Data Wrangling

Data source:
The data was retrieved from the US Department of Labor from the following link:
https://www.foreignlaborcert.doleta.gov/performancedata.cfm
For this explanation only data from 2017 was taken into account.

Process:
The data being an excel file, it was loaded using Pandas' ExcelFile() method. After examination of its containing sheets, the proper one was parsed into a Pandas data frame:

```
#Open excel data file
file = './Data/H-1B_Disclosure_Data_FY17.xlsx'
data = pd.ExcelFile(file)
```

```
#Check number of cheets in excel file
print(data.sheet_names)
```

```
['PDD_main']
```

```
#Read data frame from excel sheet
df1 = data.parse('PDD_main')
```

The data frame has several unnecessary columns for our project, so only the following columns were considered to create a clean data frame:

- EMPLOYER_NAME: Name of employer submitting labor condition application.
- JOB_TITLE: Title of the job.
- SOC_NAME: Occupational name associated with the SOC_CODE.
- FULL_TIME_POSITION: Y = Full Time Position; N = Part Time Position.
- PREVAILING_WAGE: Prevailing Wage for the job being requested for temporary labor condition.
- PW_UNIT_OF_PAY: Unit of Pay. Valid values include "Daily (DAI)," "Hourly (HR)," "Bi-weekly (BI)," "Weekly (WK)," "Monthly (MTH)," and "Yearly (YR)".
- WORKSITE_CITY :City information of the foreign worker's intended area of employment.
- WORKSITE_COUNTY: County information of the foreign worker's intended area of employment.
- WORKSITE_STATE: State information of the foreign worker's intended area of employment.
- WORKSITE_POSTAL_CODE: Zip Code information of the foreign worker's intended area of employment.

The columns above were renamed for better clarity respectively as:
'employer', 'job_title', 'occupational_name', 'full_time', 'prevailing_wage', 'wage_period', 'city', 'county', 'state', 'postal_code'.

Around 0.21% of the rows contained NULL values. The amount being insignificant, it was decided to eliminate them:

```
#Drop rows with null values and reset index
df_drop = df1_reduced.dropna(axis=0, how='any').reset_index(drop=True)
```

Columns ‘full_time’ and ‘wage_period’ were checked for their unique possible values to check consistency with data set documentation:

```
#Possible values for column full_time
df_drop.full_time.unique()
```

```
array(['Y', 'N'], dtype=object)
```

```
#Possible values for wage_period
df_drop.wage_period.unique()
```

```
array(['Year', 'Hour', 'Month', 'Week', 'Bi-Weekly'], dtype=object)
```

The results agreed with the documentation, so we proceeded to check the ‘wage_period’ distribution for rows without full time employment. Only 14 rows showed a yearly wage period which is contradictory with part-time employment. Part-time employment is associated with hourly pay, so those 14 rows were dropped from the data frame:

```
#Wage period distribution for non yearly salaries
df_drop.wage_period[df_drop.full_time == 'N'].value_counts()
```

```
Hour    13826
Year       14
Name: wage_period, dtype: int64
```

```
#There are only 14 rows with Year value for wage_period, we drop them
#and for the others later we are going to
#calculate a yearly salary based on their hourly salary
drop_list = df_drop.wage_period[(df_drop.full_time == 'N') & \
                                (df_drop.wage_period == 'Year')].index
df_drop = df_drop.drop(drop_list)
```

To reduce memory usage, the columns 'job_title', 'occupational_name', 'full_time', 'wage_period', 'state' and 'postal_code' were turned to ‘category’ type.

10 rows were found to have a value of zero for the prevailing wage, so they were filtered out from the data frame:

```
#Check how many rows have a zero value
len(df_drop[df_drop.prevailing_wage == 0])
```

```
10
```

```
#Drop rows with zero value for prevailing_wage
df_drop = df_drop[df_drop.prevailing_wage > 0]
df_drop.describe()
```

To make the wage data consistent, all fields from column ‘prevailing_wage’ were turned into yearly salaries with the information provided by the column ‘wage_period’ . This was achieved by creating a helper function called ‘annual_salary’ and then by applying it to the data frame:

```
#Function to change non annual salaries to yearly salaries
def annual_salary(row):
    if row.wage_period == 'Year': return row.prevailing_wage
    if row.wage_period == 'Month': return row.prevailing_wage * 12
    if row.wage_period == 'Bi-Weekly': return row.prevailing_wage * 26
    if row.wage_period == 'Week': return row.prevailing_wage * 52
    if row.wage_period == 'Hour': return row.prevailing_wage * 2080
```

```
#Convert all salaries to yearly salaries in data frame using
#the function annual_salary
df_drop.prevailing_wage = df_drop.apply(annual_salary, axis = 1)
```

Given that the salaries were now expressed in a yearly format and assuming full time employment, the columns ‘full_time’ and ‘wage_period’ were eliminated:

```
#Drop wage_period and full_time columns from data frame
df_drop = df_drop.drop(['full_time', 'wage_period'], axis=1)
```

Finally annual salaries bigger than $5 million (127 of them) and smaller than $15 000 (only 1) being considered outliers, their rows were filtered out from the data frame.

```
#We are going to ignore those lines
df_drop = df_drop[df_drop.prevailing_wage < 1e+06]
```

```
#We drop that line
df_drop = df_drop[df_drop.prevailing_wage > 15000]
```

Finally, the indexes in the data frame are reset and the obtained clean data frame saved into a csv file:

```
df_clean = df_drop.reset_index(drop=True)
```

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 623194 entries, 0 to 623193
Data columns (total 8 columns):
employer             623194 non-null object
job_title            623194 non-null category
occupational_name    623194 non-null category
prevailing_wage      623194 non-null float64
city                 623194 non-null object
county               623194 non-null object
state                623194 non-null category
postal_code          623194 non-null category
dtypes: category(4), float64(1), object(3)
memory usage: 28.4+ MB
```

```
df_clean.to_csv('./Data/H1B_2017.csv')
```