Introduction

The National Center for Education Statistics (NCES) publishes an annual Digest of Education Statistics with many, many tables with data about education in the U.S. from enrollment, attainment, and costs. These tables are distributed as .xls files, and while they are fairly human readable, their format is non-tabular and not suited to combining for year-over-year comparison or loading into software like Tableau for visualization.

This notebook downloads all annual versions of `Table 330.20 - Average Undergraduate Tuition, fees, room, and board charges` since 2013, transforms them to a tabular format, then combines them to facilitate comparisons over time. The original code is hosted on Kaggle, runs automatically each month, and updates this Kaggle Dataset with updated data from the latest digest once it's available.

For an example of how this aggregated data might be used, you can check out this Tableau Dashboard I created with it.

Load Libraries

In [2]:
```
import datetime
import re
from IPython.display import display

import pandas as pd
import numpy as np
```

Define function for manipulating tables

Tables from the NCES Digest are distributed as semi-structured .xls files. Parsing them into a tabular format takes a lot of steps, which are defined in the function below.

In [3]:
```
def transform_nces(xls_path: str) -> pd.DataFrame:

    """
    Takes Table330.20, as provided by the NCES as an .xls file as
    part of their annual Digest of Education Statistics, and
    transforms it into a purely tabular format.

    This function assumes the file is unmodified from how it
    is specified on the NCES website. For this reason, it's
    recommended to supply the url path rather than downloading
    the file to the local machine.

    Example:
    xls_path = 'https://nces.ed.gov/programs/digest/d20/tables/xls/tabn330.20.xls'
    df = transform_nces330_20(xls_path)
    """

    # Read in the .xls file.
    df = pd.read_excel(xls_path, header = None)

    # Drop any rows with more than 2 cells are empty
    # (Cells with cross shape are not empty)
```

```
df.dropna(thresh=2,inplace=True)

# Remove row of sequential numbers
df = df[df.iloc[:,0] != '1']

# Get Expense categories, manually adding in
# "Tuition and required fees" in 4 locations
expense = df.iloc[2]
expense[-3:] = ['Tuition and required fees']*3
expense[7] = 'Tuition and required fees'
header1 = pd.DataFrame(expense).T

# Retrieve periods from 2nd row (index 1) (Ex. 2016-17)
# By iterating over cells and parsing with RegEx
period_row = df.iloc[1]

period_list = []
for element in period_row:
    if pd.isna(element):
        period = np.nan
    else:
        period = re.search("\d{4}-\d{2}", element).group(0)
    period_list.append(period)

# Forward fill the parsed periods, except for the
# last 3 values, which are the earlier period once
# followed by the more recent period twice
years = np.unique(period_list)
period_list[-3:] = [years[0]] + [years[1]]*2
header2 = pd.DataFrame(period_list).fillna(method = 'ffill').T

# Create header row displaying if column is:
# # Public In-State
# # Public Out-of-State
# # Private
header3 = pd.DataFrame([np.nan] + ['Public In-State']*6 + \
                       ['Public Out-of-State'] + ['Private']*6 + \
                       ['Public In-State']*2 + ['Public Out-of-State']).T

# Create header row displaying if column is 4-year or 2-year university
header4 = pd.DataFrame([np.nan] + ['4-year']*13 + ['2-year']*3).T

df = pd.concat([header1,
                header2,
                header3,
                header4,
                df]).reset_index(drop=True)

# Assign names to each header
df.iloc[0:4,0] = ['Expense','Year','Type','Length']

# Delete extra rows we're no longer referencing
df.drop(index = [4,5,6], axis = 0, inplace = True)

# Pivot so that previously defined headers are now
# categorical indexes, then designate top row as
# the new headers.
df = df.T
df.columns = df.iloc[0]
df = df[1:]

# Remove "." from State headers
df.columns = [string.replace(".","").strip() for string in df.columns]

# Pivot again so that State columns are another categorical index
id_vars = df.columns[:4]
```

```
    df = df.melt(id_vars = id_vars, var_name = "State", value_name = "Value")

    # Arrange Columns in desired order
    df = df[['Year','State','Type','Length','Expense','Value']]

    # Filter for latest year only
    df = df.iloc[np.where(df['Year'] == years[1])]

    # Return transformed DataFrame
    return(df)
```

Generate paths

The web addresses of each table are structured predictably, with only a small part of the URL changing based on the year of the publication you want to download from.

In [4]:
```
# Generate URLs to retrieve tables.
# Format:
# https://nces.ed.gov/programs/digest/d{year}/tables/xls/tabn330.20.xls

# Get current year
curr_year = datetime.date.today().strftime("%Y")

# Convert last two digits to integer
year_int = int(curr_year[2:])

# Get list of years, from 2013 to Current,
# in YY format
years = [str(i) for i in range(13,year_int)]

# Generate URLs
paths = [("https://nces.ed.gov/programs/digest/"
          f"d{year}"
          "/tables/xls/tabn330.20.xls") for year in years]
```

Preview the "raw" sheet format

With the paths generated, I show a recent table as an example. In an effort to be human readable, the table uses several headers with merged cells to convey information succinctly. They aren't particularly machine readable, however, especially if one wants to combine them to show changes over time.

In [5]:
```
preview = pd.read_excel(paths[-2], header=None)
display(preview.head(10))
```

	0	1	2	3	4	5	6	7	8	9	10
0	Table 330.20. Average undergraduate tuition, f...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	[In current dollars]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	State or jurisdiction	Public 4-year	NaN	NaN	NaN	NaN	NaN	NaN	Private 4-year	NaN	NaN

3	NaN	In-state, \n2019-20	NaN	In-state, 2020-21	NaN	NaN	NaN	Out-of-state tuition and required fees, 2020-21	2019-20	NaN	2020-21
4	NaN	Total	Tuition and required fees\1\	Total	Tuition and required fees\1\	Room	Board	NaN	Total	Tuition and required fees	Total
5	1	2	3	4	5	6	7	8	9	10	11
6	United States	21035.38	9349.14	21337.37	9374.52	6774.23	5188.63	27090.84	45925.28	32763.97	46312.51
7	Alabama	20497.45	10322.92	20992.79	10616.86	5777.15	4598.79	27004.54	27099.14	16755.53	27966.09
8	Alaska	19618.75	8296.67	22185.45	8849.2	6515.07	6821.18	25535.4	27774.07	19682.34	28361.63
9	Arizona	24016.21	11072.27	24681.26	11409.74	7729.15	5542.37	25426.11	22652.22	12895.01	22862.16

Load, transform, and combine tables

Using the large user-defined function from before inside of a loop, I download and reshape every annual publication of Table 330.20 since 2013 (the format differs before then).

In [6]:
```
# Iterate over paths, downloading and parsing each
# table with user-defined function, then adding them
# to a list.
df_list = []
for path in paths:
    print(f"Processing {path}")
    try:
        df_list.append(
            transform_nces(path)
        )
    except:
        print("Previous file not found. It likely has not been updated for this year yet
```

```
Processing https://nces.ed.gov/programs/digest/d13/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d14/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d15/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d16/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d17/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d18/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d19/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d20/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d21/tables/xls/tabn330.20.xls
Processing https://nces.ed.gov/programs/digest/d22/tables/xls/tabn330.20.xls
Previous file not found. It likely has not been updated for this year yet
```

Combine Tables

Now that the tables from each year have been collected, I combine them into one and display a preview of the tabular format.

In [7]:
```
nces_df = pd.concat(df_list, axis = 0)

# Preview a random sample of the data
nces_df.sample(5)
```

Out[7]:

	Year	State	Type	Length	Expense	Value
734	2018-19	Utah	Public In-State	2-year	Tuition and required fees	3843
139	2017-18	Delaware	Private	4-year	Room	5911.448
195	2012-13	Hawaii	Public In-State	4-year	Tuition and required fees	7731.203
614	2020-21	Oregon	Public Out-of-State	4-year	Tuition and required fees	33935.19
198	2015-16	Hawaii	Public Out-of-State	4-year	Tuition and required fees	27910.831

Further edits

Now that the tables are combined, more edits are implemented. These are edits that do not need to be done on each table individually, and it is more efficient to call most methods from the `pandas` library once on a large DataFrame rather than many times on small DataFrames.

In [8]:
```
# Remove Nationwide average
nces_df = nces_df[nces_df['State'] != 'United States']

# Remove Total Expense
nces_df = nces_df[nces_df['Expense'] != 'Total']

# Replace cross symbol with NaN, and drop
nces_df.replace("†", np.nan, inplace = True)
nces_df.dropna(inplace=True)

# Simplify Year to only the end year
nces_df['Year'] = "20" + nces_df['Year'].str.slice(start=5)

# Create an explicit Public Out-of-State Room/Board. This figure is
# the same as in-state Room/Board.
outstate_room_board = nces_df[(nces_df['Type'] == 'Public In-State') &
                              ((nces_df['Expense'] == 'Room') |
                               (nces_df['Expense'] == 'Board'))].copy()

outstate_room_board['Type'] = 'Public Out-of-State'
nces_df = pd.concat( [nces_df, outstate_room_board], axis = 0)

# Combine Room and Board into a single expense,
# by first separating room and board from tuition,
# aggregating summing them, then re-attaching
# Room/Board & Tuition/Fees

# This approach omits "Total" expenses. This is by design
room_board = nces_df[(nces_df['Expense'] == 'Room')|
                     (nces_df['Expense'] == 'Board')]

grouped = room_board.groupby(['Year','State','Type','Length'],
                             as_index = False)[['Value']].sum()

grouped['Expense'] = 'Room/Board'
tuition = nces_df[(nces_df['Expense'] == 'Tuition and required fees')]

nces_df = pd.concat([ tuition, grouped ], axis = 0)
```

```
# Simplify "Tuition and required Fees" to "Fees/Tuition"
nces_df['Expense'] = np.where(nces_df['Expense'] == "Tuition and required fees",
                              "Fees/Tuition", nces_df["Expense"])

# Round to nearest dollar. Cents aren't relevant
# for something as expensive as college
nces_df["Value"] = round(nces_df["Value"], 0).astype("int")

# Sort dataframe as desired
nces_df.sort_values(['Year','State','Type','Length','Expense'],inplace = True)

# Reset Index, accounting for concatenated tables and
# dropped rows
nces_df = nces_df.reset_index(drop = True)
```

View a sample

All edits are complete, and a preview of the finished product is displayed

In [9]:
```
nces_df.sample(5)
```

Out[9]:

	Year	State	Type	Length	Expense	Value
2111	2018	Illinois	Public In-State	2-year	Fees/Tuition	3891
1927	2017	South Carolina	Private	4-year	Room/Board	9225
2016	2018	Alabama	Public In-State	4-year	Fees/Tuition	9827
1497	2016	Oklahoma	Public In-State	4-year	Room/Board	7998
1322	2016	Indiana	Public Out-of-State	2-year	Fees/Tuition	7992

Write to .csv

This .csv is automatically uploaded to this Kaggle dataset: Avg Cost of Undergraduate College by State in a separate process.

In [10]:
```
# write to csv file
nces_df.to_csv("nces330_20.csv", index=False)
```

Conclusion & Link

This data is useful for getting a bird's eye view of how the cost of education in the United States has changed (mainly by getting more expensive!) over the last decade. It also is useful for showing how costs differ between states, types of University, and degree types. You can see how I used this transformed data to create a dashboard for easy comparisons at this Tableau Public link.