PROJECT TITLE:

**Employee Data Cleaning and Analysis using Python (Pandas & Numpy)**

PROJECT DESCRIPTION :

This project focuses on cleaning, standardizing, and analysing employee data using Python libraries such as Pandas and Datetime. The dataset contains details of employees, including EmpID, Name, Job Title, Department, Salary, and Joining Date. The project performs data cleaning, removes duplicates, handles missing values, standardizes fields, and generates summary reports. Additionally, it calculates Years of Service and identifies the highest-paid employees in each department.

TEAM MEMBER:

1. NAME = PRARTHANA N

2.USN = 4GW23CI039

3.BRANCH = CSE-(AIML)

4.SEMESTER = 5

INDEX:

1. Project Description

2. Dataset Overview

3. Data Cleaning Process

4. Feature Implementation

5. UML Diagrams

8. Code Implementation

9. Code Explanation

10. Output with Screenshot

11. Conclusion / Bibliography

DETAILED EXPLANATION :

DATASET SOURCE:

The dataset used is employee.csv, which contains employee details such as EmpID, Name, Job Title, Department, Salary, and Joining Date.

STEPS IMPLEMENT :

1. Loading Dataset – The dataset is loaded using Pandas.

2. Inspection – Basic information and description of the dataset are displayed.

3. Removing Duplicates – Duplicate employee IDs are removed.

4. Handling Missing Values – Missing Salary values are filled with the median salary, and missing Job Titles are filled with “Unknown.”

5. Standardization – Department and Job Titles are standardized for consistency.6. Feature Creation – A new column Years Of Service is added using the Joining Date.7. Aggregation – Department-wise and Job Title-wise summaries are created.

8. Highest-Paid Employee – The highest-paid employee from each department is identified.

9. Export – Results are exported as separate CSV summary files.

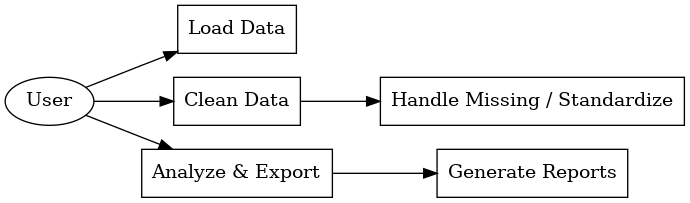
UML DIAGRAM:

**1. Use Case Diagram**

**Purpose**: Shows what the **user** expects the system to do.

* **Actors**: User (you or whoever runs the script).
* **Use Cases**:
  + **Load Data** → Reads employee.csv into a DataFrame.
  + **Clean Data** → Removes duplicates, fills missing values, standardizes departments and job titles.
  + **Analyze & Export** → Generates summaries and exports them.
* **Extensions**:
  + Handling missing values and standardizing titles happen as part of "Clean Data".
  + Generating department/job summaries & highest-paid report happen inside "Analyze & Export".

👉 This diagram answers: *What does the system provide to the user?*



**2. Activity Diagram**

**Purpose**: Describes the **workflow** (step-by-step execution).

* Starts with **Load CSV** → Inspect data → Drop duplicates.
* Then moves to **Handle Missing Data** → **Standardize Titles** → **Compute YearsOfService**.
* Next, it performs **Aggregations** (Department → Job Title → Highest Paid).
* Ends with **Export CSV + Print Results**.

👉 This diagram answers: *In what order does the script process the data?*

*A diagram of a service

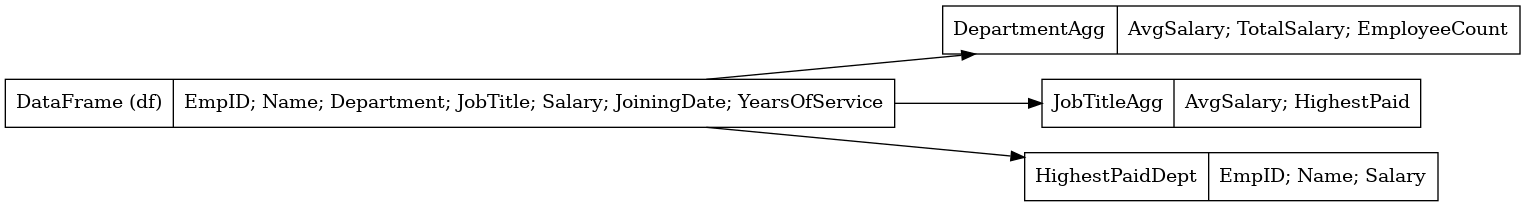
AI-generated content may be incorrect.*

**3. Class Diagram (Conceptual)**

**Purpose**: Even though your script isn’t object-oriented, we can still treat each **data entity** as a class-like structure.

* **DataFrame (df)**: Represents raw employee data with attributes like EmpID, Name, Salary, etc.
* **DepartmentAgg**: Aggregated results for each department (AvgSalary, TotalSalary, EmployeeCount).
* **JobTitleAgg**: Aggregated results for each job title (AvgSalary, HighestPaid).
* **HighestPaidDept**: Special table holding the highest-paid employee in each department.

👉 This diagram answers: *What main entities/data structures are manipulated?*

**

**4. Sequence Diagram**

**Purpose**: Shows the **interaction over time** between user, script, and libraries.

* **User** → runs the script.
* **Script** → calls Pandas to load CSV, clean data, perform aggregations, and export results.
* **Tabulate** → used for pretty printing results to console.
* **User** → gets the final output.

👉 This diagram answers: *Who talks to whom, and in what order?*

*A diagram of a software development process

AI-generated content may be incorrect.*

**5. Component Diagram**

**Purpose**: Shows the **modules and dependencies**.

* **employee.csv** → Input data source.
* **Main Script** → Orchestrates everything.
* **Pandas** → Handles data loading, cleaning, aggregation.
* **Datetime** → Calculates YearsOfService.
* **Tabulate** → Produces formatted output.

👉 This diagram answers: *Which external components/libraries are involved?*

*A diagram of a script

AI-generated content may be incorrect.*

✅ Together, these diagrams give you multiple perspectives:

* **Use Case** = *User view*
* **Activity** = *Workflow view*
* **Class** = *Data/Entity view*
* **Sequence** = *Interaction view*
* **Component** = *System architecture view*

CODE:

pandas as pd

from datetime import datetime

# 1. Load dataset

df = pd.read\_csv('employee import.csv')

# 2. Inspect dataset

print("\n========== Dataset Info ==========")

print(df.info())

print("\n========== Dataset Description ==========")

print(df.describe(include='all'))

# 3. Remove duplicates

df = df.drop\_duplicates(subset=['EmpID'])

# 4. Handle missing salaries/job titles (simulate: fill with median salary, 'Unknown' job title)

df['Salary'] = pd.to\_numeric(df['Salary'], errors='coerce')

df['Salary'].fillna(df['Salary'].median(), inplace=True)

df['JobTitle'].fillna('Unknown', inplace=True)

# 5. Standardize department and job titles

dept\_map = {

    'HR': 'HR',

    'Finance': 'Finance',

    'IT': 'IT',

    'Sales': 'Sales'

}

job\_map = {

    'Software Engg': 'Software Engineer',

    'Software Engineer': 'Software Engineer',

    'Data Scientist': 'Data Scientist',

    'Accountant': 'Accountant',

    'Financial Analyst': 'Financial Analyst',

    'Senior Accountant': 'Senior Accountant',

    'System Administrator': 'System Administrator',

    'Sales Executive': 'Sales Executive',

    'Sales Manager': 'Sales Manager',

    'HR Manager': 'HR Manager',

    'HR Executive': 'HR Executive',

    'Recruiter': 'Recruiter'

}

df['Department'] = df['Department'].map(dept\_map).fillna(df['Department'])

df['JobTitle'] = df['JobTitle'].map(job\_map).fillna(df['JobTitle'])

# 6. Add YearsOfService column

today = datetime.today()

df['JoiningDate'] = pd.to\_datetime(df['JoiningDate'], errors='coerce')

df['YearsOfService'] = ((today - df['JoiningDate']).dt.days / 365).round(1)

# 7. Department Aggregations

dept\_summary = df.groupby('Department').agg(

    AvgSalary=('Salary', 'mean'),

    TotalSalary=('Salary', 'sum'),

    EmployeeCount=('EmpID', 'count')

).reset\_index()

# 8. Job Title Aggregations

job\_summary = df.groupby('JobTitle').agg(

    AvgSalary=('Salary', 'mean'),

    HighestPaid=('Salary', 'max')

).reset\_index()

# Highest-paid employee per department

highest\_paid = df.loc[df.groupby('Department')['Salary'].idxmax()][['Department', 'EmpID', 'Name', 'Salary']]

# 9. Export summaries

dept\_summary.to\_csv('dept\_summary.csv', index=False)

job\_summary.to\_csv('job\_summary.csv', index=False)

highest\_paid.to\_csv('highest\_paid\_per\_dept.csv', index=False)

# 10. Beautiful output using tabulate

try:

    from tabulate import tabulate

    print("\n========== Department Summary ==========")

    print(tabulate(dept\_summary, headers='keys', tablefmt='fancy\_grid', showindex=False))

    print("\n========== Job Title Summary ==========")

    print(tabulate(job\_summary, headers='keys', tablefmt='fancy\_grid', showindex=False))

    print("\n========== Highest Paid Employee Per Department ==========")

    print(tabulate(highest\_paid, headers='keys', tablefmt='fancy\_grid', showindex=False))

except ImportError:

    print("\nInstall 'tabulate' for beautiful tables: pip install tabulate")

    print("\nDepartment Summary:\n", dept\_summary)

    print("\nJob Title Summary:\n", job\_summary)

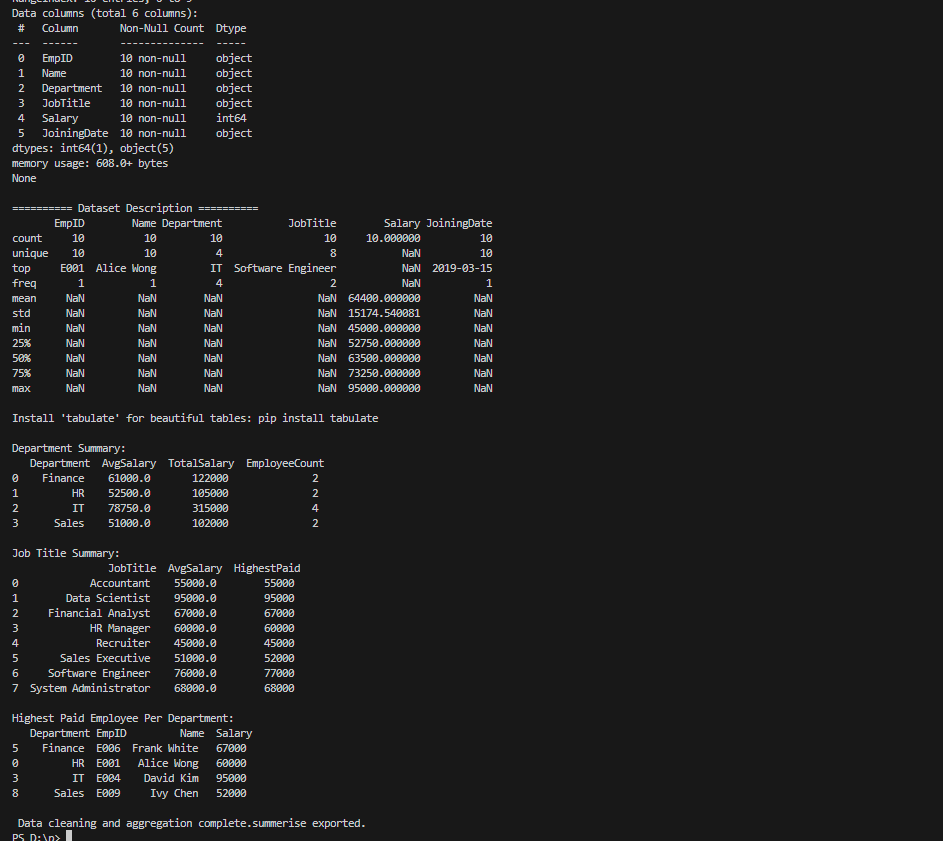
    print("\nHighest Paid Employee Per Department:\n", highest\_paid)

print("\n Data cleaning and aggregation complete.summerise exported.")

EXPLAINATION OF THE CODE:

1. **Import Libraries**
   * [pandas](vscode-file://vscode-app/c:/Users/prath/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) for data manipulation.
   * [datetime](vscode-file://vscode-app/c:/Users/prath/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) for date calculations.
2. **Load Dataset**
   * Reads [employee.csv](vscode-file://vscode-app/c:/Users/prath/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) into a DataFrame.
3. **Inspect Dataset**
   * Prints info and descriptive statistics to understand the data structure and summary.
4. **Remove Duplicates**
   * Ensures each employee (EmpID) is unique.
5. **Handle Missing Values**
   * Converts Salary to numeric, fills missing salaries with the median.
   * Fills missing job titles with 'Unknown'.
6. **Standardize Department and Job Titles**
   * Uses mapping dictionaries to ensure consistent naming.
7. **Add Years of Service**
   * Calculates years since joining for each employee.
8. **Department Aggregations**
   * Groups by department to compute:
     + Average salary
     + Total salary expenditure
     + Employee count
9. **Job Title Aggregations**
   * Groups by job title to compute:
     + Average salary
     + Highest paid employee for each job title
10. **Highest Paid Employee Per Department**
    * Finds the highest paid employee in each department.
11. **Export Summaries**
    * Saves the aggregated results to CSV files for reporting.
12. **Output Results**
    * If [tabulate](vscode-file://vscode-app/c:/Users/prath/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) is installed, prints beautiful tables.
    * Otherwise, prints DataFrames in plain text.
13. **Completion Message**
    * Indicates that data cleaning and aggregation are complete.

SCREENSHORT OF THE OUTPUT:



EXPLAINATION OF THE OUTPUT:

* **Dataset Info & Description:**  
  Shows the structure, data types, and summary statistics of your employee data.
* **Department Summary:**  
  Table with average salary, total salary, and employee count for each department.
* **Job Title Summary:**  
  Table with average salary and highest paid employee for each job title.
* **Highest Paid Employee Per Department:**  
  Table listing the top earner in each department.
* **Completion Message:**  
  Confirms that the process is finished and CSVs are exported.

Closure (Bibliography)

Python Software Foundation. Python Language Reference, version 3.x. Available at: [https://www.python.org/](vscode-file://vscode-app/c:/Users/prath/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

Wes McKinney. "pandas: powerful Python data analysis toolkit." Available at: [https://pandas.pydata.org/](vscode-file://vscode-app/c:/Users/prath/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

Python datetime module documentation. Available at: [https://docs.python.org/3/library/datetime.html](vscode-file://vscode-app/c:/Users/prath/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)