✅ Project Title

"ESG Data Intelligence Platform: Scalable ETL & Analytics on Databricks with Delta Lake, Power BI & Tableau"

📌 Project Summary

This project implements an end-to-end ESG (Environmental, Social, and Governance) Data Intelligence Platform using Databricks, designed to support large-scale ingestion, transformation, and analysis of ESG metrics across companies and industries. The platform delivers curated ESG insights for business analysts, sustainability teams, and executive stakeholders through integrated dashboards built in both Power BI and Tableau.

The pipeline follows a robust Bronze–Silver–Gold Delta Lake architecture, and is fully orchestrated using Databricks Job Scheduler, ensuring regular updates and high data quality. It also includes data quality checks, automated test cases, and robust Python logging mechanisms for operational traceability.

This project mimics a real-world delivery model used by Big 4 consulting firms, combining Data Engineering best practices with Business Intelligence & Sustainability Analytics.

🎯 Project Objectives

Ingest & normalize ESG datasets from multiple sources (company CSVs, API, etc.)

Apply transformations and business rules to cleanse, standardize, and join ESG metrics

Implement a structured Delta Lake pipeline: Bronze → Silver → Gold

Schedule automated ETL pipelines using Databricks Jobs

Perform exploratory and advanced ESG analytics using PySpark and SQL

Deliver interactive, stakeholder-ready dashboards using Tableau & Power BI

Implement robust logging and test coverage for production-grade reliability

🏗️ Architecture Overview

+-----------------------+

| Raw ESG Data (CSV, |

| JSON, APIs) |

+-----------+-----------+

|

[ Bronze Layer ]

(Raw Ingestion)

|

[ Silver Layer ]

(Cleaned + Validated + Normalized ESG Metrics)

|

[ Gold Layer ]

(Curated ESG KPIs & Star Schema for Reporting)

|

+----------------+--------------------+

| |

[ Power BI Dashboards ] [ Tableau Dashboards ]

| |

[ ESG KPIs, Sector Trends, [ Carbon Emissions, ]

Sentiment Scores, Risk Board Diversity, etc. ]

Alerts, Compliance ]

🔧 Technology Stack

Component Tool/Service

Platform Databricks (Community / Pro Edition)

Data Processing PySpark

Storage Delta Lake (on DBFS or cloud mount)

Scheduling Databricks Jobs

Visualization Power BI, Tableau

Transformation Framework Bronze–Silver–Gold Architecture

Logging Python logging module

Testing pytest, Data Quality Check Suite

Format Support CSV, JSON, Delta

📊 Sample Data Sources

Source Type Format Description

Company Disclosures CSV ESG annual reports (emissions, diversity, governance)

Public ESG APIs JSON Simulated REST API data (sentiment, ratings)

ESG Ratings Dataset CSV Ratings from agencies (e.g., Refinitiv, Sustainalytics)

📁 Modules & Folder Structure

pgsql

Copy

Edit

esg\_data\_platform/

├── notebooks/

│ ├── 01\_data\_ingestion.ipynb

│ ├── 02\_data\_cleaning\_silver.ipynb

│ ├── 03\_transform\_gold\_layer.ipynb

│ ├── 04\_analysis\_kpis.ipynb

│ ├── 05\_dashboard\_output.ipynb

│ └── 06\_tests.ipynb

├── src/

│ ├── logging\_config.py

│ ├── utils.py

│ ├── data\_quality\_checks.py

│ └── kpi\_metrics.py

├── tests/

│ ├── test\_ingestion.py

│ ├── test\_transformations.py

│ └── test\_kpi\_metrics.py

├── data/

│ ├── esg\_companies.csv

│ ├── esg\_sentiment.json

├── sql/

│ └── kpi\_queries.sql

├── dashboard/

│ ├── ESG\_PowerBI.pbix

│ └── ESG\_Tableau.twbx

├── job\_configs/

│ └── databricks\_job\_config.json

├── README.md

└── requirements.txt

📈 Analytics Output & KPI Examples

KPI Name Description

ESG Score (Normalized) Weighted combination of E, S, G metrics

Carbon Emission Intensity CO₂ emissions per revenue unit

Gender Diversity Ratio % of female employees/board members

Board Independence % Ratio of independent directors

ESG Risk Flag Based on thresholds from Silver layer

Year-over-Year ESG Delta Trend in ESG performance

🧪 Testing & Logging

Testing: Includes PyTest files to validate:

Schema integrity

Null value thresholds

Business rule accuracy (e.g., ESG score ≥ 0 and ≤ 100)

Logging:

Centralized logger with file + console handler

Captures job status, failures, data quality errors

Log rotation support to avoid size bloat

⏰ Scheduled Jobs (Databricks)

Modular pipeline notebooks are scheduled as Databricks Jobs:

Ingestion → Silver Transform → Gold Transform → KPI Analysis → Export

Jobs are parameterized for reusability and automation

Alerting (via email) can be added for job failure events

🏁 Final Outputs

Two executive dashboards:

📊 Power BI for ESG trend visualizations

📈 Tableau for interactive stakeholder engagement

Delta Lake tables:

/bronze/esg\_raw/

/silver/esg\_clean/

/gold/esg\_kpis/

Job workflows: JSON-configured and schedulable

Test coverage reports and log files for observability

📜 Fully documented with architectural diagrams and user guide

💼 Resume/LinkedIn-Ready Summary

Designed and implemented an enterprise-scale ESG Data Intelligence Platform using Databricks, Delta Lake, and Power BI/Tableau to analyze sustainability metrics for 1000+ companies. Built a robust ETL pipeline with Bronze–Silver–Gold architecture, automated via Databricks Jobs, and enriched with data quality tests and operational logging. Delivered actionable ESG insights across carbon emissions, diversity, and governance risks to simulate real-world advisory projects at Big 4 consulting scale.

https://www.linkedin.com/pulse/azure-data-engineering-project-implementing-medalion-from-%C3%A7elik-upyae/

# Main Project Tech Stack

1. We’re building an ESG Data Intelligence Platform using Databricks Free Edition to process and analyze sustainability data for 1000+ companies.

2. A robust PySpark ETL pipeline with Bronze–Silver–Gold Delta architecture will clean, transform, and store data in DBFS.

3. Final CSV outputs will be manually uploaded to S3 and loaded into Amazon Redshift using Python for centralized access.

4. Post-loading, we’ll perform business-driven data analysis in Jupyter Notebooks to extract strategic ESG insights.

5. These insights will be visualized through interactive Power BI and Tableau dashboards, simulating Big 4–grade advisory solutions. The pipeline includes automated job scheduling, data quality testing, and logging, simulating real-world Big 4 delivery.

✅ Tech Stack (Databricks, PySpark, Delta, Redshift, Power BI/Tableau)  
✅ Architecture (Bronze–Silver–Gold)  
✅ Workflow (ETL → S3 → Redshift → Analysis → Dashboards)  
✅ Real-world practices (scheduling, testing, logging)  
✅ Business value (strategic ESG insights, Big 4 simulation)

You can confidently use this in your resume, LinkedIn, or project report — it's spot-on. ✅🔥

# ESG Project Details

Thanks for sharing the PDF — it’s rich in detail and gives a crystal-clear view of the **data structure**, **business relevance**, and **technical scope** of the Climate TRACE dataset. Based on this, here's a professional breakdown:

**✅ Business Problem We Will Solve**

**How can carbon-intensive electricity generation facilities be identified, analyzed, and benchmarked globally to support ESG investment decisions and emission-reduction strategies?**

This project will simulate the role of a Big 4 ESG analyst or consultant working with a **government agency**, **investment firm**, or **corporate ESG strategy team** to:

* Detect **high-emission power plants**
* Assess **regional vs. global benchmarks**
* Evaluate **emission trends by source type, gas type, and geography**
* Prioritize **target companies** or regions for **ESG improvement or risk mitigation**

**🎯 Specific Analytical Goals**

* Rank power plants by **CO₂-equivalent emissions**
* Compare **GHG vs. non-GHG** pollutants across countries
* Segment facilities by **source type** (e.g., coal, gas, oil, wind, etc.)
* Map emission **hotspots** by region
* Correlate **capacity factors vs. emission intensity**
* Highlight plants with **low reporting confidence** for audit risk

**📊 ESG KPI Examples We Can Derive**

|  |  |
| --- | --- |
| **KPI** | **Description** |
| Total CO₂e Emissions | CO₂-equivalent (100 yr GWP) across power plants |
| Emission Intensity | Emissions per MW of capacity |
| High-Risk Plants | Plants with high emissions and low data confidence |
| Capacity Utilization Risk | Plants operating above rated capacity |
| Emission Mix Profile | % split across CO₂, CH₄, N₂O, and pollutants |
| Governance Red Flag | Facilities with unknown or poor ownership traceability |

**💼 Resume-Ready Project Pitch**

**Designed and deployed a large-scale ESG Data Intelligence Platform to analyze carbon-intensive electricity generation sources using the Climate TRACE dataset. Built a modular PySpark ETL pipeline in Databricks with Delta Lake (Bronze–Silver–Gold architecture), extracting GHG and non-GHG emissions across 100,000+ facility records globally. Performed analytical modeling to identify high-risk emission zones, benchmark performance across geographies, and assess governance traceability. Outputs were exported to Redshift and visualized in Tableau and Power BI to simulate advisory recommendations for ESG investing and regulatory compliance—aligned with Big 4 ESG consulting practices.**

If you'd like, we can now:

* Choose the **first file to ingest**
* Start building the **Bronze layer notebook** in Databricks  
  Let me know!

# **Final Project Architecture:**

Excellent, Pranoy — here’s a full summary of your **ESG Emissions Intelligence Platform project** workflow, end-to-end, in bullet points:

**🔧 ETL & Data Engineering (Databricks Free Edition)**

* ✅ Ingest and process the **Climate TRACE Electricity Generation dataset (v4.4.0)** using **PySpark**
* ✅ Implement the **Medallion Architecture**:
  + **Bronze Layer**: Raw facility-level emissions data from CSVs
  + **Silver Layer**: Cleaned, validated data (emission types, units, source classification)
  + **Gold Layer**: ESG KPIs (emission intensity, CO₂e totals, confidence flags, governance risk)
* ✅ Store all layers as **Delta tables in DBFS**
* ✅ Automate the pipeline using **Databricks Jobs Scheduler**
* ✅ Add **data quality tests** and **Python logging** for operational observability

**🧾 Data Export & Integration**

* ✅ Export **Gold Layer** output as **CSV files from DBFS**
* ✅ **Manually upload** the CSVs to an **AWS S3 bucket**
* ✅ Trigger an **AWS Lambda function** (written in Python) to:
  + Connect to **Amazon Redshift**
  + Run the COPY command to load data from S3 into a Redshift table
* ✅ Redshift is used as the **central analytics warehouse**

**📊 Data Analysis & Visualization**

* ✅ Connect to **Amazon Redshift** from your **local Jupyter Notebook**
  + Perform **business analysis** on ESG metrics:
    - Emission hotspots
    - Governance gaps
    - Benchmarking emission intensity
    - ESG compliance risks
* ✅ Connect **Power BI and Tableau** to Redshift
  + Build **interactive ESG dashboards** with filtering, trends, KPIs

**💼 Business Problem Solved**

Identify and benchmark carbon-intensive power plants globally to support ESG investing, regulatory compliance, and sustainability advisory reporting — simulating Big 4–level client delivery.

**✅ Technology Stack Summary**

|  |  |
| --- | --- |
| **Layer** | **Tools** |
| Data Processing | Databricks (Free), PySpark |
| Storage | Delta Lake (DBFS), CSV |
| Orchestration | Databricks Jobs |
| Data Export | AWS S3 |
| Data Load | Amazon Redshift via AWS Lambda |
| Analytics | Jupyter Notebook, Power BI, Tableau |
| Logging & Testing | Python logging, pytest |

Let me know when you're ready to begin setting up the DBFS workspace — we’ll start building this step by step! 💼📊🔥

# Medallion Architechture:

==========================================================================================================================

A bronze, silver, and gold layer data engineer project using Azure implements a medallion architecture for data processing, with each layer representing a stage of data refinement. Raw data resides in the bronze layer, cleaned and transformed data in the silver layer, and curated, aggregated data in the gold layer. Azure services like Data Factory, Databricks, and Synapse Analytics are commonly used to orchestrate and execute these transformations. [1, 2, 3, 4]

Here's a breakdown of the project structure and Azure services involved:

1. Bronze Layer (Raw Data):

• Purpose: Stores data in its original format as ingested from source systems (e.g., CSV, JSON, databases).

• Azure Services:

• Azure Data Lake Storage Gen2: Used for storing the raw data in its native format.

• Azure Data Factory (ADF): Used for orchestrating the initial data ingestion from various sources into the bronze layer.

• Example: ADF can be used to extract data from an Azure SQL database or an on-premises system and load it into Azure Data Lake Storage in CSV or JSON format. [1, 1, 4, 4, 5, 6]

2. Silver Layer (Cleaned and Standardized Data):

• Purpose: Cleans, transforms, and standardizes the raw data, applying data quality checks and preparing it for further analysis. [1, 2]

• Azure Services:

• Azure Databricks: Utilized for data transformation using PySpark and other data processing frameworks. [1, 4]

• Delta Lake: Often used in conjunction with Databricks for efficient and reliable storage of transformed data. [7]

• Example: Databricks can be used to remove duplicates, handle missing values, apply schema validation, and perform data type conversions. [1, 4]

3. Gold Layer (Curated and Aggregated Data):

• Purpose: Aggregates and structures the data for specific analytical and reporting purposes, often using dimensional modeling and creating fact and dimension tables. [1, 1, 7, 7]

• Azure Services:

• Azure Synapse Analytics: Can act as a data warehouse for storing the aggregated data and providing a SQL endpoint for querying. [1, 1, 4, 4]

• Power BI: Used for data visualization and creating dashboards on top of the curated data. [1, 1, 4, 4, 8, 9]

• Example: Aggregated sales data might be stored in the gold layer for reporting and analysis in Power BI, using a star schema or other dimensional models. [1, 1, 4, 4]

Orchestration and Automation:

• Azure Data Factory: Orchestrates the entire data pipeline, scheduling and monitoring the execution of different stages across the bronze, silver, and gold layers.

• Example: ADF can trigger Databricks jobs to process data in the silver layer and then load the results into Synapse Analytics for the gold layer. [1, 4]

Key Benefits of the Medallion Architecture:

• Improved Data Quality: By processing data in stages, the architecture ensures that data is refined and cleansed at each step, leading to higher quality data in the gold layer.

• Flexibility and Scalability: The modular nature of the architecture allows for easy modification and scaling of individual layers as needed.

• Enhanced Data Accessibility: The gold layer provides a curated and structured view of the data, making it easier for business users to access and analyze the data.

• Cost-Effectiveness: By processing data incrementally, the architecture can optimize resource utilization and potentially reduce costs. [1, 1, 2, 2, 10, 11]

AI responses may include mistakes.

[1] https://medium.com/@kaushal\_akoliya/end-to-end-etl-using-azure-with-medallion-architecture-part-1-7671e66b10dc[2] https://medium.com/@gunjansinghtandon/medallion-architecture-in-azure-a-practical-implementation-5069b4963ba5[3] https://learn.microsoft.com/en-us/azure/architecture/solution-ideas/articles/azure-databricks-modern-analytics-architecture[4] https://medium.com/@vivekwarkade000/building-a-modern-data-pipeline-with-azure-data-engineering-project-69fcda317f87[5] https://learn.microsoft.com/en-us/azure/architecture/databases/architecture/azure-data-factory-on-azure-landing-zones-baseline[6] https://rafaelrampineli.medium.com/transforming-etl-from-sql-server-to-azure-data-engineering-e5d7982d3a2e[7] https://medium.com/@kanaksingh2785/azure-end-to-end-data-engineering-project-259b0e0cc38d[8] https://synthelize.com/post/data-lake-solutions/[9] https://medium.com/integration-of-sap-ecc-sap-datsphere-and-azure/integration-of-sap-ecc-sap-datasphere-and-azure-analytics-be135a1f64cc[10] https://intellifysolutions.com/blog/medallion-architecture-guide/[11] https://support.fabric.microsoft.com/en-us/blog/optimizing-spark-compute-for-medallion-architectures-in-microsoft-fabric?ft=All

Not all images can be exported from Search.

# AWS Small Project:

This guide will walk you through building a simple AWS project that automatically inserts data from a CSV file uploaded to S3 into an Amazon Redshift PostgreSQL table using an AWS Lambda function. This project is designed to stay within the AWS Free Tier as much as possible, but it's crucial to monitor your usage to avoid unexpected charges.

**Estimated Time to Complete:** 1-2 hours (excluding data upload time)

**AWS Free Tier Considerations:**

* **AWS Account:** New AWS customers are eligible for the Free Tier for 12 months.
* **Amazon S3:** 5 GB of Standard Storage, 20,000 Get Requests, 2,000 Put/Copy/List requests per month. This should be sufficient for small CSV files.
* **AWS Lambda:** 1 million free requests per month and 400,000 GB-seconds of compute time per month. This is very generous for our use case.
* **Amazon Redshift Serverless:** 1 GB of data stored and 1 TB of query data scanned each month at no cost. This is the best option for free tier Redshift. **Be careful to use Redshift Serverless and enable auto-pause to avoid charges.**

**Project Architecture:**

1. **Amazon S3 Bucket:** Stores your CSV files.
2. **AWS Lambda Function:** Triggered when a new CSV file is uploaded to the S3 bucket. It reads the CSV, connects to Redshift, and inserts the data.
3. **Amazon Redshift Serverless:** A data warehouse where your CSV data will be stored in a PostgreSQL-compatible table.
4. **AWS IAM:** Manages permissions for Lambda to access S3 and Redshift.

## AWS Account Setup:

**Step 1: Set up your AWS Account (if you don't have one)**

1. **Go to the AWS Free Tier page:** <https://aws.amazon.com/free/>
2. **Click "Create a Free Account."**
3. **Follow the on-screen instructions:**
   * You'll need a valid email address, a credit/debit card (for identity verification, you won't be charged for Free Tier usage), and a phone number.
   * Choose a strong password for your root user.
   * Select the "Basic Support (Free)" plan.
4. **Verify your identity.**
5. **Sign in to the AWS Management Console** as the root user after your account is created.

https://checkip.amazonaws.com/