Data Analyst Intern Exercise

**Exercise 1**

**Objective:** Transform current employee data from a columnar format into a historical, row-based versioning format suitable for database storage using Python.

**Task Overview:** Your task is to convert an input CSV file containing employee data into a structured format representing historical records of employee compensation, engagement, and performance reviews using Python. The new format requires transforming columnar data into a row-based historical versioning system for insertion into our data warehouse.

1. Reading the Input CSV File:

The code begins by importing the pandas library and reading the input CSV file containing employee data into a pandas DataFrame (`df`).

2. Sorting the DataFrame:

The DataFrame is sorted by 'Employee Code' and 'Date of Joining' to ensure chronological order.

3. Calculating Effective and End Dates:

A function named `calculate\_dates` is defined to derive 'Effective Date' and 'End Date' for each historical record of an employee. The 'End Date' is set as one day before the next 'Effective Date' to prevent overlap. For the latest record of an employee, a far-future date ('2100-01-01') is assigned as the 'End Date'.

4. Data Transformation:

The function `calculate\_dates` is applied to each group of data grouped by 'Employee Code'. This ensures that each employee's historical records are processed independently.

5. Forward Filling Missing Values:

Missing values in columns related to compensation, reviews, and engagement are filled using forward fill (`ffill()`). This ensures that if data for a range is missing, the most recent past record for the same employee is used.

6. Selecting Relevant Columns:

The transformed DataFrame is then subset to select relevant columns for the output. These columns include employee identifiers, compensation details, review information, engagement scores, effective dates, and end dates.

7. Renaming Columns:

The column names are renamed according to the desired output format specified in the task overview.

8. Writing Transformed Data to CSV:

Finally, the transformed DataFrame is written to a new CSV file named 'output.csv' without including the index.

9. Files

\* [GitHub Link](https://github.com/rohanraj6101/Employee_history)

Summary:

In summary, the provided Python code reads employee data from an input CSV file, transforms it into a structured format representing historical records of employee compensation, engagement, and performance reviews as per the given instructions, and writes the transformed data to an output CSV file. The transformation ensures that historical data is properly versioned, missing values are handled appropriately, and the output format meets the requirements specified in the task overview.