Endpoint Provision Check Script

This script audits active `endpoint` entries from the Digital Land Datasette to identify which endpoints are **missing expected provisions** — i.e. they do not have a matching `(dataset, organisation)` provision.

What the Script Does

1. **Fetches and filters active records**:
   - `endpoint.csv` : only active endpoints (`end_date` is `NaN`)
   - `organisation.csv` : only active organisations (`end_date` is `NaN`)
   - `source.csv` : used to link endpoints to organisations
   - `resource_endpoint.csv` + `resource_dataset.csv` : used to resolve datasets via resource mappings
   - `provision.csv` : used to determine valid `(dataset, organisation)` combinations
2. **Cleans and joins the data**:
   - Cleans `organisation_ref` fields by removing `digital-land:` prefixes
   - Merges:
     - `endpoint` with `source` and `organisation`
     - `endpoint` with `resource` and `dataset`
   - Builds a final metadata table per endpoint: `endpoint` , `endpoint_url` , `organisation` , `dataset`
3. **Identifies missing provisions**:
   - Performs a left join between the final endpoint metadata and the provisions table
   - Filters to only show unmatched records (i.e. endpoints with no provision)
4. **Handles `.pdf` links**:
   - Adds support for filtering `.pdf` URLs from the results
   - Saves two outputs:
     - `flag_endpoints_no_provision.csv` : the main report (optionally excluding `.pdf` URLs)
     - `flag_endpoints_pdf_only.csv` : contains only the `.pdf` endpoint rows

Command-Line Arguments

Argument	Description
`--output-dir`	Folder to save the CSV outputs
`--include-pdf`	(Optional) Include `.pdf` URLs in the main CSV output

Output Files

- `flag_endpoints_no_provision.csv` : endpoints with no matching provision
- `flag_endpoints_pdf_only.csv` : endpoints with `.pdf` links (always saved)

Notes

- All HTTP resources are streamed from Datasette using `?_stream=on` for efficiency.
- Assumes endpoint–organisation–dataset mappings are consistent through `source` and `resource_*` tables.

In [ ]:
```
import pandas as pd
import argparse
import os

def endpoint_provisions_check(output_dir, include_pdf):
    # Fetch and filter Endpoint table
    endpoint_url = "https://datasette.planning.data.gov.uk/digital-land/endpoint.cs
    df0 = pd.read_csv(endpoint_url)
    df0 = df0[df0['end_date'].isna()]  # Keep only active endpoints
    df_endpoint = df0[["endpoint", "end_date", "endpoint_url"]].copy()

    # Fetch and process Source table
    source_url = "https://datasette.planning.data.gov.uk/digital-land/source.csv?_s
    df1 = pd.read_csv(source_url)
    df1["organisation_ref"] = df1["organisation"].str.replace(r"^.*?:", "", regex=T
    df_source = df1[["endpoint", "organisation_ref"]].copy()

    # Fetch and filter Organisation table
    org_url = "https://datasette.planning.data.gov.uk/digital-land/organisation.csv
    df2 = pd.read_csv(org_url)
    df2 = df2[df2['end_date'].isna()]
    df2["reference"] = df2["reference"].astype(str)
    df_org = df2[["name", "reference"]].copy()
    df_org.rename(columns={"name": "organisation", "reference": "organisation_ref"}

    # Fetch and deduplicate Resource_endpoint table
    resource_endpoint_url = "https://datasette.planning.data.gov.uk/digital-land/re
    df3 = pd.read_csv(resource_endpoint_url)
    df_resource_endpoint = df3[["endpoint", "resource"]].drop_duplicates(subset="er

    # Fetch and deduplicate Resource_dataset table
    resource_dataset_url = "https://datasette.planning.data.gov.uk/digital-land/res
    df4 = pd.read_csv(resource_dataset_url)
    df_resource_dataset = df4[["dataset", "resource"]].drop_duplicates(subset="resc

    # Fetch and process Provisions table
    provisions_url = "https://datasette.planning.data.gov.uk/digital-land/provisior
    df5 = pd.read_csv(provisions_url)
    df5["organisation"] = df5["organisation"].str.replace(r"^.*?:", "", regex=True)
    df_provisions = df5[["dataset", "organisation"]].copy()
    df_provisions.rename(columns={"organisation": "organisation_ref"}, inplace=True
    df_provisions = df_provisions.merge(df_org, on="organisation_ref", how="left")
    df_provisions.drop(columns="organisation_ref", inplace=True)

    # Merge Endpoint with Source and Organisation
    df_ep_org = df_endpoint.merge(df_source, on="endpoint", how="left")
    df_ep_org = df_ep_org.merge(df_org, on="organisation_ref", how="left")
```

```
    df_ep_org = df_ep_org[["endpoint", "organisation"]]

    # Merge Endpoint with Resource and Dataset
    df_ep_ds = df_endpoint.merge(df_resource_endpoint, on="endpoint", how="left")
    df_ep_ds = df_ep_ds.merge(df_resource_dataset, on="resource", how="left")
    df_ep_ds = df_ep_ds[["endpoint", "dataset"]]

    # Final merge of endpoint metadata
    df_final = df_endpoint.merge(df_ep_org, on="endpoint", how="left")
    df_final = df_final.merge(df_ep_ds, on="endpoint", how="left")
    df_final = df_final[["endpoint", "endpoint_url", "organisation", "dataset", "er

    # Merge with provisioned dataset-organisation combinations
    df_full = df_final.merge(df_provisions, on=["dataset", "organisation"], how="le

    # Keep only rows not in provision
    df_missing = df_full[df_full["_merge"] == "left_only"].drop(columns=["_merge",

    # Separate PDF rows
    pdf_mask = df_missing["endpoint_url"].fillna("").str.lower().str.endswith(".pdf
    df_pdfs = df_missing[pdf_mask]
    df_non_pdfs = df_missing[~pdf_mask]

    # Save PDFs separately
    pdf_path = os.path.join(output_dir, "flag_endpoints_pdf_only.csv")
    df_pdfs.to_csv(pdf_path, index=False)

    # Save main CSV (either with or without PDFs)
    if include_pdf:
        final_output = df_missing
    else:
        final_output = df_non_pdfs

    csv_path = os.path.join(output_dir, "flag_endpoints_no_provision.csv")
    final_output.to_csv(csv_path, index=False)

def parse_args():
    """
    Parses command-line arguments for specifying the output directory
    and whether to include .pdf endpoint URLs.
    """
    parser = argparse.ArgumentParser(description="Check endpoints missing expected
    parser.add_argument(
        "--output-dir",
        type=str,
        required=True,
        help="Directory to save exported CSVs"
    )
    parser.add_argument(
        "--include-pdf",
        action="store_true",
        help="Include rows where endpoint_url ends in .pdf in main output"
    )
    return parser.parse_args()

if __name__ == "__main__":
    args = parse_args()
    endpoint_provisions_check(args.output_dir, include_pdf=True)

    # TEMP: for local testing without CLI
    #output_dir = "."
    #endpoint_provisions_check(output_dir, include_pdf=True)
```