Data Pipeline Learning Project

1. Project Overview

Purpose

This project guides students through building an end-to-end data pipeline using **Apache Airflow**, **dbt**, **ReportLab**, anadd d **OpenPyXL**. Students will ingest raw feature tables and variable metadata, transform and model data, and generate two distinct outputs:

1. **PDF Report**: A concise, visual executive summary highlighting key metrics and insights.
2. **Excel Workbook**: A detailed, self-service file with raw data and a complete data dictionary.

Learning Objectives

- Understand pipeline orchestration in Airflow
- Implement data transformations and modeling with dbt
- Automate report generation in Python (PDF + Excel)
- Practice best practices: modular code, documentation, and version control

Team Size: Designed for two interns collaborating effectively:

- **Intern 1:** Data ingestion & Airflow orchestration (file sensors, staging pipelines, PostgreSQL uploads).
- **Intern 2:** dbt modeling & report generation (Excel workbook and PDF report specifications).

This division provides clear ownership, easy handoffs, and collaborative problem-solving.

---

2. Architecture & Workflow

2.1 Data Sources

- **Directory Layout**:

- `bank.zip` (contains `bank-full.csv` and related feature files)

- `bank-additional.zip` (contains `bank-additional.csv` and metadata files)

- **Dataset Download**: Pull the UCI Bank Marketing data from: https://archive.ics.uci.edu/dataset/222/bank+marketing

- After downloading, extract to yield the two ZIPs above.

- **Main Table (``)**: Core features and target (`y`) ingested from `bank.zip`.

- **Additional Table (``)**: Variable definitions and allowed values ingested from `bank-additional.zip`.
- **Upload to PostgreSQL**: In the ingestion stage, load both CSVs into Postgres staging schema:
- Create tables `staging.main` and `staging.metadata`.
- Use `COPY` or client library (psycopg2) to bulk-load from local CSV files.

2.2 Pipeline Stages

```
graph LR
  A[Detect New Files] --> B[Stage Raw Tables]
  B --> C[dbt: Staging Models]
  C --> D[dbt: Core Models]
  D --> E[dbt: Reporting Models]
  E --> F[Generate Excel]
  E --> G[Generate PDF]
```

1. **Ingestion & Staging** (Airflow)
2. Sensor for file arrival
3. PythonOperators to load CSVs into a staging schema
4. **Transforms & Models** (dbt)
5. **stg_main**: Clean and cast raw feature data
6. **stg_metadata**: Normalize variable metadata
7. **dim_variables**: Combine metadata for documentation
8. **fact_features**: Core fact table for analysis
9. **report_summary**: Aggregated metrics (e.g., subscription rates by segment)
10. **report_full_dump**: Wide table for detailed data
11. **Output Generation** (Airflow + Python)
12. **Excel**: Export `report_full_dump` and `dim_variables` via OpenPyXL
13. **PDF**: Render charts and narrative from `report_summary` via ReportLab

---

3. Detailed Workflow Steps

A. Airflow DAG Configuration

- **Filename**: `pipeline_dag.py`
- **Schedule**: Every 5 minutes.
- **Tasks**:
- `wait_for_files`
- `stage_main_table`
- `stage_metadata_table`
- `dbt_run`
- `make_excel`
- `make_pdf`
- **Dependencies**:

```
[stage_main_table, stage_metadata_table] >> dbt_run >> [make_excel,
make_pdf]
```

B. dbt Model Structure

```
models/
├── staging/
│   ├── stg_main.sql
│   └── stg_metadata.sql
├── marts/
│   ├── core/
│   │   ├── dim_variables.sql
│   │   └── fact_features.sql
│   └── reporting/
│       ├── report_summary.sql
│       └── report_full_dump.sql
└── schema.yml  # Model tests and descriptions
```

C. Report Generation Specifications

Below are the detailed requirements for each output—without sample code.

Excel Workbook Requirements

- **Workbook Structure:**

- **Sheet 1: Data**

  - Complete rows from the `report_full_dump` model.

- **Sheet 2: Data Dictionary**
    - Columns: variable name, data type, description, allowed values (sourced from `dim_variables`).
- **Sheet 3: Pivot Tables (optional but recommended)**
    - E.g. subscription rate by job, summary statistics by education level.
- **Formatting Guidelines:**
- Freeze top row and first column in each sheet for easy navigation.
- Auto-adjust column widths based on content length.
- Apply bold headers with light shading for clarity.
- Use appropriate number formats (percent for rates, integer for counts).
- **Delivery:**
- Save as `report.xlsx` to a designated output directory.
- Log the file path and success message in airflow.
- (Optional) Upload or distribute via email or shared storage.

PDF Report Requirements

- **Overall Layout:**
- **Title Page** with report name and generation date.
- **Key Metrics Section** displaying total contacts and overall subscription rate.
- **Visualization Section** featuring at least two charts:
    1. Bar chart of subscription rate by job category.
    2. Line chart of monthly contact success trend.
- **Insights & Recommendations** as bullet points highlighting top findings (the result of Exploratory Data Analysis and actionable recommendation).
- **Styling Guidelines:**
- Use a clean, professional font and consistent heading hierarchy.
- Ensure charts are high-resolution and appropriately labeled (titles, axes).
- Maintain one-inch margins and clear spacing between sections.
- **Delivery:**

- Save as `summary.pdf` to the output directory.

- Log generation details in airflow.
- (Optional) Attach or publish to a reporting portal.

Objective Modeling:

Predict whether a client will subscribe to the term deposit (`y=yes` vs. `y=no`).

for example if using logic "if-then" rules over your input features (decision-tree style):

1. **Root split**
2. The algorithm evaluates all features (e.g. call duration, account balance, job type, housing loan) and picks the one that best separates subscribers from non-subscribers.
3. **Branching**
4. For each branch (Yes/No), it re-evaluates remaining features to find the next optimal split.
5. **Leaf nodes**
6. Recursion stops when the node is “pure” (all examples agree) or you hit a maximum depth/minimum sample threshold.
7. **Human-readable rules**
8. Every path from root to leaf becomes a rule, for example: > **IF** `duration > 300s` **AND** `balance > €1 000` **AND** `job = management` **THEN** predict **yes**

Note : **explore 4 model and make it model comparison chart visualization.**

---

4. Deliverables for Students Deliverables for Students

1. **Airflow DAG code** (`pipeline_dag.py`)
2. **dbt project** files (`models/`, `dbt_project.yml`, `profiles.yml`)
3. **Python scripts** for Excel and PDF generation
4. **README** with setup instructions, run commands, and expected outputs
5. **Sample outputs**: `summary.pdf` and `report.xlsx`

---

5. Assessment & Tips

- Write clear docstrings and comments
- Test dbt models with sample data
- Handle edge cases (empty tables, missing values)
- Keep configuration (paths, credentials) in environment variables

Good luck building your first end-to-end data pipeline!