**Documentation of Historical Data Transformation**

**Approach**

This script utilizes Python libraries like pandas and datetime to transform employee data from a columnar format into a row-based historical version suitable for data warehouse storage. The following steps outline the key functionalities:

**Data Import and Cleaning:**

* Reads the CSV data into a pandas DataFrame.
* Assumes the provided input data is accurate and consistent.

**Helper Function Definitions**:

* calculate\_end\_date: Calculates the "End Date" for each record by considering the subsequent record's "Effective Date" or using a far-future date (e.g., "2100-01-01") for the latest record.
* handle\_missing\_data: Fills missing data for compensation, engagement, and review scores by inheriting the most recent valid value from the same employee within the DataFrame.

**Data Transformation Loop:**

* Iterates through each row in the DataFrame.
* Creates a temporary list to store individual employee's historical records.
* Utilizes helper functions to calculate "End Date" and handle missing data for relevant columns.
* Constructs a new dictionary containing transformed data for each record:
* Employee Code, Manager Employee Code
* Last Compensation (based on most recent record)
* Compensation (current value from the loop)
* Last Pay Raise Date (derived from compensation changes)
* Variable Pay (assumed missing for this example)
* Tenure in Org (calculated using "Effective Date" and "End Date")
* Performance Rating (current value from the loop)
* Engagement Score (current value from the loop)
* Effective Date (current record's date)
* End Date (calculated using the helper function)
* Appends the constructed dictionary to the temporary list.

**Data Combination and Export**:

* Combines all temporary lists into a single DataFrame.
* Sorts the DataFrame by "Employee Code" and "Effective Date" for proper historical order.
* Exports the transformed data to a new CSV file named "output.csv" (user can specify a different filename).

**Assumptions**

* "Compensation 1" and "Compensation 2" represent the base salary and not other forms of compensation like bonuses.
* "Engagement 1" and "Engagement 2" represent engagement scores at specific points in time.
* "Date of Exit" field is not used in this transformation as the "End Date" is derived from subsequent records.
* You can choose between two options (commented out in the code) to address the type mismatch error during date subtraction:
* Option 1: Convert effective\_date to datetime.date before subtraction (recommended if most operations involve dates).
* Option 2: Convert end\_date to pandas.Timestamp before subtraction (better if most operations involve timestamps).
* This approach effectively transforms employee data into a historical format, enabling further analysis and storage within a data warehouse. You can adapt this methodology to incorporate additional functionalities or handle more complex data structures based on your specific data and requirements.