# How to use this Prompt Template

This template helps you streamline your data transformation to the CHECK-IN data format using an AI assistant. We provide this prompt template that guides a chatbot (like ChatGPT, GitHub Copilot, or Gemini) to write the necessary Python code for you, saving you from having to learn the specific format requirements.

**Workflow**

* **Use the Prompt Template**: Copy the text after the horizontal line and paste it into the AI chatbot of your choice.
* **Provide Metadata**: The chatbot will start an interactive dialogue. Follow its instructions to provide the required metadata (file and column names). The prompt is designed to work without needing your actual data content.
* **Review the Generated Code**: Treat the Python script from the chatbot as a draft and check it for accuracy, security and compliance with your internal policies.
* **Execute the Code**: Paste the validated script into the jupyter notebook *notebooks/ai\_assisted\_data\_transformation\_for\_checkin.ipynb*.

**Important Considerations**

* **Third-Party Service**: Please be aware that by using an external chatbot, you are interacting with a third-party service outside the *futureEXPERT* environment.
* **Confidentiality**: If your metadata (file or column names) is confidential, use a company-approved, private AI system.
* **Your Responsibility**: You are solely responsible for any prompts you share and for the code you execute. By using this template, you agree that our organisation accepts no liability for the output generated by third-party AI services.

## LLM Prompt Template: Generating a Python Script for Data Transformation

Objective:

This prompt is designed to gather information from a user about their current data, which may span multiple tables/files. Based on this, the Chatbot will generate a Python script (using the pandas library) to read, join, and transform the user's source data into a single CSV file suitable for a timeseries forecasting check-in process.

**I. Target CSV File Format Requirements (Output of the Python script):**

The Python script should produce a single CSV file that strictly adheres to the following:

1. **File Type:** .csv.
2. **Table Structure:** Each row should represent observations for a specific timestamp. If multiple value columns are selected, these will appear as separate columns for that timestamp.
3. **Core Columns (in the output CSV):**
   * **Date Column (Exactly one):** Derived from the user's specified source date column(s) from one of the tables. If combined from multiple columns within a table, it might be named 'Date' by default, or the user can suggest a name. Otherwise, it will retain its original name (without table alias). It must contain parsed date/timestamp objects.
   * **Value Column(s) (One or more):** The user's original value column(s) from the source table(s), retaining their original names (without table aliases), and containing the numerical target values.

**II. Information Required from the User (Input for the LLM to generate the Python script):**

**A. Describe Your Current Data Source(s):**

1. **Input File(s) and Aliases:** What data files do you want to use? List your data file names and assign an alias to each.
   * Example: T1: source\_data.csv, T2: product\_lookup.csv
   * Specify here:
     + Alias1: FileName1 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + Alias2: FileName2 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + (Add more if needed)
2. **Input File Type(s)/Format(s):** For each file listed above (using its alias):
   * What type of file is it?
   * If it's a CSV or Text file: How are the values separated? E.g. by a comma?
   * If it's a CSV or Text file: What symbol is used for decimal numbers? E.g. a period?
   * Example: T1: CSV, delimiter=';', decimal='.'
   * Specify here:
     + Alias1: FormatDescription1 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + Alias2: FormatDescription2 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + (Add more corresponding to files in A.1)
3. **Combining Tables (If multiple files):** If you're using more than one file, how do they connect? We need to know which column in one file matches a column in another file to link them up.
   * Example: Join T1 (left) with T2 (inner) on T1.ProductID = T2.SKU
   * Example: Merge T1 with T2 on 'CommonKey', then merge result with T3 on 'AnotherKey'
   * Example: Connect 'My Sales Data' with 'Product Details'. The 'Product ID' column in 'My Sales Data' matches the 'SKU Number' column in 'Product Details'.
   * Your joining instructions: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   * *(If only one input file, write "N/A" or leave empty)*
4. **Current Date Column Name(s) & Source Table:** Where can we find the date information? Specify the table alias and column name(s) for your date information.
   * Example (single column): T1.OrderDate
   * Example (multiple columns): T1.Year, T1.Month, T1.Day
   * Specify here:
     + **Single** date column (Alias.ColumnName): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + OR **Multiple** date columns (Alias.Column1, Alias.Column2, ... ): \_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_, ...
5. **Current Date Column(s) Format/Arrangement (for columns in A.4):** How do the dates look in your file(s)?
   * If your date is in a **single column**, please give a format or an example of how a date looks in that column.
     + Example: 25/Dec/2023 or YYYY-MM-DD or %Y-%m-%d %H:%M:%S
     + Specify here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   * If your date is in **multiple date columns**: Describe how they form the date.
     + Example description: Year column has YYYY, Month column has MM, Day column has DD.
     + Specify here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. **Current Target Column Name(s) & Source Table(s):** Which column(s) have the numbers you want to use for your forecast? List each column.
   * Example (single value column): T1.SalesAmount
   * Example (multiple value columns): T1.QuantitySold, T1.Revenue
   * Specify here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ , ...
7. **Other Relevant Columns to Keep or Drop (Specify with Table Alias):**Besides the date and target columns, are there any other columns from your original files that you want to include in the final output file?
   * Example: T1.CustomerSegment, T2.ProductLine
   * Example: Keep all, except: T1.InternalNotes
   * Specify here:
     + Columns to keep (optional, comma-separated, with alias): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**B. Output File and Code Style Specification:**

1. **Output CSV File:**
   * output\_filename: What do you want to name the **output CSV file**?
     + Example: transformed\_for\_checkin.csv
     + Specify here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   * output\_delimiter: The character to use to separate values in the **output CSV file**. (Optional, defaults to ,)
     + Example: , or ;
     + Specify here: \_\_
   * output\_decimal: The character to use for decimal points in numerical values in the **output CSV file**. (Optional, defaults to .)
     + Example: . or ,
     + Specify here: \_\_
2. **Generated Python Code Style:** How should the Python code be structured?
   * Options:
     + notebook: Code snippets suitable for copying into a Jupyter Notebook (e.g., separate logical blocks, less formal function structure if appropriate).
     + script: A standalone executable script for using a terminal (with def main(): ... and if \_\_name\_\_ == "\_\_main\_\_": main()).
   * Example: notebook
   * Specify here (script or notebook): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**III. Task for the LLM:**

**Phase 1: Determine Mode of Interaction**

Your first task is to check if the user has already provided information in the template above.

1. **Check for Filled Values:** Examine the user's input. If key fields in Section II (like Input File(s), Date Column Name(s), and Value Column Name(s)) have been filled with actual values instead of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, assume the template is filled.
2. **Decide the Mode:**

* **If the template is mostly filled:** Proceed directly to **Phase 3: Generate the Script**.
* **If the template is mostly empty or partially filled:** Proceed to **Phase 2: Conversational Interview**.

**Phase 2: Conversational Interview (If Triggered)**

If the user has not filled out the template, do not ask them to "fill the template." Instead, initiate a guided conversation to gather the necessary information.

1. **Start the Conversation:** Greet the user and explain the process. For example: "I can help create a Python script to prepare your data. I'll ask a few questions to get the details I need. First..."
2. **Ask Questions Sequentially:** Ask for information one piece at a time, corresponding to the fields in Section II. Wait for the user's response before moving to the next question. Use the examples from the template to guide the user with each question.

* **Question Order:**

1. Start with **Input Files** (II.A.1). Ask if they're using one or multiple files, then ask for names and assign aliases.
2. Ask about **File Formats** for each file (II.A.2).
3. *If* they are using multiple files, ask about the **Joining Logic** (II.A.3).
4. Ask for the **Date Column(s)** (II.A.4). This is a critical question.
5. Ask about the **Date Format** (II.A.5).
6. Ask for the **Value/Target Column(s)** (II.A.6). This is also a critical question.
7. Ask about **Other Columns to Keep/Drop** (II.A.7), clearly stating this is optional.
8. Ask for the **Output Filename** (II.B.1), stating it's optional and will be defaulted if left blank.
9. Ask for the desired **Code Style** (script or notebook) (II.B.2), stating the default if left blank.
10. **Handle Partial Fills:** If the user already provided some information (e.g., they filled in the input file but nothing else), acknowledge it ("Okay, I see you're using my\_data.csv. Let's continue from there.") and only ask for the remaining necessary information.
11. **End the Interview:** Once you have collected all the *critical* information (at a minimum: input file(s), date column(s), and value column(s)), summarize it for the user. For example: "Great, I have what I need. I'll create a script to process sales.csv, using TransactionDate as the date and Amount as the target value. I will now generate the script for you." Then, proceed to Phase 3.

**Phase 3: Generate the Script (The Final Output)**

Based on the information (either from the initially filled template or collected during the conversational interview), generate the final output.

1. **Generate a Python script** using the pandas library, structured according to the user's choice in II.B.2 (or default to notebook).
2. **The script/notebook code should:**

* Use all the logic previously defined (handling file types, joins, date parsing, column selection, etc.).
* Apply sensible defaults for any optional information the user did not provide (e.g., output filename, delimiter, code style).
* Include clear comments and basic error handling.

1. **Provide a brief explanation of how to use the generated Python code**, tailored to the chosen style (script or notebook).