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 :

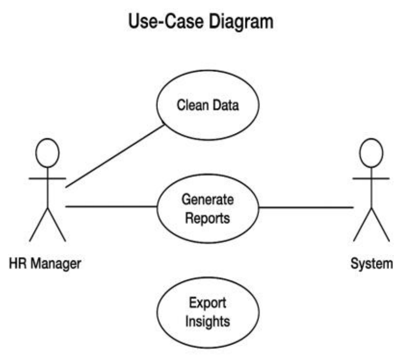
- Employee ID
- Name
- Department
- Job Title
- Salary
- Joining Date

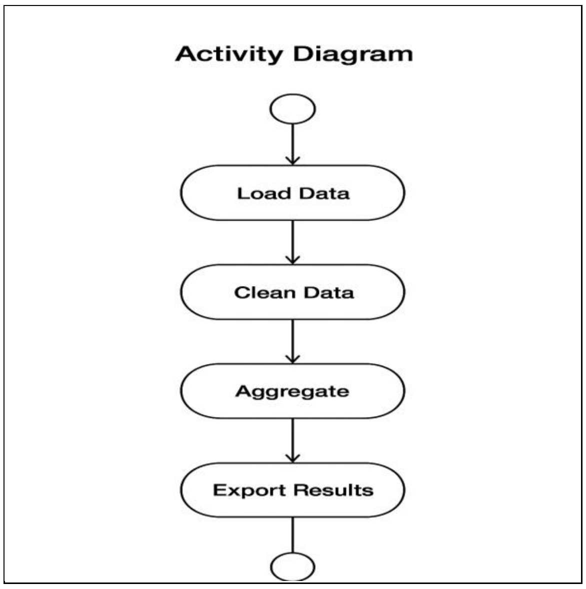
Use-Cases :

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

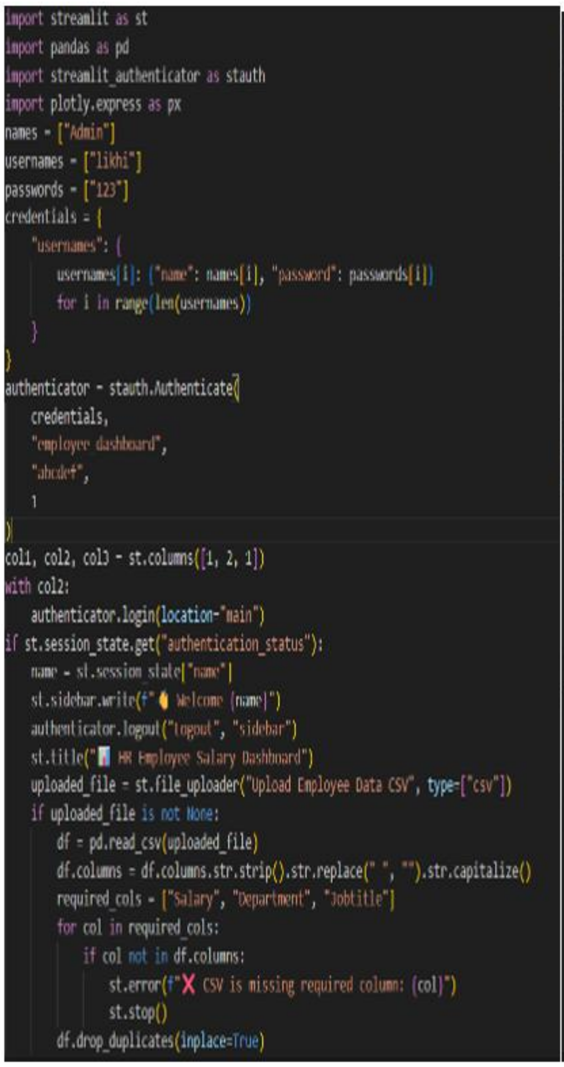




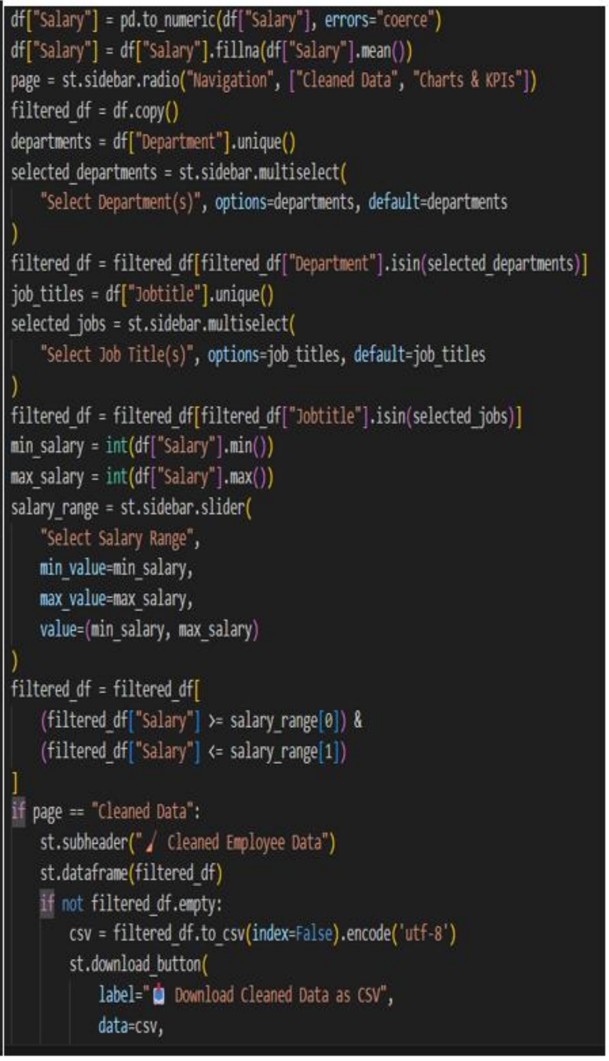
Front-End / Interface Design :

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

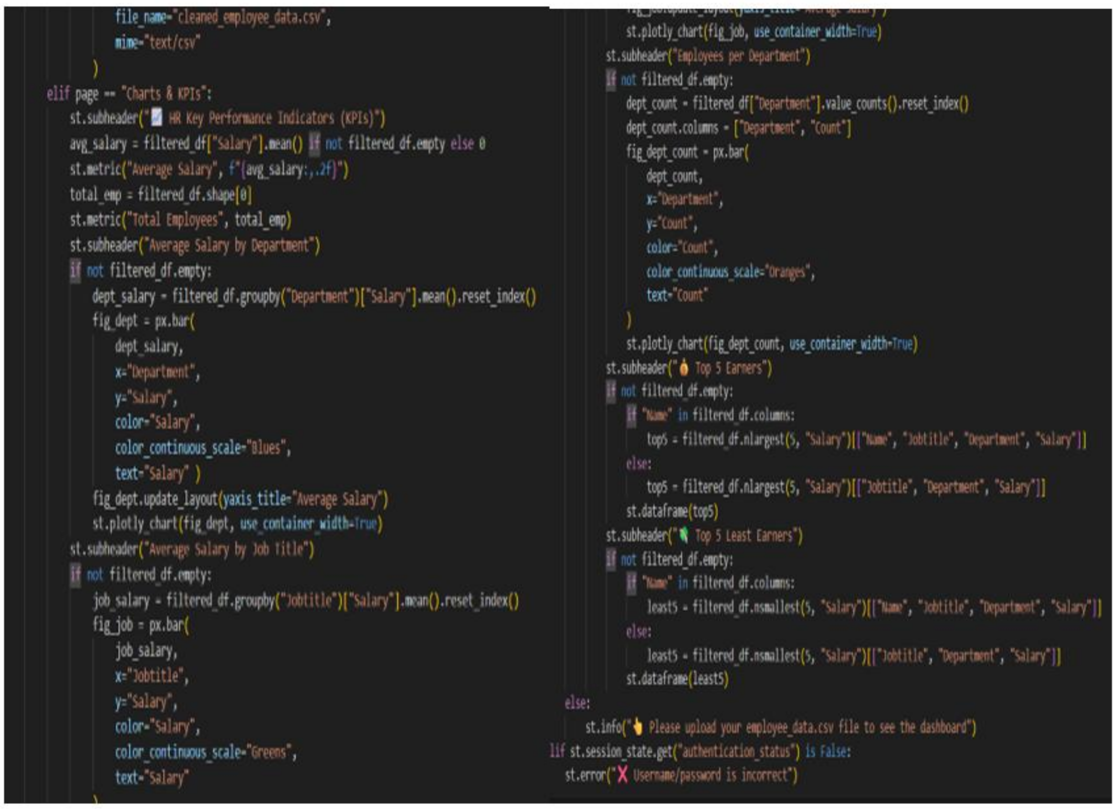
Code Implementation:



```python
import streamlit as st
import pandas as pd
import streamlit_authenticator as stauth
import plotly.express as px
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
)
col1, col2, col3 = st.columns([1, 2, 1])
with col2:
    authenticator.login(location="main")
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")
    uploaded_file = st.file_uploader("Upload Employee Data CSV", type=["csv"])
    if uploaded_file is not None:
        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)
```



```python
df["Salary"] = pd.to_numeric(df["Salary"], errors="coerce")
df["Salary"] = df["Salary"].fillna(df["Salary"].mean())
page = st.sidebar.radio("Navigation", ["Cleaned Data", "Charts & KPIs"])
filtered_df = df.copy()
departments = df["Department"].unique()
selected_departments = st.sidebar.multiselect(
    "Select Department(s)", options=departments, default=departments
)
filtered_df = filtered_df[filtered_df["Department"].isin(selected_departments)]
job_titles = df["Jobtitle"].unique()
selected_jobs = st.sidebar.multiselect(
    "Select Job Title(s)", options=job_titles, default=job_titles
)
filtered_df = filtered_df[filtered_df["Jobtitle"].isin(selected_jobs)]
min_salary = int(df["Salary"].min())
max_salary = int(df["Salary"].max())
salary_range = st.sidebar.slider(
    "Select Salary Range",
    min_value=min_salary,
    max_value=max_salary,
    value=(min_salary, max_salary)
)
filtered_df = filtered_df[
    (filtered_df["Salary"] >= salary_range[0]) &
    (filtered_df["Salary"] <= salary_range[1])
]
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(
            label="📥 Download Cleaned Data as CSV",
            data=csv,
```



```
                file_name="cleaned_employee_data.csv",
                mime="text/csv"
            )
    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)
        st.subheader("Average Salary by Department")
        if not filtered_df.empty:
            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" )
            fig_dept.update_layout(yaxis_title="Average Salary")
            st.plotly_chart(fig_dept, use_container_width=True)
        st.subheader("Average Salary by Job Title")
        if not filtered_df.empty:
            job_salary = filtered_df.groupby("Jobtitle")["Salary"].mean().reset_index()
            fig_job = px.bar(
                job_salary,
                x="Jobtitle",
                y="Salary",
                color="Salary",
                color_continuous_scale="Greens",
                text="Salary"
            )
```

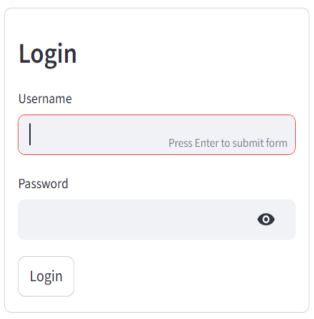
```
            st.plotly_chart(fig_job, use_container_width=True)
        st.subheader("Employees per Department")
        if not filtered_df.empty:
            dept_count = filtered_df["Department"].value_counts().reset_index()
            dept_count.columns = ["Department", "Count"]
            fig_dept_count = px.bar(
                dept_count,
                x="Department",
                y="Count",
                color="Count",
                color_continuous_scale="Oranges",
                text="Count"
            )
            st.plotly_chart(fig_dept_count, use_container_width=True)
        st.subheader("💰 Top 5 Earners")
        if not filtered_df.empty:
            if "Name" in filtered_df.columns:
                top5 = filtered_df.nlargest(5, "Salary")[["Name", "Jobtitle", "Department", "Salary"]]
            else:
                top5 = filtered_df.nlargest(5, "Salary")[["Jobtitle", "Department", "Salary"]]
            st.dataframe(top5)
        st.subheader("📉 Top 5 Least Earners")
        if not filtered_df.empty:
            if "Name" in filtered_df.columns:
                least5 = filtered_df.nsmallest(5, "Salary")[["Name", "Jobtitle", "Department", "Salary"]]
            else:
                least5 = filtered_df.nsmallest(5, "Salary")[["Jobtitle", "Department", "Salary"]]
            st.dataframe(least5)
    else:
        st.info("👆 Please upload your employee_data.csv file to see the dashboard")
lif st.session_state.get("authentication_status") is False:
    st.error("❌ Username/password is incorrect")
```

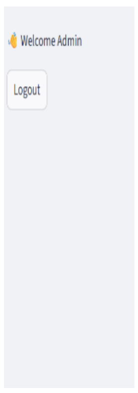
Explanation of Code :

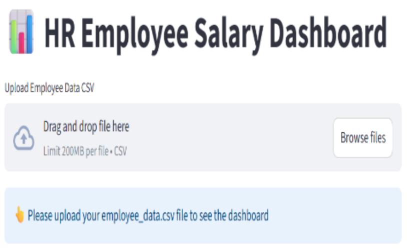
- The dashboard is designed to help HR departments analyze employee salary data, track workforce distribution, and gain meaningful insights.
- Built using Streamlit (frontend UI), Pandas (data cleaning), Plotly (visualizations), and Streamlit Authenticator (security).
- Load dataset → Reads employee data (employee_data.csv).
- Clean data
    - The system automatically Removes duplicates ,
    - Standardizes column names,
    - Converts salary values to numeric,
    - Replaces missing salaries with the average.
- Shows the cleaned and filtered dataset in a table view, with an option to download it as CSV.
- Aggregate data
    - Average Salary by Department,
    - Average Salary by Job Title,

    - Number of Employees per Department.
    - Summary statistics (mean, min, max, etc.)
    - KPIs: Average salary and total employees.
- Export results
    - Save cleaned dataset.
    - Export reports (summary, average salaries, counts, top and low salaries).
- By combining data cleaning, validation, interactive filtering, and visualization in a secure platform, it enables HR professionals to make data-driven decisions efficiently.

Screenshots of Output :






HR Employee Salary Dashboard

Please upload your employee_data.csv file to see the dashboard

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://pandas.pydata.org/docs/
- Streamlit :- https://docs.streamlit.io/
- Plotly :- https://plotly.com/python/