Setting env

VS code- python -m venv .venv

TO ACTIVATE- .venv\Scripts\Activate.ps1

C:\Users\admin\Desktop\Bhavyas Desktop\ETL project>.venv\Scripts\Activate.ps1

-------------------------------------------------------------------------------------------------

pip install streamlit pandas pyspark boto3 requests pyarrow – one time

iterpreter should be this- .../ETL project/.venv/Scripts/python.exe

Command for streamlit execution- python -m streamlit run dashboard/app.py

Or

Vs code terminal- streamlit run dashboard/app.py

ADZUNA API- <https://developer.adzuna.com/>

Bhavyavora-08, A\*\*\*\*\*#\*\*\*\*

PYTHON SCRIPT- ingestion\fetch\_jobs.py

import requests

import json

# Replace these with your Adzuna credentials

APP\_ID = "d2c08e2a"

APP\_KEY = "d5c4c6cff3e58d013813fa775152c49b"

url = "https://api.adzuna.com/v1/api/jobs/us/search/1"

params = {

    "app\_id": APP\_ID,

    "app\_key": APP\_KEY,

    "results\_per\_page": 10,  # keep small for testing

    "what": "data engineer",

}

response = requests.get(url, params=params)

data = response.json()

# Save locally in your project

with open("data/jobs\_raw.json", "w") as f:

    json.dump(data, f, indent=2)

print("✅ Raw job data saved to data/jobs\_raw.json")

application keys- <https://developer.adzuna.com/admin/access_details>

fetching **live job listings** using **Adzuna’s Jobs API**- Data Engineer jobs in the US

jobs\_raw.json

| **Field** | **Description** |
| --- | --- |
| title | Job title |
| company.display\_name | Company name |
| location.display\_name | City/State |
| salary\_min / salary\_max | Salary range |
| created | Job posting date |
| category.label | Job category |

The base URL we used in Python was:

<https://api.adzuna.com/v1/api/jobs/us/search/1>

* search/1 → first page of results

We added **query parameters in fetch\_jobs.py**:

| **Parameter** | **Purpose** |
| --- | --- |
| app\_id | Your Adzuna app ID |
| app\_key | Your Adzuna API key |
| results\_per\_page | Number of jobs to fetch (10 for prototype) |
| what | Search term, e.g., data engineer |

Now we we have fetched live data

# Next step- PySpark ETL (Transform Raw Data → Curated Data)

The goal here is to **clean, flatten, and transform your raw JSON** into a format ready for analysis or storage in S3.

After running spark\_etl.py

 Spark read the JSON ✅

 Flattened results ✅

 Selected the curated fields ✅

 Showed sample rows ✅

 Wrote **Parquet file** successfully to **data/jobs\_curated.parquet** ✅

Perfect 🚀 You’ve now completed the **ETL stage**:

* **E**xtract → JSON from Adzuna API ✅
* **T**ransform → Flattened & curated with PySpark ✅
* **L**oad → Written to Parquet ✅

Sure 👍 here’s the short recap of what you did **after creating the Parquet file**:

1. **Loaded Parquet into Pandas** → Read the curated Parquet file into a DataFrame.
2. **Cleaned & Validated Data** → Checked missing values, duplicates, salary consistency, and date ranges.
3. **Explored Data** → Got summary stats (salary distribution, top companies, categories).
4. **Saved Clean Data** → Wrote out a clean CSV (jobs\_curated\_clean.csv) for easy reuse

Script ran- analysis/explore.py and it generated csv.

Wrote script for streamlit dashboard- dashboard/app.py

Run command- **streamlit run c:/Users/admin/Desktop/BhavyasDesktop/ETLproject/dashboard/app.py**

**Summary-**

| **Step** | **Process / Action** | **Tools / Libraries** | **Output / Purpose** |
| --- | --- | --- | --- |
| 1 | **Data Fetching** | Adzuna API, Requests (Python) | Retrieve live job listings in **JSON format** |
| 2 | **Raw Data Storage** | Local File System | Save **JSON files** as raw input |
| 3 | **Data Ingestion** | PySpark | Load JSON into Spark DataFrame for processing |
| 4 | **ETL & Transformation** | PySpark | Flatten nested JSON, select relevant columns, handle missing values, clean & standardize data |
| 5 | **Optimized Storage** | PySpark | Save curated data as **Parquet** and CSV for fast querying and analytics |
| 6 | **Data Analysis** | Pandas, NumPy | Explore dataset, calculate stats, prepare aggregations |
| 7 | **Visualization** | Matplotlib, Plotly | Generate graphs: salary distribution, jobs by company/category, trends over time |
| 8 | **Dashboard Development** | Streamlit | Interactive dashboard with filters, dynamic plots, and data previews |
| 9 | **User Interaction** | Streamlit widgets | Users can filter by **company** & **category**, explore insights in real-time |
| 10 | **Deployment / Presentation** | Streamlit Local / Cloud | Showcase **real-time analytics dashboard** for data-driven insights |

**Job Analytics Dashboard using ETL & Streamlit**

### **Project Description:**

* Engineered a robust **data ingestion pipeline** using **PySpark**, efficiently **fetching live job listings** from the **Adzuna API (JSON)**, and transforming them into **optimized Parquet/CSV formats** for downstream analysis.
* Designed and developed an **interactive analytics dashboard** using **Streamlit**, enriched with **Pandas**, **NumPy**, **Matplotlib**, and **Plotly**, enabling dynamic exploration of **salary distributions, hiring trends, and company/category insights**.
* Implemented **intelligent filtering, interactive visualizations, and real-time data previews** to deliver actionable insights and facilitate **data-driven decision-making**.