02_transform

May 30, 2019

This notebook combines many spreadsheets across time into one dataset.

1 Transform

Opens and combines annual H2A visa disclosure data files from the Office of Foreign Labor Certification at the United States Department of Labor.

[51]:
```
import os
import pandas as pd
```

[52]:
```
import warnings
warnings.filterwarnings("ignore")
```

[53]:
```
input_dir = os.path.join(os.getcwd(), 'input')
output_dir = os.path.join(os.getcwd(), 'output')
```

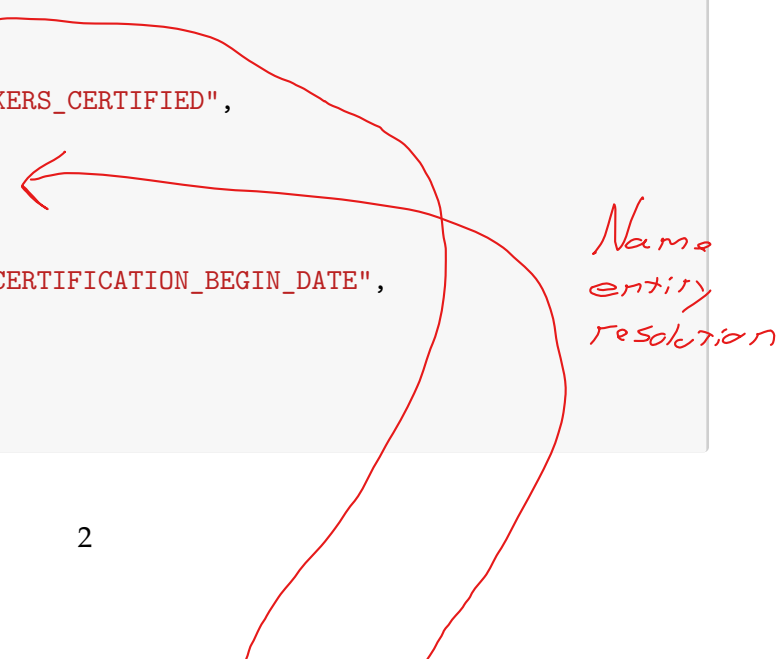
[54]:
```
%%capture
os.path.exists(input_dir) or os.mkdirs(intput_dir)
os.path.exists(output_dir) or os.mkdirs(output_dir)
```

A crosswalk that standardizes the headers from each year's file to a common schema

[55]:
```
schema = {
    2008: dict(
        case_number="CASE_NO",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="ALIEN_WORK_CITY",
        state="ALIEN_WORK_STATE",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",

    ),
    2009: dict(
        case_number="CASE_NO",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="ALIEN_WORK_CITY",
        state="ALIEN_WORK_STATE",
        job_title="JOB_TITLE",
```

```
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2010: dict(
        case_number="CASE_NO",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="ALIEN_WORK_CITY",
        state="ALIEN_WORK_STATE",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2011: dict(
        case_number="CASE_NO",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="ALIEN_WORK_CITY",
        state="ALIEN_WORK_STATE",
        crop="PRIMARY_CROP",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2012: dict(
        case_number="CASE_NO",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="ALIEN_WORK_CITY",
        state="ALIEN_WORK_STATE",
        crop="PRIMARY_CROP",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2013: dict(
        case_number="CASE_NO",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="ALIEN_WORK_CITY",
        state="ALIEN_WORK_STATE",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2014: dict(
        case_number="CASE_NO",
        case_status="CASE_STATUS",
```



```
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="WORKSITE_LOCATION_CITY",
        state="WORKSITE_LOCATION_STATE",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2015: dict(
        case_number="CASE_NUMBER",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="WORKSITE_CITY",
        state="WORKSITE_STATE",
        crop="PRIMARY_CROP",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2016: dict(
        case_number="CASE_NUMBER",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="WORKSITE_CITY",
        state="WORKSITE_STATE",
        crop="PRIMARY_CROP",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
    2017: dict(
        case_number="CASE_NUMBER",
        case_status="CASE_STATUS",
        workers_certified="NBR_WORKERS_CERTIFIED",
        employer="EMPLOYER_NAME",
        city="WORKSITE_CITY",
        state="WORKSITE_STATE",
        crop="PRIMARY_CROP",
        job_title="JOB_TITLE",
        certification_start_date="CERTIFICATION_BEGIN_DATE",
    ),
}
```

Recon-
Scaling
Schemas

[56]:
```
def transform_xls(year):
    """
    Transforms the H2A visa data from the provided year.
```

```
    Returns it cleaned up, deduped and standardized for consolidation with other years.
    """
    print("Transforming {} data".format(year))

    # Open the raw file
    input_path = os.path.join(
        input_dir,
        "{}.xlsx".format(year)
    )
    df = pd.read_excel(input_path)

    # Pull the schema for this year
    s = schema[year]

    # Reverse the schemaand standardize the column names
    i = {v: k for k, v in s.iteritems()}
    df = df.rename(columns=i)

    # Trim down to just the columns we want to keep
    trimmed = df[s.keys()]

    # Add the fiscal year from the file name
    trimmed['fiscal_year'] = year

    # Add a column with the calendar year where the jobs were certified
    trimmed.certification_start_date = pd.to_datetime(trimmed.certification_start_date)
    trimmed['certification_start_year'] = trimmed.apply(lambda x: x.certification_start_date.year, axis=1)

    # Trim applications from outside the time range we want to analyze
    trimmed = trimmed[
        (trimmed.certification_start_year > 2007) &
        (trimmed.certification_start_year < 2018)
    ]

    # Add incremental id
    trimmed['row_number'] = range(len(trimmed))

    # Combine that with the fiscal year into a unique id
    trimmed['latimes_id'] = trimmed.apply(lambda x: "{}-{}".format(x.fiscal_year, x.row_number), axis=1)

    # A little clean up on numbers
    trimmed.workers_certified.fillna(0, inplace=True)
    trimmed.workers_certified = trimmed.workers_certified.astype(int)
```

Subset data ?

Add repeat col.

Change Semantics to date

Add Calculated Field

Filter by 2007 < year < 2018

add col

add col for uniq id

replace na w/ zero

change type to int

```
    # Drop duplicate case records, as prescribed by the DOL.
    ## They say, "When analyzing H-2A program use, it is recommended to
    ## use the total number of workers certified (NBR_WORKERS_CERTIFIED)
    ## listed on the "master" record (i.e., the first employer record
    ## listed in the series of duplicate case numbers), which
    ## represents the total unique number of workers certified for
    ## the entire H-2A application, including all of the "sub" employer records.

    # HOWEVER! Close examination of the data shows that we cannot depend on the␣
↪master
    # master record coming first in the spreadsheets.

    # On the advice of the DOL, we will resort the data so that the record
    # with the maximum workers certified comes first and will then infer
    # that is must be the parent record.
    dupes = trimmed.groupby("case_number").filter(lambda x: len(x) > 1).
↪sort_values(
        ["case_number", "workers_certified"],
        ascending=[True, False]
    )
    master_cases = dupes.drop_duplicates("case_number", keep="first")
    master_cases['master_case'] = True
    sub_cases = dupes[~dupes['latimes_id'].isin(master_cases.latimes_id)]
    deduped = trimmed[~trimmed.case_number.isin(dupes.case_number)]
    deduped['master_case'] = False
    deduped = deduped.append(master_cases)

    # Filter it down to only those applications that were approved
    approved_status_list = [
        'DETERMINATION ISSUED - CERTIFICATION',
        'DETERMINATION ISSUED - PARTIAL CERTIFICATION',
        'Certified - Full',
        'Certified - Partial',
        # According to interview with a DOL official,
        # the expired certifications should be included.
        # They are only marked that way due to a bookkeeping error
        # when the records are pulled for public disclosure.
        'DETERMINATION ISSUED - CERTIFICATION EXPIRED',
        'DETERMINATION ISSUED - PARTIAL CERTIFICATION EXPIRED',
    ]
    certified = deduped[deduped.case_status.isin(approved_status_list)]
    sub_cases = sub_cases[sub_cases.case_status.isin(approved_status_list)]

    # Pass the DataFrames back
    return certified, sub_cases
```

Process all the annual files

[57]:
```
master_list, sub_list = zip(*[transform_xls(y) for y in sorted(schema.keys())])
```

```
Transforming 2008 data
Transforming 2009 data
Transforming 2010 data
Transforming 2011 data
Transforming 2012 data
Transforming 2013 data
Transforming 2014 data
Transforming 2015 data
Transforming 2016 data
Transforming 2017 data
```

Merge all the dataframes into big ones

Join by Union

[58]:
```
master_df = pd.concat(master_list)
```

[59]:
```
sub_df = pd.concat(sub_list)
```

Filter out any case number that appear in multiple years

[60]:
```
master_df.sort_values(
    ["case_number", "fiscal_year"],
    ascending=[True, False]
).drop_duplicates("case_number", keep="first", inplace=True)
```

Sort values and drop duplicates.

Standardize common variations on crop names

[61]:
```
clean_dict = {
    'Ag Eq Operator': 'Agricultural Equipment Operators',
    'Ag Eqp Operator;': 'Agricultural Equipment Operators',
    'Ag Equip Oper': 'Agricultural Equipment Operators',
    'Ag Equipment Operator': 'Agricultural Equipment Operators',
    'Ag equip operator': 'Agricultural Equipment Operators',
    'Agricultural Equipment Operator': 'Agricultural Equipment Operators',
    'Air Cured': 'Tobacco',
    'Apple Drops': 'Apples',
    'Asian Pears': 'Pears',
    'Bartlett Pears': 'Pears',
    'Bell Peppers': 'Peppers',
    'Dwarf Apples': 'Apples',
    'Romaine': 'Lettuce',
    'Spinich': 'Lettuce',
    'Spinach': 'Lettuce',
    'Onion': 'Onions',
    "Romaine Lettuce": "Lettuce",
    "Iceburg Lettuce": "Lettuce",
    "sheep": "Sheep",
    'Vineyards': 'Grapes',
    'Iceburg Lettuce': 'Lettuce',
    'Harvest Strawberries': "Strawberries",
```

Perform name entity resolution

Variant : Standard.

```
   "Vidalia Onions": "Onion",
   'Nursery': 'Nursery or greenhouse',
   'Nursery Work': 'Nursery or greenhouse',
   "Greenhouses": 'Nursery or greenhouse',
   "Nurseries & Greenhouses": 'Nursery or greenhouse',
   'Nursery and Greenhouse Worker': 'Nursery or greenhouse',
   'Nursery and Greenhouse Workers': 'Nursery or greenhouse',
   'Purple Hull Peas': 'Other crops',
   'Sugercane': 'Sugarcane',
   'Sugar Cane': 'Sugarcane',
   'Sugarcane': 'Sugarcane',
   'Sugar Beets': 'Beets',
   'Sweet Onions': 'Onions',
   'Sweet Peppers': 'Peppers',
   'Valencia Oranges': 'Oranges',
   'Watermelon': 'Watermelons',
   'Yellow Cherries': 'Cherries',
   'Jalapeno Peppers': 'Peppers',
   'Chile Peppers': 'Peppers',
   'Kale': "Lettuce",
   "Construction of Livestock buildings": 'Construction of Livestock␣
↪Buildings',
   "Construction of Livestock Buildings": "Construction of Livestock␣
↪Buildings",
   "Construction, Livestock Building": "Construction of Livestock Buildings",
   'Construction of Livestock  Buildings': 'Construction of Livestock␣
↪Buildings',
   'Construction of livestock buildings.': 'Construction of Livestock␣
↪Buildings',
   'Construction of livestocks buildings': 'Construction of Livestock␣
↪Buildings',
   'Constructionof Livestock Buildings': 'Construction of Livestock Buildings',
   'Consturcition of Livestock Buildings': 'Construction of Livestock␣
↪Buildings',
   'Construction Livestock Buildings': 'Construction of Livestock Buildings',
   'Custom Combine ': 'Custom Combine',
   "Chili Peppers": "Peppers",
   "Flue Cured": "Tobacco",
   "HARVEST, GATHER, COUNT AND PACKAGE;WATERMELON": "Watermelosn",
   "Hand harvest peaches": "Peaches",
   "Harvest Corn": "Corn",
   "Harvesting Citrus and other fruits": "Citrus",
   "Harvesting Watermelons": "Watermelons",
   "Hay And Straw": "Hay",
   "Hay/Straw": "Hay",
   "Open range cattle": "Cattle",
```

```
    "ALL PRODUCITON OF HANDLING SMALL BLAES OF HAY": "Hay",
    "Grapes Harvest": "Grapes",
    "Vineyards": "Grapes",
    "Grass Hay": "Hay",
    "Hay & Straw": "Hay",
    'Straw': "Hay",
    'Straw/Hay': "Hay",
    "Straw": "Hay",
    'Straw': "Hay",
    "Custom Harvester Wheat, corn, small grain": "Wheat",
    "OPEN RANGE CATTLE": "Cattle",
    "Wathermelon Unloaders and Watermelon Packing": "Watermelons",
    'Apple': "Apples",
    'Apple Harvest': "Apples",
    'Detassel Corn': "Corn",
    'Detasseling Corn': "Corn",
    'Calves': "Cattle",
    'Calving': "Cattle",
    'Cattle Herder': "Cattle",
    'Cattle Worker': "Cattle",
    'TOBACCO': "Tobacco",
    'Tobacco - Topping & Oiling': "Tobacco",
    'Tobacco -Setting': "Tobacco",
    'Tobbaco': "Tobacco",
    'Sweet Corn': "Corn",
    'Sweet corn, harvest': "Corn",
    'Watermelosn': "Watermelons", ← Typo
    'Stripping Tobacco': "Tobacco",
    "Farm worker": "Other",
    "General Farmworker": "Other",
    "Grain": "Grains",
    "Nursery Stock": "Nursery or greenhouse",
    "Sheepherder": "Sheep",
    "Farm Worker": "Other",
    "Sheep Shearer": "Sheep",
}
```

[62]:
```
master_df['latimes_crop'] = master_df.apply(lambda x: clean_dict.get(x.crop, x.
↪crop), axis=1)
```

[63]:
```
sub_df['latimes_crop'] = sub_df.apply(lambda x: clean_dict.get(x.crop, x.crop),␣
↪axis=1)
```

Create a combined file that merges master and sub cases.
First, sum up the total number of workers certified for subcases of each case number

[64]:
```
subcase_workers = sub_df.groupby("case_number").
↪agg(dict(workers_certified="sum")).reset_index()
```

[65]:
```
subcase_workers.columns = ['case_number', 'workers_subcases']
```

Rename cols

Merge that number to the master list for comparison

[66]:
```
combined_df = pd.merge(
    master_df,
    subcase_workers,
    how="left",
    on="case_number"
)
```

Left join

Append all subcases with case numbers that appear in the master list to the combined list

[67]:
```
linked_subcases = sub_df[sub_df.case_number.isin(combined_df.case_number)]
```

[68]:
```
linked_subcases['sub_case'] = True
```

Add Col

[69]:
```
combined_df = combined_df.append(linked_subcases)
```

Zero out the subcases worker count where it is empty

[70]:
```
combined_df.workers_subcases.fillna(0, inplace=True)
```

Replace Values

Calculate a net worker count by subtracting the subcase count from the total. This prevents double counting subcase positions included in master cases.

[71]:
```
combined_df['net_workers'] = combined_df.apply(
    lambda x: x.workers_certified - x.workers_subcases,
    axis=1
)
```

Create Calc Field.

Drop master cases with zero workers after subtracting their subcases

[72]:
```
net_df = combined_df[combined_df.net_workers > 0]
```

Filter rows

Write out the transformed files for analysis

[73]:
```
master_df.to_csv(
    os.path.join(output_dir, "transformed_master_cases.csv"),
    index=False,
    encoding="utf-8"
)
```

Print

[74]:
```
sub_df.to_csv(
    os.path.join(output_dir, "transformed_sub_cases.csv"),
    index=False,
    encoding="utf-8"
)
```

[75]:
```
net_df.to_csv(
    os.path.join(output_dir, "transformed_all_cases.csv"),
    index=False,
    encoding="utf-8"
)
```