1. Loading data from CSV and Excel files

You just got hired as the first and only data practitioner at a small business experiencing exponential growth. The company needs more structured processes, guidelines, and standards. Your first mission is to structure the human resources data. The data is currently scattered across teams and files and comes in various formats: Excel files, CSVs, JSON files, SQL databases…

The Head of People Operations wants to have a general view gathering all available information about a specific employee. Your job is to gather it all in a file that will serve as the reference moving forward. You will merge all of this data in a pandas DataFrame before exporting to CSV.

Data management at your company is not the best, but you need to start somewhere. You decide to tackle the most straightforward tasks first, and to begin by loading the company office addresses. They are currently saved into a CSV file, `office_addresses.csv` , which the Office Manager sent over to you. Additionally, an HR manager you remember interviewing with gave you access to the Excel file, `employee_information.xls` , where the employee addresses are saved. You need to load these datasets in two separate DataFrames.

In [83]:
```
# Import the library you need
import pandas as pd

# Load office_addresses.csv
df_office_addresses = pd.read_csv("datasets/office_addresses.csv")

# Load employee_information.xls
df_employee_addresses = pd.read_excel("datasets/employee_information.xl
s")

# Take a look at the first rows of the DataFrames
print(df_office_addresses.head())
print(df_employee_addresses.head())
```

```
          office office_country    office_city   office_street  \
0  Leuven Office             BE         Leuven  Martelarenlaan
1     ESB Office             US  New York City    Fifth Avenue
2  WeWork Office             GB         London      Old Street

   office_street_number
0                    38
1                   350
2                   207
  employee_id employee_last_name employee_first_name employee_country
\
0      A2R5H9             Hunman                Jax               BE
1      H8K0L6               Siff               Tara               GB
2      G4R7V0              Sagal              Gemma               US
3      M1Z7U9             Coates                Tig               FR

  employee_city      employee_street  employee_street_number
0        Leuven          Grote Markt                       9
1        London         Baker Street                     221
2      New-York         Perry Street                      66
3         Paris  Rue de l'Université                       7
```

In [84]:
```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

correct_office_addresses = pd.read_csv("datasets/office_addresses.csv")
correct_employee_addresses = pd.read_excel("datasets/employee_informatio
n.xls")

def test_office_addresses():
    assert correct_office_addresses.equals(df_office_addresses), \
    "It seems your there's something wrong with your `df_office_addresse
s` DataFrame.\n\
     Are you sure you loaded the `office_addresses.csv` located in the `
datasets` folder\n\
     using pandas `read_csv()` method?"

def test_employee_addresses():
    assert correct_employee_addresses.equals(df_employee_addresses), \
    "It seems your there's something wrong with your `df_employee_addres
ses` DataFrame.\n\
     Are you sure you loaded `employee_addresses.csv` located in the `da
tasets` folder\n\
     using pandas `read_excel()` method?"
```

Out[84]: 2/2 tests passed

2. Loading employee data from Excel sheets

It turns out the `employee_information.xls` file also holds information about emergency contacts for each employee in a second sheet titled `emergency_contacts`. However, this sheet was edited at some points, and the header was removed! Looking at the data, you were able to figure out what the header should be, and you confirmed that they were appropriate with the HR
manager: `employee_id`, `last_name`, `first_name`, `emergency_contact`, `emergency_contact_num`

In [85]:
```
# Load data from the second sheet of employee_information.xls
df_emergency_contacts = pd.read_excel("datasets/employee_information.xl
s", sheet_name=1, header=None)

# Declare a list of new column names
emergency_contacts_header = ["employee_id", "last_name", "first_name",
                             "emergency_contact", "emergency_contact_num
ber", "relationship"]

# Rename the columns
df_emergency_contacts.columns = emergency_contacts_header

# Take a look at the first rows of the DataFrame
df_emergency_contacts.head()
```

Out[85]:

	employee_id	last_name	first_name	emergency_contact	emergency_contact_number	relationsh
0	A2R5H9	Hunman	Jax	Opie Hurst	+32-456-5556-84	Brotl
1	H8K0L6	Siff	Tara	Wendy de Matteo	+44-020-5554-333	Sis
2	G4R7V0	Sagal	Gemma	John Newmark	+1-202-555-194	Husba
3	M1Z7U9	Coates	Tig	Venus Noone	+1-202-555-0130	W

In [86]:
```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

correct_emergency_contacts = pd.read_excel("datasets/employee_informatio
n.xls", sheet_name=1, header=None)
correct_emergency_contacts_header = ["employee_id", "last_name", "first_
name",
                                     "emergency_contact", "emergency_contact_num
ber", "relationship"]
correct_emergency_contacts.columns = correct_emergency_contacts_header


def test_emergency_headers():
    correct_emergency_contacts_header == emergency_contacts_header
    "It seems there's something wrong with your `emergency_contacts_head
er` list.\n\
     Are you sure you used the column names provided by the HR manager?"

def test_emergency_columns():
    correct_emergency_contacts.columns == df_emergency_contacts.columns
    "It seems there's something wrong with your DataFrame's column title
s.\n\
     Are you sure you used the list storing column names provided by the
HR manager?"

def test_emergency_contacts():
    assert correct_emergency_contacts.equals(df_emergency_contacts), \
    "It seems there's something wrong with your `df_emergency_contacts`
 DataFrame.\n\
     Are you sure you loaded `employee_information.xls` located in the `
datasets` folder\n\
     using pandas `read_excel()` method, specifying the correct sheet\n\
     and paying attention to the state of the headers?"
```

Out[86]:
```
3/3 tests passed
```

3. Loading role data from JSON files

All right, you're making good progress! Now the next step is to gather information about employee roles, teams, and salaries. This information usually lives in a human resources management system, but the Head of People Operations exported the data for you into a JSON file titles `employee_roles.json`.

Looking at the JSON file, you see entries are structured in a specific way. It is built as a Python dictionary: the keys are employee IDs, and each employee ID has a corresponding dictionary value holding role, salary, and team information. Here are the first few lines of the file:

```
{"001":
  {
    "title": "CEO",
    "monthly_salary": "$4500",
    "team": "Leadership"
  },
 ...
}
```

Load the JSON file to a variable `df_employee_roles`, choosing the appropriate orientation.

In [87]:
```
# Load employee_roles.json
df_employee_roles = pd.read_json("datasets/employee_roles.json", orient="index")
df_employee_roles = df_employee_roles.reindex(sorted(df_employee_roles.columns), axis=1)

# Take a look at the first rows of the DataFrame
df_employee_roles.head()
```

Out[87]:

	monthly_salary	team	title
A2R5H9	$4500	Leadership	CEO
H8K0L6	$4500	Leadership	CFO
G4R7V0	$3000	Sales	Business Developer
M1Z7U9	$2000	People Operations	Office Manager

In [88]:
```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

correct_employee_roles = pd.read_json("datasets/employee_roles.json", or
ient="index")
correct_employee_roles = correct_employee_roles.reindex(sorted(correct_e
mployee_roles.columns), axis=1)

records_employee_roles = pd.read_json("datasets/employee_roles.json", or
ient="records")

def test_records_orient():
    assert not records_employee_roles.reindex(sorted(records_employee_ro
les.columns), axis=1)\
                .equals(df_employee_roles.reindex(sorted(df_employee_role
s.columns), axis=1)), \
    "It seems you used the 'records', 'columns' or 'values' orientatio
n.\n\
     This puts the employee ID as column titles\n\
     and the employee title, monthly salary and team as the index.\n\
     You want the other way around: employee ID as index\n\
     and title, monthly salary and team as the column titles.\n\
     Try another `orient` value!"

def test_employee_roles():
    assert correct_employee_roles.equals(df_employee_roles), \
    "It seems there's something wrong with your `df_employee_roles` Data
Frame.\n\
     Are you sure you loaded `employee_roles.json` located in the `datas
ets` folder\n\
     using pandas `read_json()` method, specifying the correct orientati
on?"
```

Out[88]:
```
2/2 tests passed
```

4. Merging several DataFrames into one

You now have all the data required! All that's left is bringing it all in a unique DataFrame. This unique DataFrame will enable the Head of People Operations to access all employee data at once.

In this step, you will merge all DataFrames. In the next step, you will remove duplicates and reorganize the columns - don't worry about this for now.

In [89]:
```
# Merge df_emergency_contacts with df_employee_addresses
df_employees = df_employee_addresses.merge(df_emergency_contacts, how="l
eft", on="employee_id")

## Merge df_employee_roles with df_employees
df_employees = df_employees.merge(df_employee_roles, how="left", left_on
="employee_id",
                                  right_on=df_employee_roles.index)
#
## Merge df_office_adresses with df_employees
df_employees = df_employees.merge(df_office_addresses, how="left",
                                  left_on="employee_country", right_on=
"office_country")
#
## Take a look at the first rows of the DataFrame and its columns
print(df_employees.head())
print(df_employees.columns)
```

```
  employee_id employee_last_name employee_first_name employee_country
\
0      A2R5H9             Hunman                Jax               BE
1      H8K0L6               Siff               Tara               GB
2      G4R7V0              Sagal              Gemma               US
3      M1Z7U9             Coates                Tig               FR

  employee_city      employee_street  employee_street_number last_name
\
0        Leuven          Grote Markt                       9    Hunman
1        London         Baker Street                     221      Siff
2      New-York         Perry Street                      66     Sagal
3         Paris  Rue de l'Université                       7    Coates

  first_name emergency_contact emergency_contact_number relationship  \
0        Jax        Opie Hurst          +32-456-5556-84      Brother
1       Tara   Wendy de Matteo         +44-020-5554-333       Sister
2      Gemma      John Newmark           +1-202-555-194      Husband
3        Tig       Venus Noone          +1-202-555-0130         Wife

  monthly_salary               team               title         office
\
0          $4500         Leadership                 CEO  Leuven Office
1          $4500         Leadership                 CFO  WeWork Office
2          $3000              Sales  Business Developer     ESB Office
3          $2000  People Operations      Office Manager            NaN

  office_country    office_city   office_street  office_street_number
0             BE         Leuven  Martelarenlaan                  38.0
1             GB         London      Old Street                 207.0
2             US  New York City    Fifth Avenue                 350.0
3            NaN            NaN             NaN                   NaN
Index(['employee_id', 'employee_last_name', 'employee_first_name',
       'employee_country', 'employee_city', 'employee_street',
       'employee_street_number', 'last_name', 'first_name',
       'emergency_contact', 'emergency_contact_number', 'relationship',
       'monthly_salary', 'team', 'title', 'office', 'office_country',
       'office_city', 'office_street', 'office_street_number'],
      dtype='object')
```

In [90]:
```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Task 1
correct_office_addresses = pd.read_csv("datasets/office_addresses.csv")

correct_employee_addresses = pd.read_excel("datasets/employee_informatio
n.xls")

# Task 2
correct_emergency_contacts = pd.read_excel("datasets/employee_informatio
n.xls", sheet_name=1, header=None)
correct_emergency_contacts_header = ["employee_id", "last_name", "first_
name",
                                     "emergency_contact", "emergency_contact_num
ber", "relationship"]
correct_emergency_contacts.columns = correct_emergency_contacts_header

# Task 3
correct_employee_roles = pd.read_json("datasets/employee_roles.json", or
ient="index")
correct_employee_roles = correct_employee_roles.reindex(sorted(correct_e
mployee_roles.columns), axis=1)

# Task 4
correct_employees = correct_employee_addresses.merge(correct_emergency_c
ontacts,
                                                     how="left",
                                                     on="employee_id")
correct_employees = correct_employees.merge(correct_employee_roles,
                                            how="left",
                                            left_on="employee_id",
                                            right_on=correct_employee_ro
les.index)
correct_employees = correct_employees.merge(correct_office_addresses,
                                            how="left",
                                            left_on="employee_country",
                                            right_on="office_country")

def test_index_length():
    assert len(correct_employees.index) == len(df_employees.index), \
    "It looks like your DataFrame does not have the right count of colum
ns.\n\
     Be careful not to change the merge method used (\"left\"), as other
methods might drop rows."

def test_column_length():
    assert len(correct_employees.columns) == len(df_employees.columns),
 \
    "It looks like your DataFrame does not have the right count of colum
ns.\n\
```

```
     You should end up with 20 columns."

def test_column_titles():
    assert correct_employees.columns.to_list().sort() == df_employees.co
lumns.to_list().sort(), \
    "It looks like your DataFrame does not have the right columns title
s.\n\
     Here is the column index you should end up with:\n\
     `['employee_id', 'employee_last_name', 'employee_first_name',\n\
       'employee_country', 'employee_city', 'employee_street',\n\
       'employee_street_number', 'last_name', 'first_name',\n\
       'emergency_contact', 'emergency_contact_number', 'relationshi
p',\n\
       'monthly_salary', 'team', 'title', office', 'office_country',\n\
       'office_city', 'office_street', 'office_street_number']`."

def test_employees():
    assert correct_employees.equals(df_employees), \
    "It seems there's something wrong with your `df_employees` DataFram
e.\n\
     Are you sure you merged all the DataFrames in the order specified?
\n\
     You should always use a left join here."
```

Out[90]: 4/4 tests passed

5. Editing column names

Now that you merged all of your DataFrames into one let's make sure we have the information required by People Ops.

Currently, your `df employees` DataFrame has the following column titles: `employee_id`, `employee_last_name`, `employee_first_name`, `employee_country`, `employe`

The columns `employee last name` and `last name` are duplicates. The columns `employee_first_name` and `first_name` are duplicates as well. On top of this, People Ops wants to rename some of the columns:

- `employee id` should be `id`
- `employee country` should be `country`
- `employee city` should be `city`
- `employee street` should be `street`
- `employee_street_number` should be `street_number`

So your header should look like this in the end: `id`, `country`, `city`, `street`, `street_number`, `last_name`, `first_name`, `emergency_cont`

In [91]:
```
# Drop the columns
df_employees_renamed = df_employees.drop(["employee_first_name", "employee_last_name"], axis=1)

# Declare a list of new columns names
new_header = ["id",
              "country", "city", "street", "street_number",  "last_name", "first_name",
              "emergency_contact", "emergency_number", "emergency_relationship",
              "monthly_salary", "team", "title", "office", "office_country",
              "office_city", "office_street", "office_street_number"]

# Rename the columns
df_employees_renamed.columns = new_header

# Take a look at the first rows of the DataFrame
df_employees_renamed.head()
```

Out[91]:

	id	country	city	street	street_number	last_name	first_name	emergency_conta
0	A2R5H9	BE	Leuven	Grote Markt	9	Hunman	Jax	Opie Hu
1	H8K0L6	GB	London	Baker Street	221	Siff	Tara	Wendy de Matt
2	G4R7V0	US	New-York	Perry Street	66	Sagal	Gemma	John Newma
3	M1Z7U9	FR	Paris	Rue de l'Université	7	Coates	Tig	Venus Noo

In [92]:
```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Task 1
correct_office_addresses = pd.read_csv("datasets/office_addresses.csv")

correct_employee_addresses = pd.read_excel("datasets/employee_informatio
n.xls")

# Task 2
correct_emergency_contacts = pd.read_excel("datasets/employee_informatio
n.xls", sheet_name=1, header=None)
correct_emergency_contacts_header = ["employee_id", "last_name", "first_
name",
                                     "emergency_contact", "emergency_con
tact_number", "relationship"]
correct_emergency_contacts.columns = correct_emergency_contacts_header

# Task 3
correct_employee_roles = pd.read_json("datasets/employee_roles.json", or
ient="index")
correct_employee_roles = correct_employee_roles.reindex(sorted(correct_e
mployee_roles.columns), axis=1)

# Task 4
correct_employees = correct_employee_addresses.merge(correct_emergency_c
ontacts,
                                                     how="left",
                                                     on="employee_id")
correct_employees = correct_employees.merge(correct_employee_roles,
                                            how="left",
                                            left_on="employee_id",
                                            right_on=correct_employee_ro
les.index)
correct_employees = correct_employees.merge(correct_office_addresses,
                                            how="left",
                                            left_on="employee_country",
                                            right_on="office_country")

# Task 5
correct_employees_renamed = correct_employees.drop(["employee_first_nam
e", "employee_last_name"], axis=1)
correct_header = ["id", "country", "city", "street", "street_number", "l
ast_name", "first_name",
                  "emergency_contact", "emergency_number", "emergency_re
lationship",
                  "monthly_salary", "team", "title", "office", "office_c
ountry",
                  "office_city", "office_street", "office_street_numbe
r"]
correct_employees_renamed.columns = correct_header
```

```
print(correct_employees_renamed.head())

def test_column_name():
    assert correct_header == new_header, \
    "It seems your `new_column_names` is incorrect.\n\
     Are you sure you used the column names given by People Ops,\n\
     in the order they were provided?\n\
     Make sure there's no typo."

def test_employees_renamed():
    assert correct_employees_renamed.equals(df_employees_renamed), \
    "It seems there's something wrong with your `df_employees_oredered`
 DataFrame.\n\
     Are you sure you renamed the column titles and then reordered the
m?"
```

```
       id country      city                street  street_number last_na
me  \
0  A2R5H9      BE    Leuven           Grote Markt              9    Hunm
an
1  H8K0L6      GB    London         Baker Street            221      Si
ff
2  G4R7V0      US  New-York         Perry Street             66     Sag
al
3  M1Z7U9      FR     Paris  Rue de l'Université              7    Coat
es

  first_name emergency_contact  emergency_number emergency_relationship
\
0        Jax        Opie Hurst   +32-456-5556-84                Brother
1       Tara   Wendy de Matteo  +44-020-5554-333                 Sister
2      Gemma      John Newmark    +1-202-555-194                Husband
3        Tig       Venus Noone   +1-202-555-0130                   Wife

  monthly_salary               team               title         office
\
0          $4500         Leadership                 CEO  Leuven Office
1          $4500         Leadership                 CFO  WeWork Office
2          $3000              Sales  Business Developer     ESB Office
3          $2000  People Operations      Office Manager            NaN

  office_country    office_city   office_street  office_street_number
0             BE         Leuven  Martelarenlaan                  38.0
1             GB         London      Old Street                 207.0
2             US  New York City    Fifth Avenue                 350.0
3            NaN            NaN             NaN                   NaN
```

Out[92]: 2/2 tests passed

6. Changing column order

Now that you have the appropriate column names, you can reorder the columns.

In [93]:
```
# Declare a list for the new column's order and reorder columns
new_column_order = ["id", "last_name", "first_name", "title", "team", "m
onthly_salary",
                    "country", "city", "street", "street_number",
                    "emergency_contact", "emergency_number", "emergency_
relationship",
                    "office", "office_country", "office_city", "office_s
treet", "office_street_number"]

# Reorder the columns
df_employees_ordered = df_employees_renamed[new_column_order]

# Take a look at the result
df_employees_ordered.head()
```

Out[93]:

	id	last_name	first_name	title	team	monthly_salary	country	city	s
0	A2R5H9	Hunman	Jax	CEO	Leadership	$4500	BE	Leuven	
1	H8K0L6	Siff	Tara	CFO	Leadership	$4500	GB	London	
2	G4R7V0	Sagal	Gemma	Business Developer	Sales	$3000	US	New-York	
3	M1Z7U9	Coates	Tig	Office Manager	People Operations	$2000	FR	Paris	R l'Univ

In [94]:
```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Task 1
correct_office_addresses = pd.read_csv("datasets/office_addresses.csv")

correct_employee_addresses = pd.read_excel("datasets/employee_informatio
n.xls")

# Task 2
correct_emergency_contacts = pd.read_excel("datasets/employee_informatio
n.xls", sheet_name=1, header=None)
correct_emergency_contacts_header = ["employee_id", "last_name", "first_
name",
                                     "emergency_contact", "emergency_contact_num
ber", "relationship"]
correct_emergency_contacts.columns = correct_emergency_contacts_header

# Task 3
correct_employee_roles = pd.read_json("datasets/employee_roles.json", or
ient="index")
correct_employee_roles = correct_employee_roles.reindex(sorted(correct_e
mployee_roles.columns), axis=1)

# Task 4
correct_employees = correct_employee_addresses.merge(correct_emergency_c
ontacts,
                                                     how="left",
                                                     on="employee_id")
correct_employees = correct_employees.merge(correct_employee_roles,
                                            how="left",
                                            left_on="employee_id",
                                            right_on=correct_employee_ro
les.index)
correct_employees = correct_employees.merge(correct_office_addresses,
                                            how="left",
                                            left_on="employee_country",
                                            right_on="office_country")

# Task 5
correct_employees_renamed = correct_employees.drop(["employee_first_nam
e", "employee_last_name"], axis=1)
correct_header = ["id", "country", "city", "street", "street_number", "l
ast_name", "first_name",
                  "emergency_contact", "emergency_number", "emergency_re
lationship",
                  "monthly_salary", "team", "title", "office", "office_c
ountry",
                  "office_city", "office_street", "office_street_numbe
r"]
correct_employees_renamed.columns = correct_header
```

```
# Task 6
correct_column_order = ["id", "last_name", "first_name", "title", "tea
m", "monthly_salary",
                        "country", "city", "street", "street_number",
                        "emergency_contact", "emergency_number", "emerge
ncy_relationship",
                        "office", "office_country", "office_city", "offi
ce_street", "office_street_number"]
correct_employees_ordered = correct_employees_renamed[correct_column_ord
er]

def test_column_order():
    assert correct_column_order == new_column_order, \
    "It seem your `new_column_names` is incorrect.\n\
     Are you sure you used the column names given by People Ops,\n\
     in the order they were provided?\n\
     Make sure there's no typo."

def test_employees_renamed():
    assert correct_employees_ordered.equals(df_employees_ordered), \
    "It seems there's something wrong with your `df_employees_oredered`
 DataFrame.\n\
     Are you sure you renamedthe column titles and then reordered the
m?."
```

Out[94]: 2/2 tests passed

7. The last minute request

Last touches! You were ready to let People Ops know that the DataFrame was ready, but the department head just went over to your desk after lunch, asking about some last-minute requirements.

Let's polish the DataFrame before exporting the data, sending it over to People Ops, and deploying the pipeline:

- All street numbers should be integers
- The index should be the actual employee ID rather than the row number
- If the value for office is `NaN` then the employee is remote: add a column named "status", right after `monthly_salary` indicating whether the employee is "On-site" or "Remote."

In [95]:
```
# Reset the index and drop the column
df_employees_final = df_employees_ordered.set_index(df_employees_ordered
["id"]).drop(columns=["id"])

# Loop through the row values and append to status_list accordingly
status_list = []
for index, row in df_employees_final.iterrows():
    if pd.isnull(row["office"]):
        status_list.append("Remote")
    else:
        status_list.append("On-site")

# Or
status_list = ["Remote" if pd.isnull(row["office"]) else "On-site" for i
ndex, row in df_employees_final.iterrows()]

# Insert status as a new column
df_employees_final.insert(loc=5, column="status", value=status_list)

# Take a look at the first rows of the DataFrame
df_employees_final.head()
```

Out[95]:

id	last_name	first_name	title	team	monthly_salary	status	country	city
A2R5H9	Hunman	Jax	CEO	Leadership	$4500	On-site	BE	Leuven
H8K0L6	Siff	Tara	CFO	Leadership	$4500	On-site	GB	London
G4R7V0	Sagal	Gemma	Business Developer	Sales	$3000	On-site	US	New-York
M1Z7U9	Coates	Tig	Office Manager	People Operations	$2000	Remote	FR	Paris

In [96]:
```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Task 1
correct_office_addresses = pd.read_csv("datasets/office_addresses.csv")

correct_employee_addresses = pd.read_excel("datasets/employee_informatio
n.xls")

# Task 2
correct_emergency_contacts = pd.read_excel("datasets/employee_informatio
n.xls", sheet_name=1, header=None)
correct_emergency_contacts_header = ["employee_id", "last_name", "first_
name",
                                     "emergency_contact", "emergency_contact_num
ber", "relationship"]
correct_emergency_contacts.columns = correct_emergency_contacts_header

# Task 3
correct_employee_roles = pd.read_json("datasets/employee_roles.json", or
ient="index")
correct_employee_roles = correct_employee_roles.reindex(sorted(correct_e
mployee_roles.columns), axis=1)

# Task 4
correct_employees = correct_employee_addresses.merge(correct_emergency_c
ontacts,
                                                     how="left",
                                                     on="employee_id")
correct_employees = correct_employees.merge(correct_employee_roles,
                                            how="left",
                                            left_on="employee_id",
                                            right_on=correct_employee_ro
les.index)
correct_employees = correct_employees.merge(correct_office_addresses,
                                            how="left",
                                            left_on="employee_country",
                                            right_on="office_country")

# Task 5
correct_employees_renamed = correct_employees.drop(["employee_first_nam
e", "employee_last_name"], axis=1)
correct_header = ["id", "country", "city", "street", "street_number", "l
ast_name", "first_name",
                  "emergency_contact", "emergency_number", "emergency_re
lationship",
                  "monthly_salary", "team", "title", "office", "office_c
ountry",
                  "office_city", "office_street", "office_street_numbe
r"]
correct_employees_renamed.columns = correct_header
```

```
# Task 6
correct_column_order = ["id", "last_name", "first_name", "title", "tea
m", "monthly_salary",
                        "country", "city", "street", "street_number",
                        "emergency_contact", "emergency_number", "emerge
ncy_relationship",
                        "office", "office_country", "office_city", "offi
ce_street", "office_street_number"]
correct_employees_ordered = correct_employees_renamed[correct_column_ord
er]

# Task 7
correct_employees_final = correct_employees_ordered.set_index(correct_em
ployees_ordered["id"]).drop(columns=["id"])

correct_status = []
for index, row in correct_employees_final.iterrows():
    if pd.isnull(row["office"]):
        correct_status.append("Remote")
    else:
        correct_status.append("On-site")

correct_employees_final.insert(loc=5, column="status", value=correct_sta
tus)

def test_status():
    assert correct_status == status_list, \
    "It seems `status_list` is not correct.\n\
     Make sure you're looping through the rows of the `df_employees_fina
l` DataFrame.\n\
     You should append 'Remote' to `status` if the value is `NaN` and 'O
n-site' otherwise."

def test_employees_final():
    assert correct_employees_final.equals(df_employees_final), \
    "It seems your `df_employees_final` DataFrame is not correct.\n\
     Are you sure you added your `status_list` as the 5th column?"
```

Out[96]: 2/2 tests passed

8. Saving your work

Good job! You now have everything People Ops requested. The different people responsible for these various files can currently keep working on these files if they want. As long as they save it in the `datasets` folder, People Ops will have to execute this unique script to obtain just one file from the ones scattered across different teams.

You bumped into the Head of People Ops and shared a few caveats and areas of improvement. She booked a meeting with you so you can explain:

- How the current structure isn't robust to role changes: what if an existing employee takes on a new role?
- How the current structure doesn't fit best practices in terms of database schema:
  - having data all over the place like it's the case right now is a no-go
  - but gathering everything in a single table is inefficient: you have to query all information even if all you want is a phone number
  - there should be a single SQL database for employee data, with several tables that can be joined
  - views can be built on top of the database to simplify non-data practitioners access.

In any case, you still need to show up with what was requested - so let's export your DataFrame to a CSV file.

In [97]:
```
# Write to CSV
df_employees_final.to_csv("employee_data.csv")
```

In [98]:
```
%%nose

# Task 1
correct_office_addresses = pd.read_csv("datasets/office_addresses.csv")

correct_employee_addresses = pd.read_excel("datasets/employee_informatio
n.xls")

# Task 2
correct_emergency_contacts = pd.read_excel("datasets/employee_informatio
n.xls", sheet_name=1, header=None)
correct_emergency_contacts_header = ["employee_id", "last_name", "first_
name",
                                     "emergency_contact", "emergency_contact_num
ber", "relationship"]
correct_emergency_contacts.columns = correct_emergency_contacts_header

# Task 3
correct_employee_roles = pd.read_json("datasets/employee_roles.json", or
ient="index")
correct_employee_roles = correct_employee_roles.reindex(sorted(correct_e
mployee_roles.columns), axis=1)

# Task 4
correct_employees = correct_employee_addresses.merge(correct_emergency_c
ontacts,
                                                     how="left",
                                                     on="employee_id")
correct_employees = correct_employees.merge(correct_employee_roles,
                                            how="left",
                                            left_on="employee_id",
                                            right_on=correct_employee_ro
les.index)
correct_employees = correct_employees.merge(correct_office_addresses,
                                            how="left",
                                            left_on="employee_country",
                                            right_on="office_country")

# Task 5
correct_employees_renamed = correct_employees.drop(["employee_first_nam
e", "employee_last_name"], axis=1)
correct_header = ["id", "country", "city", "street", "street_number", "l
ast_name", "first_name",
                  "emergency_contact", "emergency_number", "emergency_re
lationship",
                  "monthly_salary", "team", "title", "office", "office_c
ountry",
                  "office_city", "office_street", "office_street_numbe
r"]
correct_employees_renamed.columns = correct_header

# Task 6
correct_column_order = ["id", "last_name", "first_name", "title", "tea
m", "monthly_salary",
                        "country", "city", "street", "street_number",
                        "emergency_contact", "emergency_number", "emerge
```

```
ncy_relationship",
                          "office", "office_country", "office_city", "offi
ce_street", "office_street_number"]
correct_employees_ordered = correct_employees_renamed[correct_column_ord
er]

# Task 7
correct_employees_final = correct_employees_ordered.set_index(correct_em
ployees_ordered["id"]).drop(columns=["id"])

correct_status = []
for index, row in correct_employees_final.iterrows():
    if pd.isnull(row["office"]):
        correct_status.append("Remote")
    else:
        correct_status.append("On-site")

correct_employees_final.insert(loc=5, column="status", value=correct_sta
tus)

df_employees_final.to_csv("final.csv")

correct_csv = pd.read_csv("final.csv")
student_csv = pd.read_csv("employee_data.csv")

def test_csv():
    assert correct_csv.equals(student_csv), \
    "It seems your CSV file is not correct.\n\
     Make sure your `df_employees_final` DataFrame is correct,\n\
     and that it's the one you're exporting to CSV."
```

Out[98]: 1/1 tests passed