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:

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.

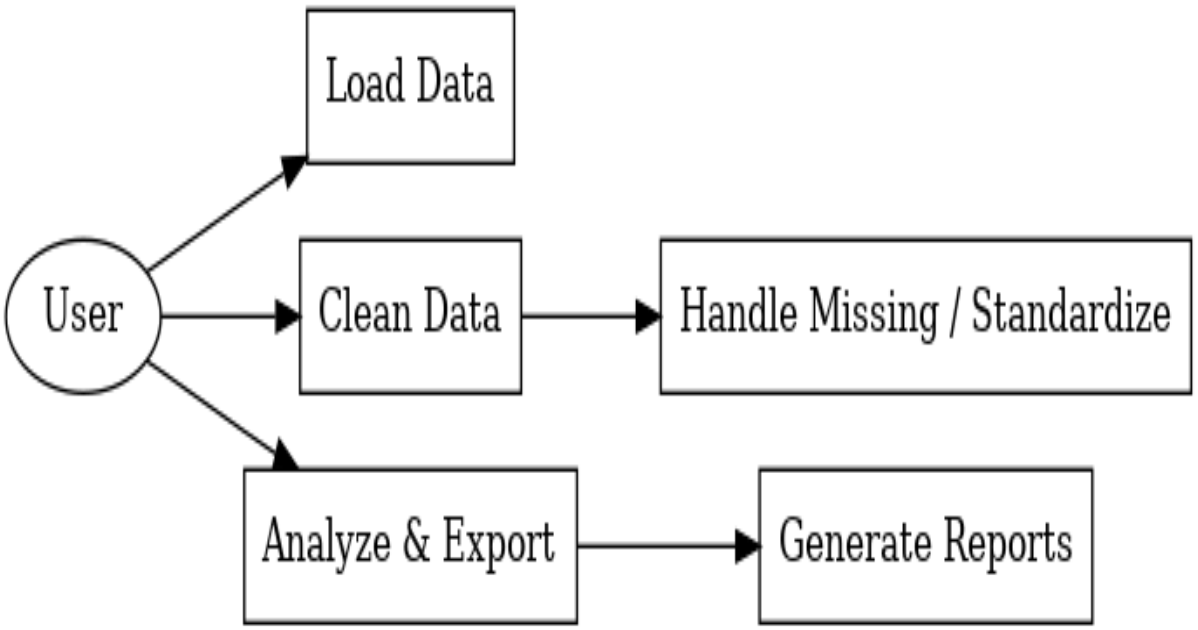
UML DIAGRAM:

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?*



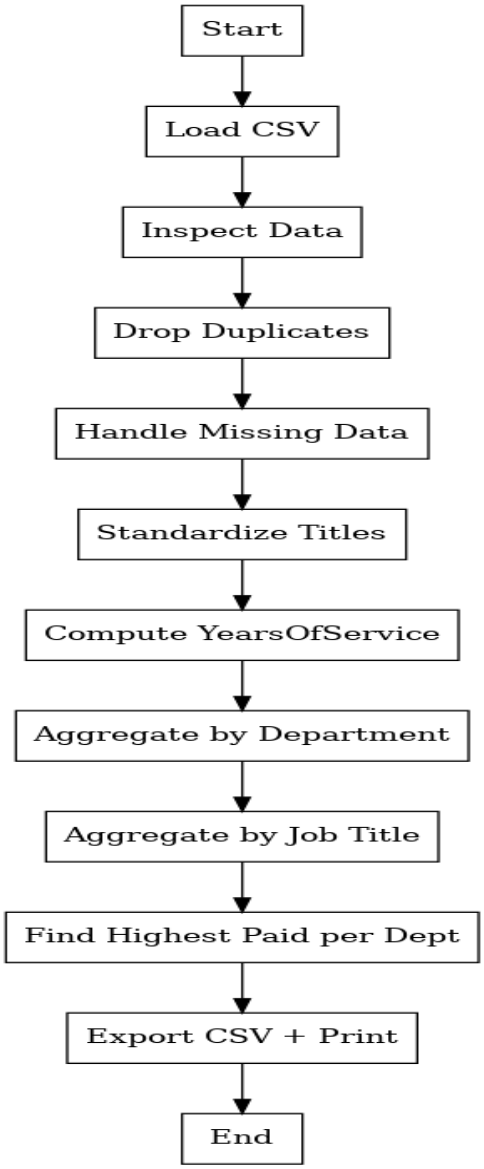
---

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?*



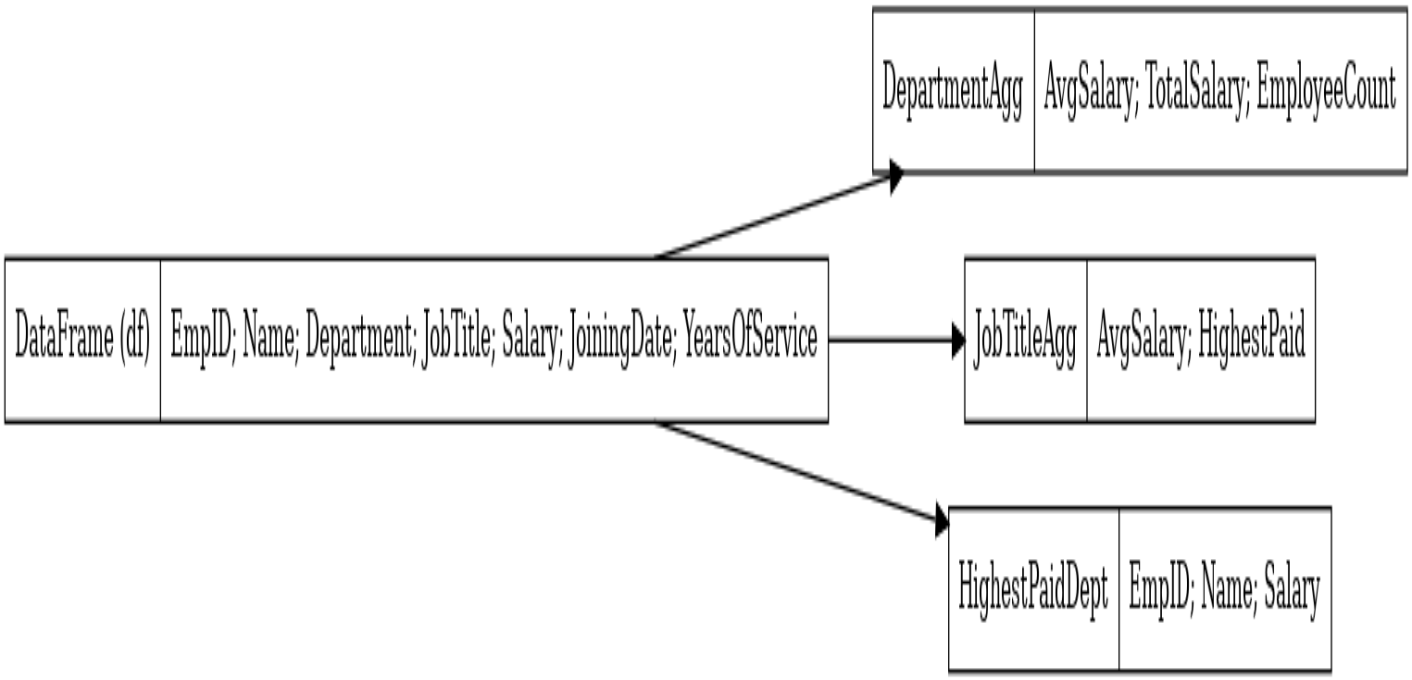
---

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?*



---

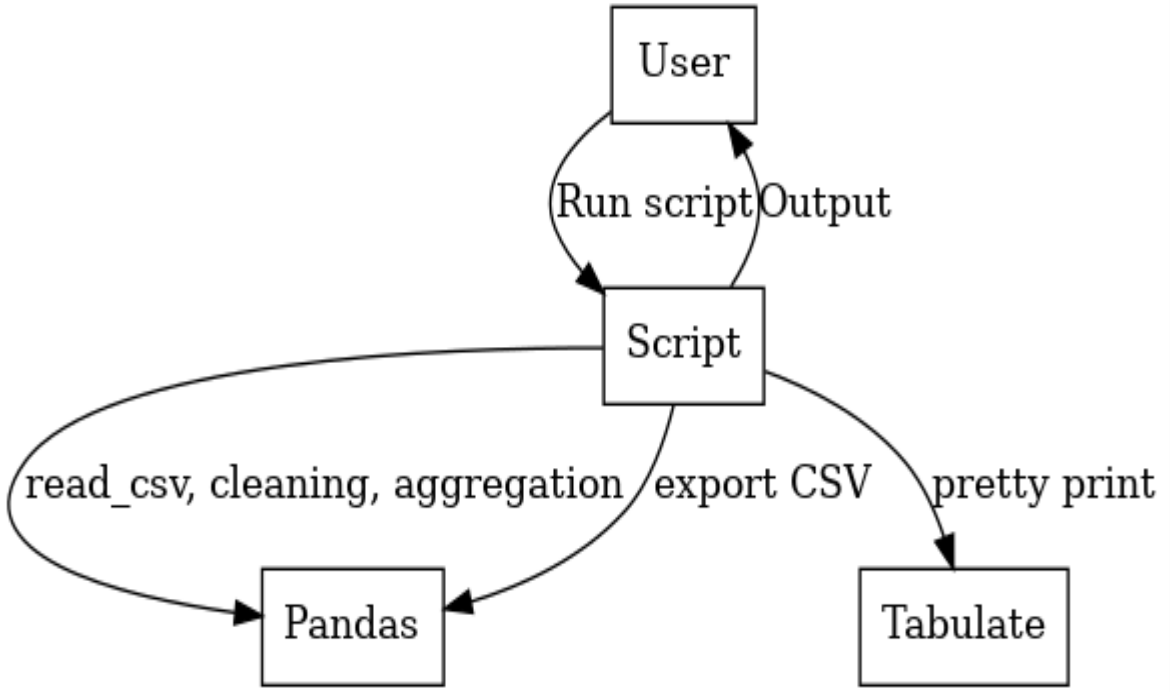
4. Sequence Diagram

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?*



---

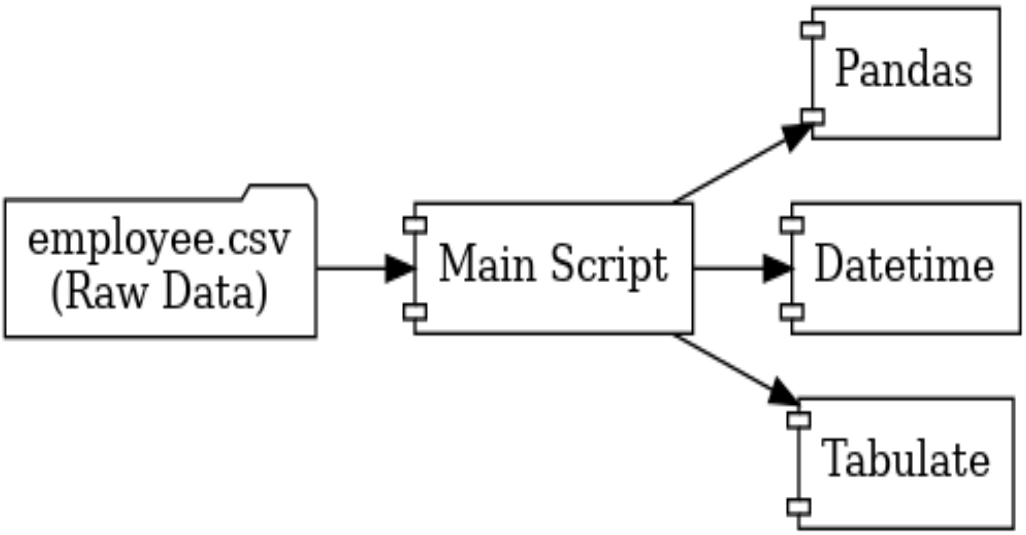
5. Component Diagram

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?*



✅ 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(
```

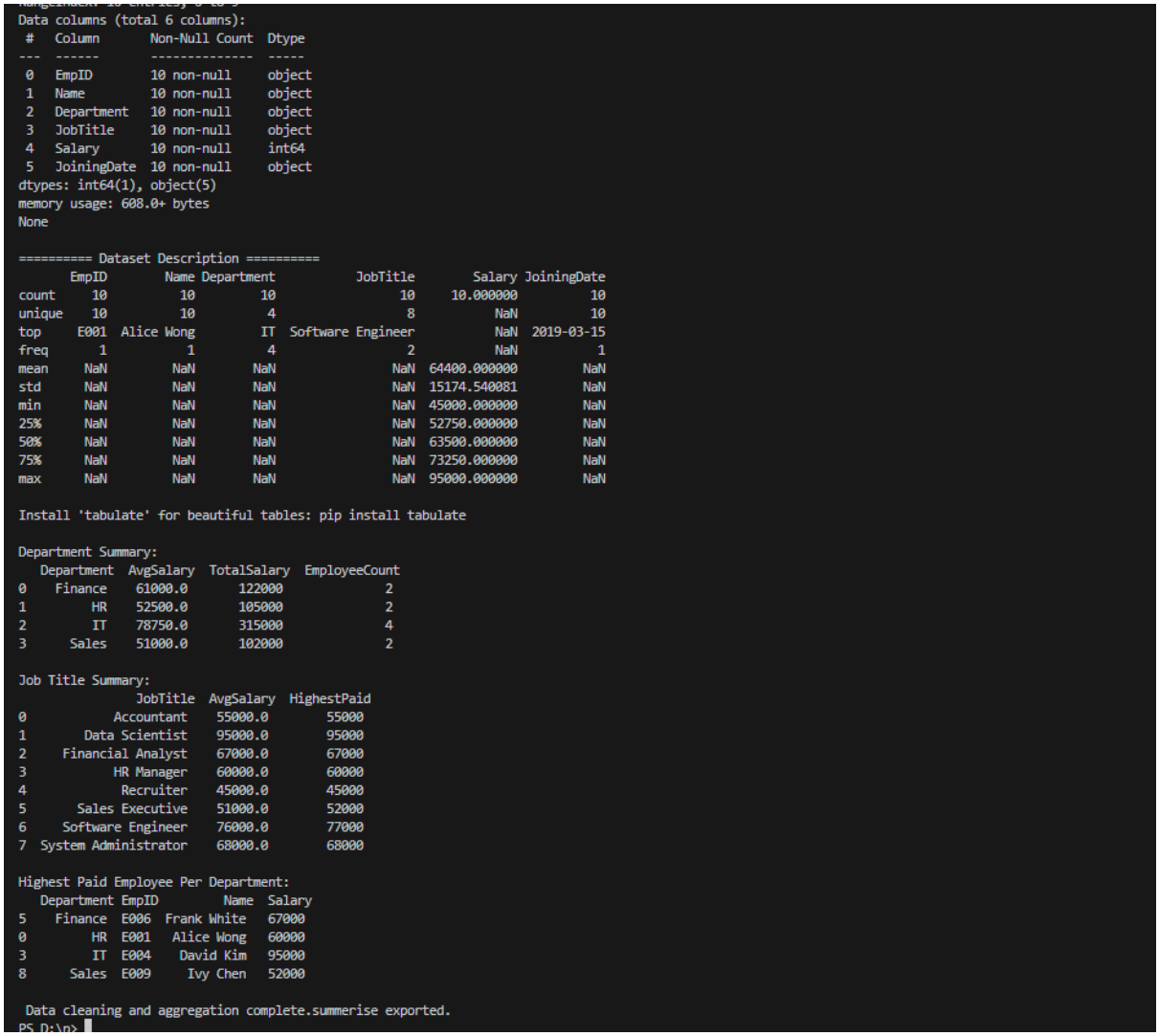
```
    AvgSalary=('Salary', 'mean'),
    HighestPaid=('Salary', 'max')
).reset_index()

# 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 for data manipulation.
    - datetime for date calculations.
2. **Load Dataset**
    - Reads employee.csv 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 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:



```
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   EmpID        10 non-null     object
 1   Name         10 non-null     object
 2   Department   10 non-null     object
 3   JobTitle     10 non-null     object
 4   Salary       10 non-null     int64
 5   JoiningDate  10 non-null     object
dtypes: int64(1), object(5)
memory usage: 608.0+ bytes
None

========== Dataset Description ==========
       EmpID        Name Department          JobTitle        Salary JoiningDate
count     10          10         10                10     10.000000          10
unique    10          10          4                 8           NaN          10
top     E001  Alice Wong         IT  Software Engineer           NaN  2019-03-15
freq       1           1          4                 2           NaN           1
mean     NaN         NaN        NaN               NaN  64400.000000         NaN
std      NaN         NaN        NaN               NaN  15174.540081         NaN
min      NaN         NaN        NaN               NaN  45000.000000         NaN
25%      NaN         NaN        NaN               NaN  52750.000000         NaN
50%      NaN         NaN        NaN               NaN  63500.000000         NaN
75%      NaN         NaN        NaN               NaN  73250.000000         NaN
max      NaN         NaN        NaN               NaN  95000.000000         NaN

Install 'tabulate' for beautiful tables: pip install tabulate

Department Summary:
   Department  AvgSalary  TotalSalary  EmployeeCount
0     Finance    61000.0       122000              2
1          HR    52500.0       105000              2
2          IT    78750.0       315000              4
3       Sales    51000.0       102000              2

Job Title Summary:
               JobTitle  AvgSalary  HighestPaid
0            Accountant    55000.0        55000
1        Data Scientist    95000.0        95000
2     Financial Analyst    67000.0        67000
3           HR Manager    60000.0        60000
4             Recruiter    45000.0        45000
5       Sales Executive    51000.0        52000
6     Software Engineer    76000.0        77000
7  System Administrator    68000.0        68000

Highest Paid Employee Per Department:
   Department EmpID        Name  Salary
5     Finance  E006  Frank White   67000
0          HR  E001   Alice Wong   60000
3          IT  E004    David Kim   95000
8       Sales  E009     Ivy Chen   52000

 Data cleaning and aggregation complete.summerise exported.
PS D:\p>
```

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/

Wes McKinney. "pandas: powerful Python data analysis toolkit." Available at: https://pandas.pydata.org/

Python datetime module documentation. Available at: https://docs.python.org/3/library/datetime.html