HW 2- Extracting Data from Excel- Arianna Burford 3/25/24

```
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\arian\appdata\local\programs\python\python311\lib\site-packages (3.1.2)
Requirement already satisfied: et-xmlfile in c:\users\arian\appdata\local\programs\python\python311\lib\site-packages (from openpyxl) (1.1.0)

[notice] A new release of pip is available: 23.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
import openpyxl
```

```
import openpyxl

xlsx_file_path = 'unicef_sowc.xlsx'
workbook = openpyxl.load_workbook(xlsx_file_path)
```

```
sheet_names = workbook.sheetnames

print("Names of the sheets in the workbook:")
for sheet_name in sheet_names:
    print(sheet_name)
```

```
Names of the sheets in the workbook:
Data Notes
Table 9
```

```
sheet _name = 'Table 9'
sheet = workbook [sheet_name]
```

```
  Cell In[25], line 1
    sheet _name = 'Table 9'
          ^
SyntaxError: invalid syntax
```

```
sheet_name = 'Table 9 '
sheet = workbook [sheet_name]
print (sheet)
```

```
<Worksheet "Table 9 ">
```

```
print (dir(sheet))
```

```
['BREAK_COLUMN', 'BREAK_NONE', 'BREAK_ROW', 'HeaderFooter', 'ORIENTATION_LANDSCAPE', 'ORIENTATION_PORTRAIT', 'PAPERSIZE_A3', 'PAPERSIZE_A4', 'PAPERSIZE_A
4_SMALL', 'PAPERSIZE_A5', 'PAPERSIZE_EXECUTIVE', 'PAPERSIZE_LEDGER', 'PAPERSIZE_LEGAL', 'PAPERSIZE_LETTER', 'PAPERSIZE_LETTER_SMALL', 'PAPERSIZE_STATEMEN
T', 'PAPERSIZE_TABLOID', 'SHEETSTATE_HIDDEN', 'SHEETSTATE_VERYHIDDEN', 'SHEETSTATE_VISIBLE', '_WorkbookChild__title', '__class__', '__delattr__', '__deli
tem__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__i
nit__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
'__setitem__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_add_cell', '_add_column', '_add_row', '_cells', '_cells_by_col', '_cells_by_
row', '_charts', '_clean_merge_range', '_comments', '_current_row', '_default_title', '_drawing', '_get_cell', '_hyperlinks', '_id', '_images', '_invalid
_row', '_move_cell', '_move_cells', '_parent', '_path', '_pivots', '_print_area', '_print_cols', '_print_rows', '_rel_type', '_rels', '_setup', '_table
s', 'active_cell', 'add_chart', 'add_data_validation', 'add_image', 'add_pivot', 'add_table', 'append', 'array_formulae', 'auto_filter', 'calculate_dimen
sion', 'cell', 'col_breaks', 'column_dimensions', 'columns', 'conditional_formatting', 'data_validations', 'defined_names', 'delete_cols', 'delete_rows',
'dimensions', 'encoding', 'evenFooter', 'evenHeader', 'firstFooter', 'firstHeader', 'freeze_panes', 'insert_cols', 'insert_rows', 'iter_cols', 'iter_row
s', 'legacy_drawing', 'max_column', 'max_row', 'merge_cells', 'merged_cell_ranges', 'merged_cells', 'mime_type', 'min_column', 'min_row', 'move_range',
'oddFooter', 'oddHeader', 'page_margins', 'page_setup', 'parent', 'path', 'print_area', 'print_options', 'print_title_cols', 'print_title_rows', 'print_t
itles', 'protection', 'row_breaks', 'row_dimensions', 'rows', 'scenarios', 'selected_cell', 'set_printer_settings', 'sheet_format', 'sheet_properties',
'sheet_state', 'sheet_view', 'show_gridlines', 'tables', 'title', 'unmerge_cells', 'values', 'views']
```

```
print (sheet. rows)
```

```
<generator object Worksheet._cells_by_row at 0x000002107FF132E0>
```

```
help (sheet. rows)
```

```
Help on generator object:

_cells_by_row = class generator(object)
 |  Methods defined here:
 |
 |  __del__(...)
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __next__(self, /)
 |      Implement next(self).
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __sizeof__(...)
 |      gen.__sizeof__() -> size of gen in memory, in bytes
 |
 |  close(...)
 |      close() -> raise GeneratorExit inside generator.
 |
```

```
send(...)
    send(arg) -> send 'arg' into generator,
    return next yielded value or raise StopIteration.

throw(...)
    throw(value)
    throw(type[,value[,tb]])

    Raise exception in generator, return next yielded value or raise
    StopIteration.

----------------------------------------------------------------------
Data descriptors defined here:

gi_code

gi_frame

gi_running

gi_suspended

gi_yieldfrom
    object being iterated by yield from, or None
```

```
for row in sheet.rows:
    for cell in row:
        print(cell.value, end='\t')
    print()
```

```
None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    TABLEAU 9. PROTECTION DE L'ENFANT       None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    TABLA 9. PROTECCIÓN INFANTIL    None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None    None    None    None    None    None
None    Countries and areas     None    None    Child labour (%)+
2005-2012*      None    None    None    None    None    Child marriage (%)
2005-2012*      None    None    None    Birth registration (%)+
2005-2012*      None    Female genital mutilation/cutting (%)+
2002-2012*      None    None    None    None    None    Justification of wife beating (%)
2005-2012*      None    None    None    Violent discipline (%)+
```

[34]:
```
for row_index, row_values in enumerate(sheet.iter_rows(min_row=1, values_only=True), start=1):
    row_name = f"Row {row_index}"
    print(row_name)
    for cell_index, cell_value in enumerate(row_values, start=1):
        print(f" Cell {cell_index}: {cell_value}")
    print("-" * 20)
```

```
Row 1
 Cell 1: None
 Cell 2: TABLE 9. CHILD PROTECTION
 Cell 3: None
 Cell 4: None
 Cell 5: None
 Cell 6: None
 Cell 7: None
 Cell 8: None
 Cell 9: None
 Cell 10: None
 Cell 11: None
 Cell 12: None
 Cell 13: None
 Cell 14: None
 Cell 15: None
 Cell 16: None
 Cell 17: None
```

[40]:
```
start_row = None
# Iterate over the data
for row_index, row_values in enumerate(sheet.iter_rows(min_row=1, values_only=True), start=1):
    # Check if the row contains the header string
    if "Countries and areas" in row_values:
        # If found, go to the next row
        start_row = row_index + 1
        break

# Dictionary to store extracted data
extracted_data = {}
```

```python
# Dictionary to store extracted data
extracted_data = {}

# Loop through the rows starting with start_row
if start_row is not None:
    # Extract the data from each row (i.e. country, child labor, and other data)
    for row_index, row_values in enumerate(sheet.iter_rows(min_row=start_row, values_only=True), start=start_row):
        country_name = row_values[1]
        child_labor_data = {
            'total': row_values[4],
            'male': row_values[6],
            'female': row_values[8]
        }
        other_data = row_values[10:]
        # Store data in the dictionary
        extracted_data[country_name] = {'child_labor': child_labor_data, 'other_data': other_data}
        # Print the extracted data and associated row number
        print(f"Row {row_index}: {row_values[1:4]}")
        print(f"Child Labor (%): {row_values[4]} (total), {row_values[6]} (male), {row_values[8]} (female)")
        print(f"Other Data: {row_values[10:]}")
        print("-" * 50)
else:
    print("'Countries and areas' not found")
```

```
Row 6: (None, None, None)
Child Labor (%): None (total), None (male), None (female)
Other Data: (None, None, None, None, None, None, 'prevalence', None, None, None, 'attitudes', None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None)
--------------------------------------------------
Row 7: (None, None, None)
Child Labor (%): total (total), male (male), female (female)
Other Data: ('married by 15', None, 'married by 18', None, 'total', None, 'womena  ', None, 'girlsb ', None, 'support for the practicec', None, 'male',
None, 'female', None, 'total', None, 'male', None, 'female', None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None)
--------------------------------------------------
Row 8: ('FRENCH HEADINGS', 'Pays et zones', None)
Child Labor (%): Travail des enfants (%)+
2005-2012* (total), None (male), None (female)
Other Data: ('Mariage d'enfants (%)\n2005-2012*', None, None, None, 'Enregistrement\ndes naissances\n(%)+\n2005-2012*\n', None, 'Mutilations génitales
féminines/excision (%)+\n2002-2012*', None, None, None, None, None, 'Justification de la\nviolence conjugale (%)\n2005-2012*', None, None, None, 'Disci
pline imposée par la violence (%)+\n2005-2012*', None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None)
```

[42]:
```python
# Start from row 15, the first country
start_row = 15
# Stop at row 212, the last country
stop_row = 212

# Make sure when extracting data based on the countries
if 1 <= start_row <= sheet.max_row and 1 <= stop_row <= sheet.max_row and start_row <= stop_row:
    extracted_data = {}
    # Extract the data from each row
    for row_index, row_values in enumerate(sheet.iter_rows(min_row=start_row, max_row=stop_row, values_only=True), start=start_row):
        country_name = row_values[1]
        # Skip rows where country_name is None
        if country_name is None:
            continue
        child_labor_data = {
            'total': row_values[4],
            'male': row_values[6],
            'female': row_values[8]
        }
        other_data = row_values[10:]
        # Store data in the dictionary
        extracted_data[country_name] = {'child_labor': child_labor_data, 'other_data': other_data}

    # Print the names of the countries only
    print("\nExtracted Country Names:")
    for i, name in enumerate(extracted_data.keys(), start=1):
        print(f"{i}. {name}")
else:
    print("Error with start or stop row values")
```

```
Extracted Country Names:
1. Afghanistan
2. Albania
3. Algeria
4. Andorra
5. Angola
6. Antigua and Barbuda
7. Argentina
8. Armenia
9. Australia
10. Austria
11. Azerbaijan
12. Bahamas
13. Bahrain
14. Bangladesh
15. Barbados
```

```python
# Now that we are extracting the data from the countries
# Iterate the data starting with the first country and stop processing on the last country
if 1 <= start_row <= sheet.max_row and 1 <= stop_row <= sheet.max_row and start_row <= stop_row:
    extracted_data = {}

    # Get the headers
    headers_row = next(sheet.iter_rows(min_row=1, max_row=1, values_only=True))
    headers = headers_row[1:]

    # Extract the data from each row
    for row_index, row_values in enumerate(sheet.iter_rows(min_row=start_row, max_row=stop_row, values_only=True), start=start_row):
        country_name = row_values[1]
        # Skip rows where country_name is None
        if country_name is None:
            continue

        # Create a dictionary to store data for the current country
        country_data = {}

        # Process child labor data
        child_labor_labels = ['total', 'male', 'female']
        child_labor_values = [None if value in ('-', ' ', None) or not isinstance(value, (int, float)) else float(value) for value in row_values[4:7]]
        country_data['child_labor'] = dict(zip(child_labor_labels, child_labor_values))

        # Process other data
        other_data_labels = ['married_by_15', 'married_by_18']
        other_data_values = [None if value in ('-', ' ', None) or not isinstance(value, (int, float)) else float(value) for value in row_values[11:]]
        country_data['other_data'] = dict(zip(other_data_labels, other_data_values))

        # Add the country to dictionary
        extracted_data[country_name] = country_data

    # Print the extracted or pulled data that we are interested in
    for country, data in extracted_data.items():
        print(f"\nCountry: {country}")
        print("Data:")
        for category, values in data.items():
            print(f"{category}: {values}")
        print("-" * 50)
else:
    print("Error with start or stop row values")
```

```
Country: Afghanistan
Data:
child_labor: {'total': 10.3, 'male': None, 'female': 11.0}
other_data: {'married_by_15': None, 'married_by_18': 40.4}
--------------------------------------------------

Country: Albania
Data:
child_labor: {'total': 12.0, 'male': None, 'female': 14.4}
other_data: {'married_by_15': None, 'married_by_18': 9.6}
--------------------------------------------------

Country: Algeria
Data:
child_labor: {'total': 4.7, 'male': None, 'female': 5.5}
other_data: {'married_by_15': None, 'married_by_18': 1.8}
--------------------------------------------------
```

[ ]: