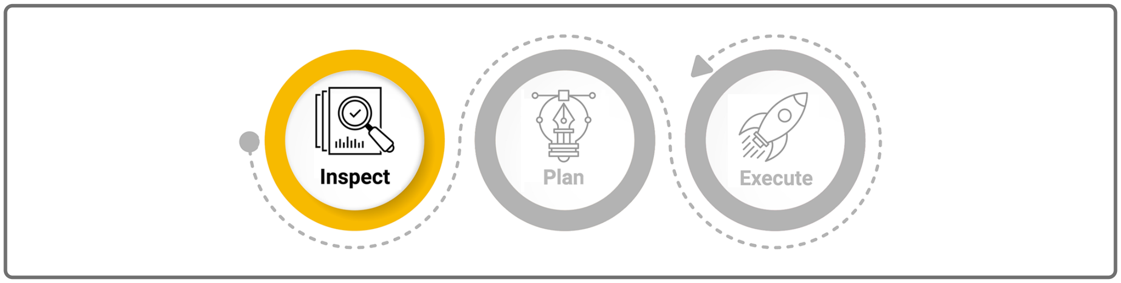
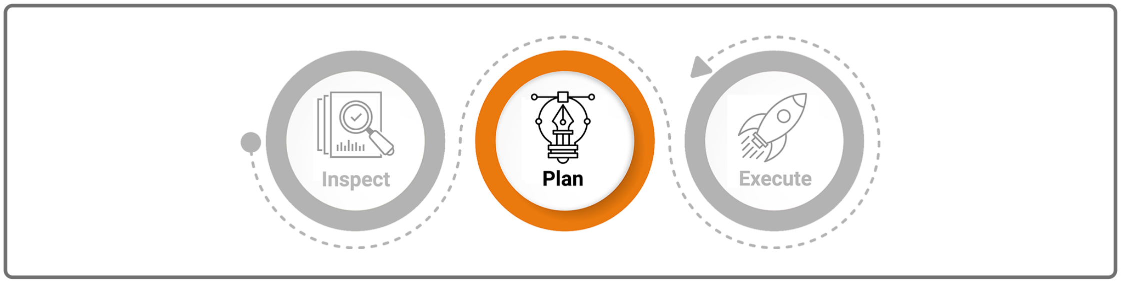
8.3.6 Create a Function to Clean the Data, Part 2

You're honest with Britta: sometimes, it takes a long time to clean the data. But you're up to the challenge! Now you'll dig into columns with oh-so-slightly different names.



There are quite a few columns with slightly different names but the same data, such as "Directed by" and "Director."



We need to consolidate columns with the same data into one column. We can use the `pop()` method to change the name of a dictionary key, because `pop()` returns the value from the removed key-value pair. We have to check if the key exists in a given movie record, so it will be helpful to make a small function inside `clean_movie()`.

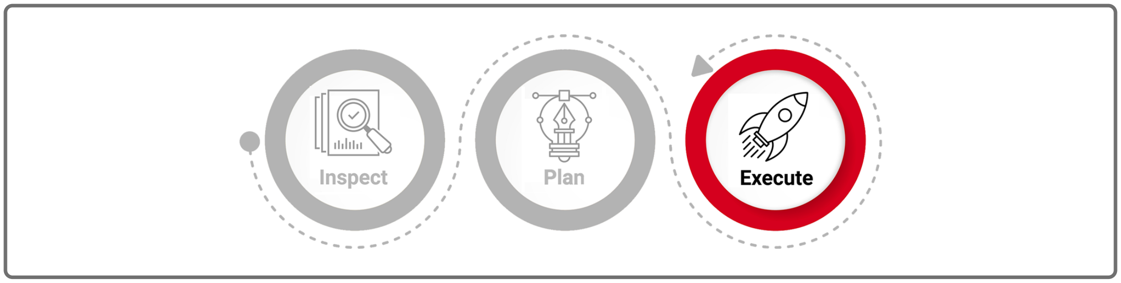
NOTE

> It's perfectly fine to define a function within another function. In fact, it's often preferable. Functions that are defined within another function live within the scope of the first equation. This means that the inner function can only be called inside the outer function. Outside the original function, it's impossible to call the inner function.

Remember, if Britta needs to go through your notebook to understand your ETL process, it'll be much easier for her to understand if you name your functions as verbs. Also, it's better to be explicit than implicit and write out full words, so we'll call our new function `change_column_name`.

NOTE

> One of the benefits of using a good, dedicated code editor is that you can autocomplete names, usually by pressing the Tab key when you've partially written a variable, function, or keyword. Programmers used to use short, confusing names so they wouldn't have to type out repetitive code so much, but with autocomplete, we can write more descriptive names for functions and variables.

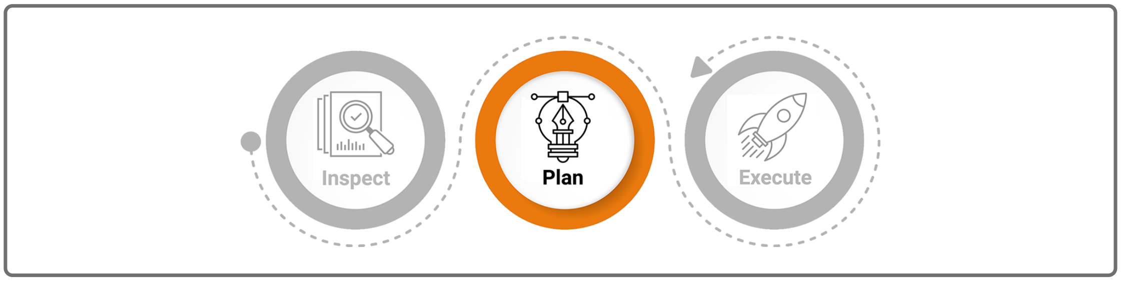


Our new function should look like the following. Remember that this new function is enclosed within the clean_movie function that we created earlier:

```
    def change_column_name(old_name, new_name):
        if old_name in movie:
            movie[new_name] = movie.pop(old_name)
```

To change every instance where the key is "Directed by" to the new key "Director," write the following inside `clean_movie()`:

```
change_column_name('Directed by', 'Director')
```

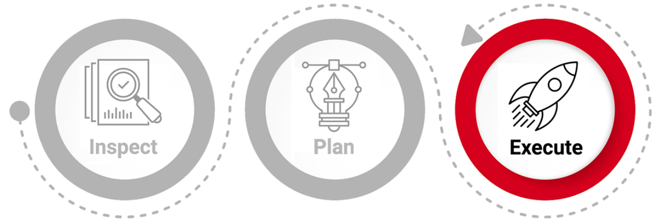


There's no easy way around the next step: we have to go through each column name and decide if there's a better name for it. If you're not sure what the column is referring to, do some research—don't guess. Use your Google-fu to gain domain knowledge.

IMPORTANT

> Domain knowledge is specific expertise in the data professional's industry or field, outside of statistics and coding. For example, a data scientist working in healthcare might need specific clinical knowledge to perform certain analyses accurately.

The most important thing to remember when consolidating the comments is to be consistent. For example, will we use "Composer" or "Composed by"? "Editor" or "Edited by"? Will our columns be capitalized? How will we handle plurals?



Below is an example of how to consolidate columns. Yours might appear slightly different.

```
change_column_name('Adaptation by', 'Writer(s)')
change_column_name('Country of origin', 'Country')
change_column_name('Directed by', 'Director')
change_column_name('Distributed by', 'Distributor')
change_column_name('Edited by', 'Editor(s)')
change_column_name('Length', 'Running time')
change_column_name('Original release', 'Release date')
change_column_name('Music by', 'Composer(s)')
change_column_name('Produced by', 'Producer(s)')
change_column_name('Producer', 'Producer(s)')
change_column_name('Productioncompanies ', 'Production company(s)')
change_column_name('Productioncompany ', 'Production company(s)')
change_column_name('Released', 'Release Date')
change_column_name('Release Date', 'Release date')
change_column_name('Screen story by', 'Writer(s)')
change_column_name('Screenplay by', 'Writer(s)')
change_column_name('Story by', 'Writer(s)')
```

```
    change_column_name('Theme music composer', 'Composer(s)')
    change_column_name('Written by', 'Writer(s)')
```

The function `clean_movie()` is starting to look a little complicated, so we should add some commenting to make it easier to understand. The whole function should look like this:

```
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    alt_titles = {}
    # combine alternate titles into one list
    for key in ['Also known as','Arabic','Cantonese','Chinese','French',
                'Hangul','Hebrew','Hepburn','Japanese','Literally',
                'Mandarin','McCune-Reischauer','Original title','Polish',
                'Revised Romanization','Romanized','Russian',
                'Simplified','Traditional','Yiddish']:
        if key in movie:
            alt_titles[key] = movie[key]
            movie.pop(key)
    if len(alt_titles) > 0:
        movie['alt_titles'] = alt_titles

    # merge column names
    def change_column_name(old_name, new_name):
        if old_name in movie:
            movie[new_name] = movie.pop(old_name)
    change_column_name('Adaptation by', 'Writer(s)')
    change_column_name('Country of origin', 'Country')
    change_column_name('Directed by', 'Director')
    change_column_name('Distributed by', 'Distributor')
    change_column_name('Edited by', 'Editor(s)')
    change_column_name('Length', 'Running time')
    change_column_name('Original release', 'Release date')
    change_column_name('Music by', 'Composer(s)')
    change_column_name('Produced by', 'Producer(s)')
    change_column_name('Producer', 'Producer(s)')
    change_column_name('Productioncompanies ', 'Production company(s)')
    change_column_name('Productioncompany ', 'Production company(s)')
    change_column_name('Released', 'Release Date')
    change_column_name('Release Date', 'Release date')
    change_column_name('Screen story by', 'Writer(s)')
    change_column_name('Screenplay by', 'Writer(s)')
    change_column_name('Story by', 'Writer(s)')
    change_column_name('Theme music composer', 'Composer(s)')
    change_column_name('Written by', 'Writer(s)')

    return movie
```

Now we can rerun our list comprehension to clean `wiki_movies` and recreate `wiki_movies_df`.

```
clean_movies = [clean_movie(movie) for movie in wiki_movies]
wiki_movies_df = pd.DataFrame(clean_movies)
sorted(wiki_movies_df.columns.tolist())
```

NOTE

> When using notebooks like Jupyter, it's easy to lose track of the order in which the code was run if you edit functions in previous cells and jump around between different cells. It's best to keep the flow of the notebook linear, if possible.
>
> To track why certain decisions were made, show the evolution of the function through multiple cells.
>
> Clear documentation is one of the best ways to set yourself apart from other programmers on the job. If your programming portfolio contains well-documented code and notebooks, it will also set you apart in interviews.