**Metaflow ETL Workflow for Airbnb Data**

**Overview**

This project implements an Extract, Transform, Load (ETL) workflow using Metaflow, Python, and PostgreSQL to process Airbnb data. The workflow consists of separate scripts for data ingestion (data\_ingestion.py) and data transformation (data\_transformation.py), orchestrated by Metaflow (etl\_flow.py). This documentation provides an overview of each component

**Components**

**1. db\_connection.py**

This module manages database connection and credentials using psycopg2 for PostgreSQL interaction and python-dotenv for environment variable management.

* **Functions**:
  + connect\_to\_db(): Establishes a connection to the PostgreSQL database using parameters loaded from .env.

**2. data\_ingestion.py**

Responsible for creating the database table and loading data from a CSV file (AB\_NYC\_2019.csv) into PostgreSQL.

* **Steps**:
  1. **Table Creation**:
     + Creates airbnb\_listings table with columns matching Airbnb dataset schema.
  2. **Data Loading**:
     + Reads data from AB\_NYC\_2019.csv.
     + Cleans data by replacing empty strings with None.
     + Inserts data into airbnb\_listings table using psycopg2.

**3. data\_transformation.py**

Handles data transformation by normalizing last\_review into date and time columns and calculating average price per neighbourhood.

* **Steps**:
  1. **Normalization**:
     + Creates airbnb\_listings\_normalized table with normalized columns.
     + Normalizes last\_review into last\_review\_date (DATE) and last\_review\_time (TIME).
  2. **Average Price Calculation**:
     + Creates neighbourhood\_avg\_price table to store average price per neighbourhood.
     + Calculates average price (avg\_price) for each neighbourhood using ROUND() and AVG() functions.
     + Updates existing entries with new average prices using ON CONFLICT DO UPDATE.

**4. etl\_flow.py**

Defines the Metaflow workflow (ETLFlow) to orchestrate data ingestion and transformation scripts.

* **Workflow**:
  + **Steps**:
    1. **start**: Initializes workflow.
    2. **run\_data\_ingestion**: Executes data\_ingestion.py.
    3. **run\_data\_transformation**: Executes data\_transformation.py.
    4. **end**: Marks completion of workflow.
  + **Execution**:
    1. Command: python etl\_flow.py run
    2. Result: Sequentially executes data ingestion and transformation, ensuring controlled workflow.