EMPLOYEE SALARY DATA

Problem Title:-

Employee Salary Data Cleaning and Aggregation for HR Insights using Pandas

NAME : LIKHITHA G S

EMAIL : likhitha0920@gmail.com

DATE : 2nd sep 2025

Description :

The HR department of a company maintains employee records containing details such as employee ID, name, department, job title, and salary. However, the dataset contains issues like duplicate entries, missing values, and inconsistent data formats, which make reporting inaccurate.
This project aims to clean the employee dataset using Python (Pandas), perform data aggregation to generate HR insights (e.g., average salary per department, highest/lowest salaries, headcount), and export the cleaned results for reporting purposes.
The project demonstrates data preprocessing, transformation, and visualization techniques useful in real HR analytics.

Index :

Problem Statement :

The HR department of a company maintains employee records, including salaries, departments, and job titles. However, the dataset contains duplicates, missing values, and inconsistent data formats. The goal is to clean the employee dataset using Pandas, perform aggregation to extract useful HR insights, and export the results for reporting.

Objectives:

1. Clean the dataset:

   Handle missing values in salaries and job titles.

   Remove duplicate employee records.

   Standardize department names and job titles.

2. Perform Aggregations:

   Calculate average salary per department.

   Identify highest-paid employee in each department.

   Find total salary expenditure per department.

   Group employees by job titles and compute their average salaries.

   Count employees per department.

3. Export cleaned and aggregated results into CSV files.

Dataset Description :

Columns :

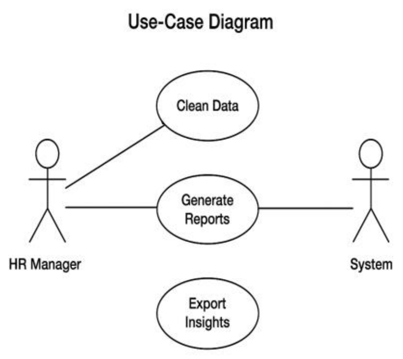
- Employee ID
- Name
- Department
- Job Title
- Salary
- Joining Date

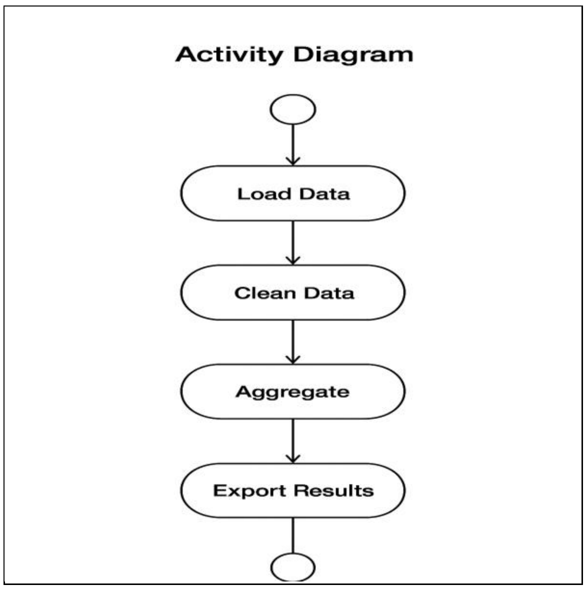
Use-Cases :

1. Data Cleaning → Handling duplicates, missing salaries, inconsistent department names.
2. Salary Insights → Average salary per department.
3. Headcount Analysis → Number of employees per department.
4. Salary Distribution → Highest/lowest salaries.
5. Exporting Data → Export to Excel/CSV for HR reporting.

UML Diagrams :

1. Use-Case Diagram
   - Actors: HR Manager, System
   - Use-Cases: Clean Data, Generate Reports, Export Insights
2. Activity Diagram
   - Steps: Load Data → Clean Data → Aggregate → Export Results
3. Data Flow Diagram (DFD)
   - Input: Raw Employee Data
   - Process: Cleaning & Aggregation
   - Output: HR Insights & Reports





Front-End / Interface Design :

- A simple dashboard (mock-up) with:
    - Dropdown to select department
    - Display average salary, headcount, etc.
    - Export button
    - Download a csv file

Code Implementation and Explanation:

```
import streamlit as st
import pandas as pd
import streamlit_authenticator as stauth
import plotly.express as px
```

- **Streamlit** (st) – Used to build the interactive web-based dashboard.
- **Pandas** (pd) – For reading, cleaning, and analyzing employee data stored in CSV format.
- **Streamlit Authenticator** (stauth) – To implement secure login and logout functionality.
- **Plotly Express** (px) – To create interactive and visually appealing charts (bar charts, metrics, etc).

```
names = ["Admin"]
usernames = ["likhi"]
passwords = ["123"]
credentials = {
    "usernames": {
        usernames[i]: {"name": names[i], "password": passwords[i]}
        for i in range(len(usernames))
    }
}
authenticator = stauth.Authenticate(credentials, "employee_dashboard", "abcdef", 1)
```

- Defines username, password, and display name for login.
- Builds a credentials dictionary required by the authenticator.
- Creates an authentication object (authenticator) with session handling.
- Ensures that only authorized users can access the dashboard.

```
col1, col2, col3 = st.columns([1, 2, 1])
with col2:
    authenticator.login(location="main")
```

- Divides the page into three columns.
- Places the login box in the middle column for a centered appearance.

```
if st.session_state.get("authentication_status"):
    name = st.session_state["name"]
    st.sidebar.write(f"👋 Welcome {name}")
    authenticator.logout("Logout", "sidebar")
    st.title("📊 HR Employee Salary Dashboard")
```

- Verifies if the user is logged in.
- Displays a personalized welcome message in the sidebar.
- Adds a Logout button.
- Shows the dashboard title.

```
uploaded_file = st.file_uploader("Upload Employee Data CSV", type=["csv"])
```

- Allows users to upload employee records in CSV format.
- If no file is uploaded, an information message is displayed.

```
df = pd.read_csv(uploaded_file)
df.columns = df.columns.str.strip().str.replace(" ", "").str.capitalize()

required_cols = ["Salary", "Department", "Jobtitle"]
for col in required_cols:
    if col not in df.columns:
        st.error(f"❌ CSV is missing required column: {col}")
        st.stop()

df.drop_duplicates(inplace=True)
df["Salary"] = pd.to_numeric(df["Salary"], errors="coerce")
df["Salary"] = df["Salary"].fillna(df["Salary"].mean())
```

- Loads the CSV file into a Pandas DataFrame.
- Standardizes column names (removes spaces and capitalizes).
- Ensures required fields (Salary, Department, Jobtitle) are present.

- Removes duplicate records.
- Converts Salary values to numeric and replaces invalid/missing entries with the average salary.

```python
if page == "Cleaned Data":
    st.subheader("🧹 Cleaned Employee Data")
    st.dataframe(filtered_df)
    if not filtered_df.empty:
        csv = filtered_df.to_csv(index=False).encode('utf-8')
        st.download_button("📥 Download Cleaned Data as CSV", data=csv,
                           file_name="cleaned_employee_data.csv", mime="text/csv")
```

- Displays the cleaned and filtered employee dataset.
- Provides a download option to export the cleaned data as a CSV file.

```python
elif page == "Charts & KPIs":
    st.subheader("📈 HR Key Performance Indicators (KPIs)")
    avg_salary = filtered_df["Salary"].mean() if not filtered_df.empty else 0
    st.metric("Average Salary", f"{avg_salary:,.2f}")
    total_emp = filtered_df.shape[0]
    st.metric("Total Employees", total_emp)
```

- Displays key performance indicators (KPIs):
    - Average Salary
    - Total Number of Employees

```python
dept_salary = filtered_df.groupby("Department")["Salary"].mean().reset_index()
fig_dept = px.bar(dept_salary, x="Department", y="Salary", color="Salary",
                  color_continuous_scale="Blues", text="Salary")
st.plotly_chart(fig_dept, use_container_width=True)
```

- Shows the average salary across departments.

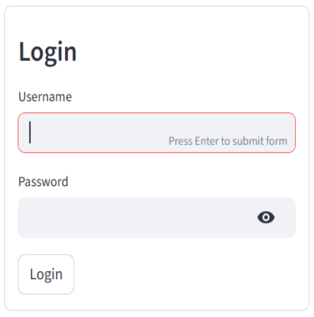
```python
top5 = filtered_df.nlargest(5, "Salary")[["Name", "Jobtitle", "Department", "Salary"]]
least5 = filtered_df.nsmallest(5, "Salary")[["Name", "Jobtitle", "Department", "Salary"]]
```

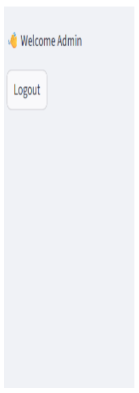
- Top 5 Earners: Displays employees with the highest salaries.
- Top 5 Least Earners: Displays employees with the lowest salaries.
- Provides useful insights into pay distribution.

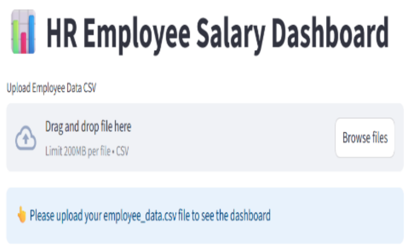
```
elif st.session_state.get("authentication_status") is False:
    st.error("❌ Username/password is incorrect")
```

- Displays an error message if invalid login credentials are entered.

Screenshots of Output :







Please upload your employee_data.csv file to see the dashboard

- This is the output of the code first we login then it will shows the dashboard after uploading a CSV file then the cleaned data will show and also charts .

Conclusion :

The HR Employee Salary Dashboard provides a comprehensive solution for salary analysis and workforce insights. The dashboard not only saves time but also improves accuracy in salary analysis, workforce planning, and compensation management.

Bibliography :

- Pandas :- https://youtu.be/vtgDGrUiUKk?si=XsAedzKFc1UIg4ot
- Streamlit :- https://docs.streamlit.io/
- Plotly :- https://plotly.com/python/