```
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /usr/local/lib/python3.10/dist-packages (3.1.2)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.10/dist-packages (from openpyxl) (1.1.0)
```

```
import openpyxl
```

```
xlsx_file_path = 'unicef_sowc.xlsx.xlsx'

workbook = openpyxl.load_workbook(xlsx_file_path)

print(workbook)
```

```
<openpyxl.workbook.workbook.Workbook object at 0x7bc802f5f970>
```

```
sheet_names = workbook.sheetnames

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

```
Names of the sheets in the workbook:
['Data Notes', 'Table 9 ']
['Data Notes', 'Table 9 ']
```

```
sheet_name = 'Table 9 '

sheet = workbook[sheet_name]
```

```
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
```

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

```
<generator object Worksheet._cells_by_row at 0x7bc802bb7bc0>
```

```
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).
 |
 |  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_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
```

```
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)
```

```
 Cell6: None
--------------------
 Cell7: None
--------------------
 Cell8: None
--------------------
 Cell9: None
--------------------
 Cell10: None
--------------------
 Cell11: None
--------------------
 Cell12: None
--------------------
 Cell13: None
--------------------
 Cell14: None
--------------------
 Cell15: None
--------------------
 Cell16: None
--------------------
 Cell17: None
--------------------
 Cell18: None
--------------------
 Cell19: None
--------------------
 Cell20: None
--------------------
 Cell21: None
--------------------
 Cell22: None
--------------------
 Cell23: None
--------------------
 Cell24: None
--------------------
 Cell25: None
--------------------
 Cell26: None
--------------------
 Cell27: None
--------------------
 Cell28: None
--------------------
 Cell29: None
--------------------
 Cell30: None
--------------------
 Cell31: None
--------------------
 Cell32: None
--------------------
 Cell33: None
--------------------
 Cell34: None
```

```
start_row = None

for row_index, row_values in enumerate(sheet.iter_rows(min_row=1, values_only=True), start=1):

  if "Countries and areas" in row_values:

    start_row = row_index + 1
    break

extracted_data = {}


if start_row is not None:

  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:]

    extracted_data [country_name] = {'child_labor': child_labor_data,'other_data': other_data}

    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")
```

```
 Child Labor (%): None (total), None(male), None (female)
 Other Data: (None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
--------------------------------------------------
Row 255: (None, None, None)
 Child Labor (%): None (total), None(male), None (female)
 Other Data: (None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
--------------------------------------------------
Row 256: ('DEFINITIONS OF THE INDICATORS', None, None)
 Child Labor (%): None (total), None(male), None (female)
 Other Data: (None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
```

```
start_row = 15

stop_row = 212

if 1 <= start_row <= sheet.max_row and 1 <= sheet.max_row and start_row <= stop_row:
  extracted_data = {}

  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]

    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:]

    extracted_data[country_name] = {'child_labor': child_labor_data,'other_data': other_data}


    print("\nExtracted Country:")
    for i, name in enumerate(extracted_data.keys(), start=1):
      print(f"{i}, {name}")

else:
  print("Error with start or stop row values")
```

```
141, Republic of Korea
142, Republic of Moldova
143, Romania
144, Russian Federation
145, Rwanda
146, Saint Kitts and Nevis
147, Saint Lucia
148, Saint Vincent and the Grenadines
149, Samoa
150, San Marino
151, Sao Tome and Principe
152, Saudi Arabia
153, Senegal
154, Serbia
155, Seychelles
156, Sierra Leone
157, Singapore
158, Slovakia
159, Slovenia
160, Solomon Islands
```

```
if 1 <= start_row <= sheet.max_row and 1 <= stop_row <= sheet.max_row and start_row <= stop_row:
  extracted_data = {}

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

  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]

    if country_name is None:
      continue

      country_data = {}

      child_labor_labels = ['total', 'male', 'female']
      child_labor_values = [None if value in ('-','',None) or not isinstance(value, (int,float)) else float(value)if isinstance (value, (int,
      country_data['child_labor'] = dict(zip(child_labor_labels, child_labor_values))

      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) if isinstance (value, (int,
      country_data['other_data'] = dict(zip(other_data_labels, other_data_values))

      extracted_data[country_name] = country_data

      for country, data in extracted_data.items():
          print(f"\nCountry: {country}")
          print("Data:")
          for category, values in data.items():
            print(f" {catgeory}: {values}")
          print("-" * 50)

else:
    print("Error with start or stop row values")
```