
project-Copy2
localhost:8889/notebooks/project-Copy2.ipynb
jupyter project-Copy2 Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
Enter Category to plot (goals/objectives/responsibilities/work_quality/productivity/attitude): goals
Employee Performance in Goals (Page 1/26)
Employee Names
Average Score
0
2
4
6
8
10
Type here to search
22°C Haze
9:28 PM
11/1/2023


Home
project-Copy2
localhost:8889/notebooks/project-Copy2.ipynb
jupyter project-Copy2 Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
Staff Appraisal System Menu:
1. Add Employee
2. Conduct Appraisal
3. Generate Performance Report (PDF)
4. Plot Performance Graph
5. Import Data from CSV
6. Export Data to CSV
7. Delete Employee
8. Save Data to CSV
9. View CSV Data
10. Plot Pie chart
11. Plot Histogram
12. Quit
Enter your choice: 2
Enter Employee ID: 1000
Enter Category (goals/objectives/responsibilities/work_quality/productivity/attitude): goals
Enter Appraisal Score: 10
Staff Appraisal System Menu:
1. Add Employee
2. Conduct Appraisal
3. Generate Performance Report (PDF)
4. Plot Performance Graph
5. Import Data from CSV
6. Export Data to CSV
7. Delete Employee
8. Save Data to CSV
9. View CSV Data
10. Plot Pie chart
11. Plot Histogram
12. Quit
Enter your choice: 4
Enter Category to plot (goals/objectives/responsibilities/work_quality/productivity/attitude): goals
Type here to search
22°C Haze
9:27 PM
11/1/2023


Home
project-Copy2
localhost:8889/notebooks/project-Copy2.ipynb
jupyter project-Copy2 Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
Average Score
Employee Performance in Goals (Page 26/26)
Employee Names
Kamla
Rubina
Taporit
Madhuri
Ganga
Meena
Vikramjeet
Akash
Nitish
Pradeep
Raghev
Tarun
Jaidev
Laxman
Chandan
Dheeraj
Balleet
Chand
Lokesh
Raja
Prerna
Bhu
Prerka
Aastha
Mani
Anushriti
Kavya
Aarohi
Flora
Nagma
Neethu
Gehana
Tanima
Veronica
Marina
Gryal
Bansi
Janni
Manvi
Ira
Supriya
Shiny
Tinnit
Benat
Tridha
Nidhi
Raunak
Raghu
Alekh
0
2
4
6
8
10
Average Score
Type here to search
22°C Haze
9:27 PM
11/1/2023