

Data Analysis

Data in csv format followed by a graphical representation:

ID,Name,Age,Position,Department,Salary,Joining Date,Performance Score

1,Alice Smith,28,Software Engineer,IT,75000,2022-01-15,85

2,Bob Johnson,35,Data Scientist,Data,90000,2019-05-10,90

3,Charlie Brown,40,Manager,HR,95000,2018-03-20,88

4,David Williams,29,UX Designer,Design,70000,2021-09-01,80

5,Eva Davis,32,Database Admin,IT,78000,2020-02-25,82

6,Frank Miller,45,CTO,Executive,150000,2017-06-15,92

7,Grace Lee,38,Product Manager,Product,85000,2019-11-05,87

8,Hank Garcia,30,Data Engineer,Data,82000,2021-04-10,84

9,Ivy Wilson,27,Frontend Developer,IT,72000,2022-03-01,81

10,Jack White,42,CEO,Executive,200000,2016-07-20,95

ID	Name	Age	Position	Department	Salary	Joining Date	Performance Score
1	Alice Smith	28	Software Engineer	IT	75000	2022-01-15	85
2	Bob Johnson	35	Data Scientist	Data	90000	2019-05-10	90
3	Charlie Brown	40	Manager	HR	95000	2018-03-20	88
4	David Williams	29	UX Designer	Design	70000	2021-09-01	80
5	Eva Davis	32	Database Admin	IT	78000	2020-02-25	82
6	Frank Miller	45	CTO	Executive	150000	2017-06-15	92
7	Grace Lee	38	Product Manager	Product	85000	2019-11-05	87
8	Hank Garcia	30	Data Engineer	Data	82000	2021-04-10	84
9	Ivy Wilson	27	Frontend Developer	IT	72000	2022-03-01	81
10	Jack White	42	CEO	Executive	200000	2016-07-20	95

Prompt(s):

Employee Distribution: Write a code to analyze the distribution of employees across different departments. How many employees does each department have?

Salary Analysis: Can you provide a code that calculates the average salary of employees in the IT department compared to the Data department?

Performance Metrics: Generate a code that identifies employees with a performance score above 90. Are there any patterns or insights you can derive from these top performers?

Tenure Analysis: Write a code to determine the average tenure (time since joining) of employees in the company. How does this compare between departments?

Age Group Analysis: Can you segment the employees into age groups (e.g., 20-30, 30-40, etc.) and analyze the average performance score within each group?

GPT-4

Employee Distribution:

```
# Count the number of employees in each department
department_distribution = df['Department'].value_counts()
print(department_distribution)
```

Salary Analysis:

```
# Calculate average salary for IT and Data departments
avg_salary_IT = df[df['Department'] == 'IT']['Salary'].mean()
avg_salary_Data = df[df['Department'] == 'Data']['Salary'].mean()

print(f"Average salary in IT department: ${avg_salary_IT:.2f}")
print(f"Average salary in Data department: ${avg_salary_Data:.2f}")
```

Performance Metrics:

```
# Filter employees with performance score above 90
top_performers = df[df['Performance Score'] > 90]

print(top_performers)
```

Tenure Analysis:

```
from datetime import datetime

# Calculate tenure in years
current_date = datetime.now()
df['Tenure'] = (current_date - pd.to_datetime(df['Joining Date'])).dt.days / 365

# Calculate average tenure for the entire company and by department
avg_tenure = df['Tenure'].mean()
avg_tenure_by_department = df.groupby('Department')['Tenure'].mean()

print(f"Average tenure in the company: {avg_tenure:.2f} years")
print(avg_tenure_by_department)
```

Age Group Analysis:

```
# Define age bins
bins = [20, 30, 40, 50]
labels = ['20-30', '30-40', '40-50']
df['Age Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

# Calculate average performance score by age group
avg_performance_by_age_group = df.groupby('Age Group')['Performance Score'].mean()

print(avg_performance_by_age_group)
```

GPT-3.5

Employee Distribution:

```
# Prompt 1: Employee Distribution
department_counts = {}
for employee in employee_list:
    department = employee.department
```

```

department_counts[department] = department_counts.get(department, 0) + 1

print("Prompt 1: Employee Distribution")
for department, count in department_counts.items():
    print(f"{department}: {count} employees")

Salary Analysis:
# Prompt 2: Salary Analysis
it_employees = [employee for employee in employee_list if employee.department == 'IT']
data_employees = [employee for employee in employee_list if employee.department == 'Data']

average_salary_it = sum(employee.salary for employee in it_employees) / len(it_employees) if
it_employees else 0
average_salary_data = sum(employee.salary for employee in data_employees) / len(data_employees) if
data_employees else 0

print("\nPrompt 2: Salary Analysis")
print(f"Average Salary in IT: {average_salary_it}")
print(f"Average Salary in Data: {average_salary_data}")

Performance Metrics:
# Prompt 3: Performance Metrics
top_performers = [employee for employee in employee_list if employee.performance_score > 90]

print("\nPrompt 3: Performance Metrics - Employees with Performance Score > 90")
for employee in top_performers:
    print(f"ID: {employee.id}, Name: {employee.name}, Performance Score:
{employee.performance_score}")

Tenure Analysis:
# Prompt 4: Tenure Analysis
department_tenures = {}
for employee in employee_list:
    department = employee.department
    if department not in department_tenures:
        department_tenures[department] = []
    tenure = (datetime.now() - employee.joining_date).days / 365.0
    department_tenures[department].append(tenure)

print("\nPrompt 4: Tenure Analysis - Average Tenure in Years by Department")
for department, tenures in department_tenures.items():
    average_tenure = sum(tenures) / len(tenures)
    print(f"{department}: {average_tenure:.2f} years")

Age Group Analysis:
# Prompt 5: Age Group Analysis
age_group_scores = {'20-30': [], '30-40': [], '40+': []}
for employee in employee_list:
    age = employee.age
    if 20 <= age < 30:
        age_group_scores['20-30'].append(employee.performance_score)
    elif 30 <= age < 40:
        age_group_scores['30-40'].append(employee.performance_score)
    else:
        age_group_scores['40+'].append(employee.performance_score)

print("\nPrompt 5: Age Group Analysis - Average Performance Score by Age Group")
for age_group, scores in age_group_scores.items():
    average_score = sum(scores) / len(scores) if scores else 0

```

```
print(f'{age_group}: {average_score:.2f}')
```

BingAI

Employee Distribution:

```
# Employee Distribution
print(df['Department'].value_counts())
```

Salary Analysis:

```
it_avg_salary = df[df['Department'] == 'IT']['Salary'].mean()
data_avg_salary = df[df['Department'] == 'Data']['Salary'].mean()
print(f'Average IT Salary: {it_avg_salary}')
print(f'Average Data Salary: {data_avg_salary}')
```

Performance Metrics:

```
# Performance Metrics
top_performers = df[df['Performance Score'] > 90]
print(top_performers)
```

Tenure Analysis:

```
# Tenure Analysis
df['Joining Date'] = pd.to_datetime(df['Joining Date'])
df['Tenure'] = (datetime.now() - df['Joining Date']).dt.days / 365.25 # in years
avg_tenure = df['Tenure'].mean()
print(f'Average Tenure: {avg_tenure} years')
```

Age Group Analysis:

```
# Age Group Analysis
bins = [20+i*10 for i in range(5)] # 20-30,...
df['Age Group'] = pd.cut(df['Age'], bins=bins)
age_group_performance = df.groupby('Age Group')['Performance Score'].mean()
print(age_group_performance)
```

You

Employee Distribution:

```
import csv

def analyze_employee_distribution(file_path):
    department_count = {}

    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)

        for row in reader:
            department = row['Department']
            if department not in department_count:
                department_count[department] = 1
            else:
                department_count[department] += 1

    return department_count

# Usage example:
file_path = 'employees.csv'
result = analyze_employee_distribution(file_path)
print(result)
```

Salary Analysis:

```
import csv

def calculate_average_salary(file_path, department):
    total_salary = 0
    employee_count = 0

    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)

        for row in reader:
            if row['Department'] == department:
                total_salary += int(row['Salary'])
                employee_count += 1

    if employee_count > 0:
        average_salary = total_salary / employee_count
        return average_salary
    else:
        return None

# Usage example:
file_path = 'employees.csv'
it_average_salary = calculate_average_salary(file_path, 'IT')
data_average_salary = calculate_average_salary(file_path, 'Data')

print(f"Average salary in IT department: {it_average_salary}")
print(f"Average salary in Data department: {data_average_salary}")
```

Performance Metrics:

```
import csv

def identify_top_performers(file_path, score_threshold):
    top_performers = []

    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)

        for row in reader:
            performance_score = int(row['Performance Score'])
            if performance_score > score_threshold:
                top_performers.append(row)

    return top_performers

# Usage example:
file_path = 'employees.csv'
score_threshold = 90
top_performers = identify_top_performers(file_path, score_threshold)

for performer in top_performers:
    print(performer['Name'], performer['Department'])
```

Tenure Analysis:

```
import csv
from datetime import datetime

def calculate_tenure(joining_date):
    today = datetime.today().date()
```

```

joining_date = datetime.strptime(joining_date, '%Y-%m-%d').date()
tenure = (today - joining_date).days / 365.25 # Assuming a leap year average of 365.25 days
return tenure

def calculate_average_tenure(file_path, department):
    total_tenure = 0
    employee_count = 0

    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)

        for row in reader:
            if row['Department'] == department:
                tenure = calculate_tenure(row['Joining Date'])
                total_tenure += tenure
                employee_count += 1

    if employee_count > 0:
        average_tenure = total_tenure / employee_count
        return average_tenure
    else:
        return None

# Usage example:
file_path = 'employees.csv'
it_average_tenure = calculate_average_tenure(file_path, 'IT')
data_average_tenure = calculate_average_tenure(file_path, 'Data')

print(f"Average tenure in IT department: {it_average_tenure} years")
print(f"Average tenure in Data department: {data_average_tenure} years")

```

Age Group Analysis:

```

import csv

def analyze_age_group_performance(file_path):
    age_ranges = {'20-30': (20, 30), '30-40': (30, 40), '40-50': (40, 50)}
    age_group_scores = {age_range: [] for age_range in age_ranges}

    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)

        for row in reader:
            age = int(row['Age'])
            performance_score = int(row['Performance Score'])

            for age_range, (min_age, max_age) in age_ranges.items():
                if min_age <= age < max_age:
                    age_group_scores[age_range].append(performance_score)

    average_scores = {age_range: sum(scores) / len(scores) if len(scores) > 0 else None
                      for age_range, scores in age_group_scores.items()}

    return average_scores

# Usage example:
file_path = 'employees.csv'
age_group_scores = analyze_age_group_performance(file_path)

for age_range, average_score in age_group_scores.items():

```

```
print(f"Average performance score for age group {age_range}: {average_score}")
```

Google Bard

Employee Distribution:

```
# Employee Distribution
print('Employee distribution across different departments:')
print(df['Department'].value_counts())
```

Salary Analysis:

```
# Salary Analysis
it_salary_avg = df[df['Department'] == 'IT']['Salary'].mean()
data_salary_avg = df[df['Department'] == 'Data']['Salary'].mean()
print('Average salary in IT department:', it_salary_avg)
print('Average salary in Data department:', data_salary_avg)
```

Performance Metrics:

```
# Performance Metrics
top_performers = df[df['Performance Score'] > 90]
print('Employees with performance score above 90:')
print(top_performers.to_string())
```

Tenure Analysis:

```
# Tenure Analysis
average_tenure = df['Joining Date'].apply(lambda x: pd.Timestamp.today() - pd.Timestamp(x)).mean()
print('Average tenure of employees in the company:', average_tenure)
```

Age Group Analysis:

```
# Read the CSV file into a Pandas DataFrame
df = pd.read_csv('employees.csv')
```

Age Group Analysis

```
age_groups = {
    '20-30': df['Age'].between(20, 30),
    '30-40': df['Age'].between(30, 40),
    '40-50': df['Age'].between(40, 50),
}
```

Convert the age_groups dictionary to a list of Pandas Series objects

```
age_group_series = [pd.Series(group, name=name) for name, group in age_groups.items()]
```

Group the DataFrame by age group and calculate the average performance score

```
age_group_performance = df.groupby(age_group_series)['Performance Score'].mean()
```

Print the average performance score by age group

```
print('Average performance score by age group:')
print(age_group_performance)
```