Assignment #2 – Extracting Data from Excel

To begin, open Jupyter Notebook through your preferred environment such as Google Colab or Anaconda. Once opened, create a new notebook. In Google Colab this is done by selecting **File** then **New Notebook**. Next, carefully write the code provided into each cell of the notebook as shown, including any comments.

Execute each cell by clicking the **Run or Play** button located on the left side of each cell. Should you have an error during execution, review the code within the cell. Pay close attention to potential typos, incorrect spacing, or misspellings, verifying each line against the provided code. Correct and rerun the cell.

Continue this process for all cells. Then save as a pdf by selecting **File** then **Print**.

Submit PDF to Canvas. Code and output must be clearly identified for full credit.

50 points

your_name_here_week_2_working_with_excel

March 21, 2024

[ ]:
```
!pip install openpyxl
```

[ ]:
```
# required packages `!pip install openpyxl`
import openpyxl
```

[ ]:
```
# name of the excel workbook
xlsx_file_path = 'unicef_sowc.xlsx'

# load the excel spreadsheet (workbook)
workbook = openpyxl.load_workbook(xlsx_file_path)

# print to make sure it loaded - `sanity` test or `debug` test
print(workbook)
```

[ ]:
```
# variable to hold the names of the sheets
sheet_names = workbook.sheetnames

# iterate through the sheet names and print them
print("Names of the sheets in the workbook:")
for sheet_name in sheet_names:
    print(sheet_name)
```

[ ]:
```
# name of the sheet you want to access
sheet_name = 'Table 9' # expect an error

# access the specific sheet by name
sheet = workbook[sheet_name]
```

[ ]:
```
# name of the sheet you want to access
sheet_name = 'Table 9 '  # fixed spacing

# access the specific sheet by name
sheet = workbook[sheet_name]

# print to make sure it loaded - `sanity` test or `debug` test
print(sheet)
```

[ ]:
```
# show what methods are available
print(dir(sheet))
```

[ ]:
```
# shows it is iterable (we can use a for loop)
print(sheet.rows)
```

[ ]:
```
# documentation on the`rows` method
help(sheet.rows)
```

[ ]:
```
# raw data from the worksheet

# iterate over each row and cell, then print the values
for row in sheet.rows:
    for cell in row:
        print(cell.value, end='\t')
    print()
```

[ ]:
```
# Print the contents of each row and cells, also improve readability

# iterate over each row
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)

    # iterate through each cell in the row
    for cell_index, cell_value in enumerate(row_values, start=1):
        print(f"  Cell {cell_index}: {cell_value}")

    # improve readability by adding a separator between each row
    print("-" * 20)
```

[ ]:
```
# skip to the header string "Countries and areas"
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 = {}
```

```
# 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 a 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")
```

[ ]:
```
# start from row 15, the first country
start_row = 15

# stop at row 212, the last country
stop_row = 212

# make sure when have are 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 country 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")
```

[ ]:
```
# 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) if isinstance(value, (int,␣
 ↪float)) else None 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) if isinstance(value, (int,␣
↪float)) else None 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")
```