Processing CIL and IFS Datasets

Overview

This script processes **Community Infrastructure Levy (CIL)** and **Infrastructure Funding Statement (IFS)** datasets. It performs **data cleaning, filtering, and merging** with reference data before saving the processed datasets into separate CSV files.

The purpose of this exercise is to format the manually collected CIL and IFS data for the Data Platform. The first stage of the process of collecting the data involved manually searching local authority sites to obtain information such as start-date, end-date and adoption-dates for the CIL's, as well as URL pathway to the given local authority's website that contains PDFs on their CIL and IFS data. This data was collected in a spreadsheet, which is then processed by this script.

Step 1: Data Cleaning and Filtering

The script starts by **removing unnecessary records** based on specific conditions:

- Rows where **`document-type`** is `NaN` or `"OTHER"` are removed.
- Rows where **`document-url`** is `NaN` or `"OTHER"` are removed.
- Rows where **`adopted-date`** is `"No CIL"` are removed.
- `"None"` and `"Cannot find a page"` values are replaced with blank entries.
- `"N/A"` values in **`adopted-date`** are replaced with blanks.
- If **`adopted-date`** contains `"On hold"` or `"In Discussion"`, the value is copied to **`notes`** and the **`adopted-date`** field is set to blank.

Step 2: Merging with Reference Data

The script then **matches local authority codes** from a reference dataset.

1. Selects only relevant columns from the reference dataset:
   - `"local-authority-code"`
   - `"official-name"`
2. Extracts a code from both datasets:
   - In the main dataset, `"organisation"` is copied to a new column **`org_code`**.
   - In the reference dataset, `"official-name"` is copied to **`org_code`**.
3. **Performs a left merge**:
   - Matches rows using **`org_code`**.
   - Replaces `"organisation"` values with their corresponding `"local-authority-code"`.

4. **Prefixes** `"local-authority:"` to all values in `"organisation"`.
5. Drops unnecessary columns:
   - `"local-authority-code"`
   - `"official-name"`
   - `"org_code"`

Step 3: Creating and Saving Final Datasets

The script **splits** the cleaned data into two separate datasets:

1. **CIL Dataset**
   - Filters rows where `document-type` is `"CIL"`.
   - Saves the result as `cil_dataset.csv`.
2. **IFS Dataset**
   - Filters rows where `document-type` is `"IFS"`.
   - Saves the result as `ifs_dataset.csv`.

Summary

✅ **Cleans and filters** the dataset.
✅ **Handles missing values and special cases** in `"adopted-date"`.
✅ **Merges local authority reference data** to replace organisation names with official codes.
✅ **Splits data into CIL and IFS datasets** and saves them as CSV files.

This script ensures **accurate and structured data processing** for planning-related documents.

In [ ]:
```
import pandas as pd
import numpy as np

def cil_process(df, df_ref):
        """
    Processes a dataset containing information on Community Infrastructure Levy (CI
    Infrastructure Funding Statements (IFS), cleaning the data, handling missing va
    and mapping local authority codes before saving separate datasets for CIL and I

    Parameters:
    ----------
    df : pandas.DataFrame
        The main dataset containing CIL and IFS documents with associated metadata.

    df_ref : pandas.DataFrame
        A reference dataset mapping local authority codes to their official names.

    Processing Steps:
    ----------------
    - Drops rows where 'document-type' or 'document-url' contain NaN or "OTHER".
```

```
    - Removes rows where 'adopted-date' is "No CIL".
    - Replaces instances of `None` and "Cannot find a page" with an empty string.
    - Standardises 'adopted-date' by replacing "N/A" and `None` with an empty strin
    - Moves "On hold" or "In Discussion" values from 'adopted-date' to 'notes' and
    - Extracts relevant columns from `df_ref` for mapping.
    - Extracts organisation codes and maps them using local authority reference dat
    - Updates the 'organisation' column with mapped local authority codes, prefixed
    - Drops redundant columns after merging.
    - Saves cleaned CIL and IFS datasets separately as CSV files.

    Outputs:
    -------
    - Saves `cil_dataset.csv` containing rows where 'document-type' is "CIL".
    - Saves `ifs_dataset.csv` containing rows where 'document-type' is "IFS".
    """

    # Define values to drop
    doc_type_values_to_drop = [np.nan, 'OTHER']

    # Drop rows
    df = df[~df["document-type"].isin(doc_type_values_to_drop) & df["document-type"
    df = df[~df["document-url"].isin(doc_type_values_to_drop) & df["document-url"].

    # Drop rows where the value in any column is "No CIL"
    df = df[df["adopted-date"] != "No CIL"]

    # Strip `None` and "Cannot find a page", leaving cells blank
    df = df.replace([None, "Cannot find a page"], "")

    # Set `adopted-date` to blank if it contains "N/A" or is None
    df['adopted-date'] = df['adopted-date'].replace(["N/A", None], "")

    # Copy "On hold" or "In Discussion" to `notes` and set `adopted-date` to blank
    mask = df['adopted-date'].isin(['On hold', 'In Discussion'])
    df.loc[mask, 'notes'] = df.loc[mask, 'adopted-date']
    df.loc[mask, 'adopted-date'] = ""

    # Only select relevant columns from df_ref
    df_ref = df_ref[["local-authority-code", "official-name"]]

    # Extract the codes from both df0 and df1
    df['org_code'] = df['organisation']
    df_ref['org_code'] = df_ref['official-name']

    # Perform a left merge and replace organisation with extracted 3-letter codes
    df = pd.merge(df, df_ref, on='org_code', how='left')
    df['organisation'] = df['local-authority-code']

    # Prepend 'local-authority:' to each entry in the 'organisation' column
    df['organisation'] = "local-authority: " + df['organisation']

    # Drop reduntant columns
    df = df.drop(columns=['local-authority-code', 'official-name', 'org_code'])

    # Create and save CIL dataset
    cil_df = df[df["document-type"]=="CIL"]
    cil_df.to_csv('cil_dataset.csv', index=False)

    # Create and save IFS dataset
    ifs_df = df[df["document-type"]=="IFS"]
    ifs_df.to_csv('ifs_dataset.csv', index=False)


# Load dataset
```

```
df = pd.read_csv("CIL_schedule_documents - Sheet1.csv")
# Load the organisation reference data
df_ref = pd.read_csv("C:/Users/DanielGodden/Documents/planning_data/local_plan_data

cil_process(df, df_ref)
```