Data Wrangling Exercise

# Project context

* In this project the ‘tabular’ module in python to extract tables from the Tata steel section tables PDF(file: “ST.pdf”). The extracted tables are then refined and saved to an excel workbook.
* A how-to guide on how to use the script can be found in page 5
* This report describes the main transformative actions performed on the raw extracted table
* The data in question is standard structural member dimensions and properties (e.g. tables of different beams with properties such as cross-section, Mass per metre, etc)
* TATA steel is a manufacturer, processor, and distributer of steel products amongst other things

# Python Modules

* Pandas
* tabula
* copy

# Skills Showcased:

* Ability to implement the Object-Oriented Programming paradigm
* Using tabular module to extract PDF into pandas df
* transformed pandas dataframe of raw pdf scrapings into organised tables using list indexing, dataframe indexing, loops etc
* saved the transformed pandas df into an excel workbook

# Project Outcome

In undergoing this project, the script created can extract and clean the tables in pages 16–25 of the source pdf such that it is analysis-ready(see file: ” pdf\_scrape\_class.py”). It can also load the refined data into an Excel workbook. it scrapes the tabular data from basketball reference and stores in either an EXCEL workbook or a table in the local SQL server. A flowchart has been made explaining how to use the tool (see page 3)

Note:

* This report is meant to showcase Data Engineering skills hence the following pages will highlight the refinement processes conducted using python.
* This project is part of a larger Structural Engineering project which will use the extracted data to automate the design process of

# Data Transformation process

## Step 1: Raw PDF extraction into dataframe: Identifying the inaccuracies

* The ‘tabular’ module does a decent job at detecting the fields of a table, but it is far from perfect as can be seen in figure 2.
* This step of the transformation process is about spotting inaccuracies in the raw data extracted as well as things to add.

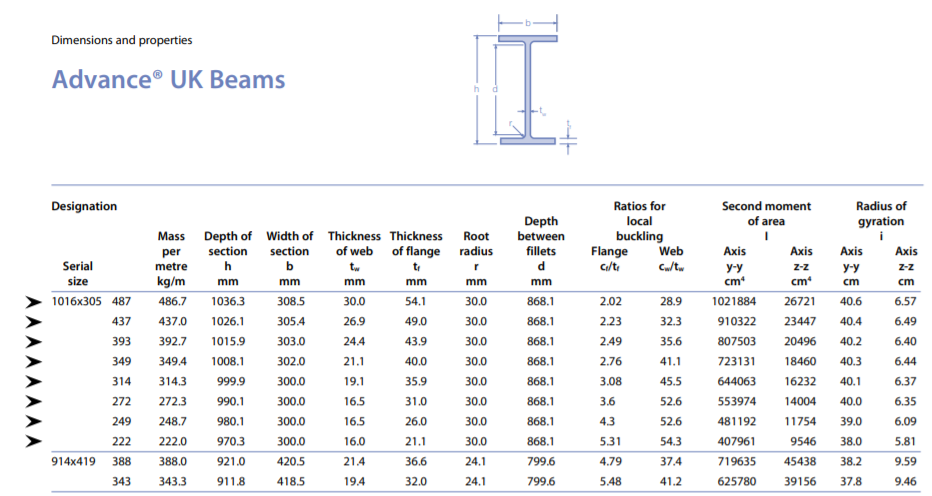
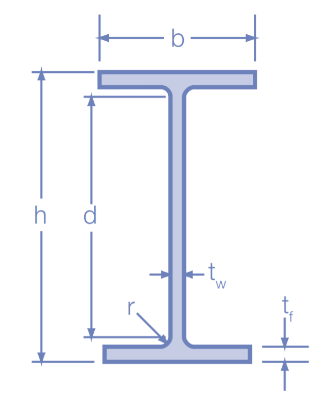
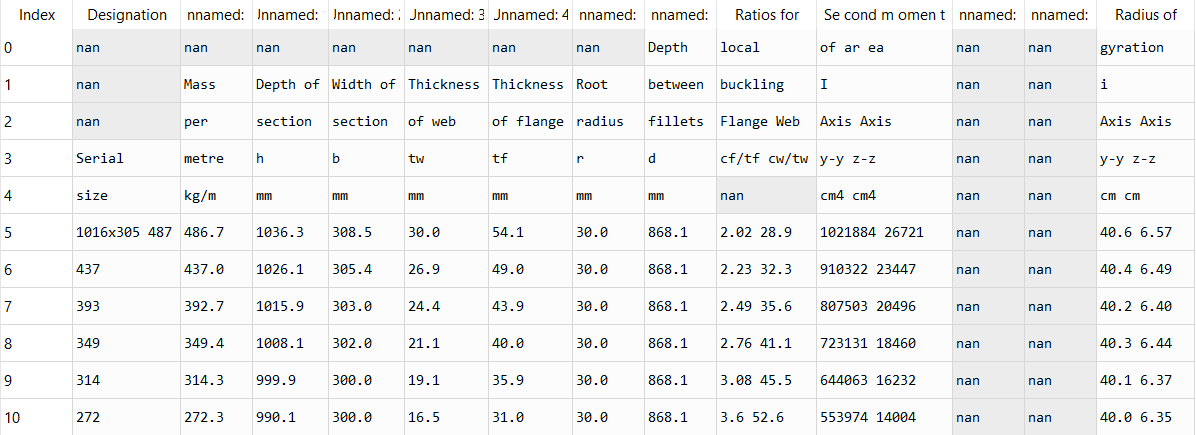


Figure 1: A cut of the Advanced beam table from page 16 of the source PDF



**Addition**: Re-structure column such that each beam has a unique identifier in the format: No.1xNo.2xNo.3 (e.g. for the first two beams: 1016x305x487, 1016x305x437)

Figure 2: The Dataframe produced by the 'tabula' module for page 16 of the source PDF



**Inaccuracy 3**:’ tabula’ has merged two columns into one

These columns need to be split and their headings need to be restructured

**Inaccuracy 2**: ’tabula’ has added empty columns

These columns need to be removed

**Inaccuracy 1**: segmented the column headings into rows

These column headings need to be appropriately arranged

## Step 2: Correcting Inaccuracies and adding serial size columns

* The below figure detail some of the corrective actions (fig 3 and 5). It also includes the transformation of the ‘Designation Serial size’ column (fig 4). Both actions have been programmed in Python (file: “pdf\_scrape\_class.py”)

Figure 3: Creating the correct columns headings for page 16 of the source pdf (see figure 1 and 2 to compare)

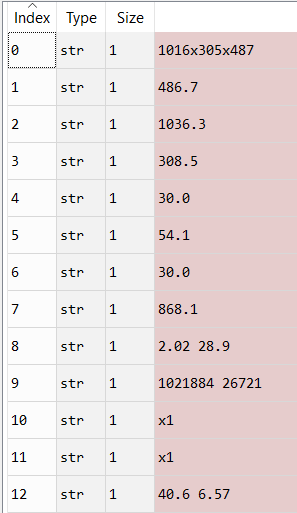
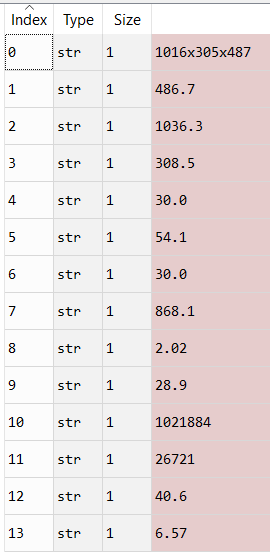
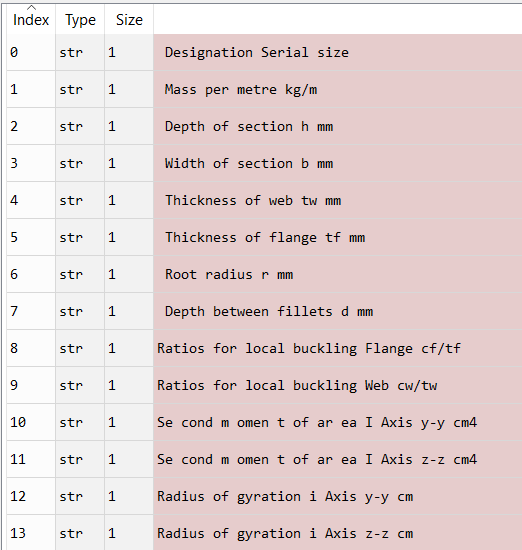


Figure 4: Splitting merged column for row 1 of table in page 16 of source pdf (see figure 1 and 2 to compare)

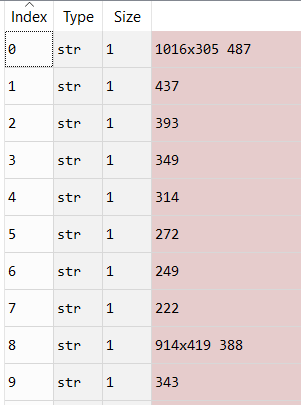
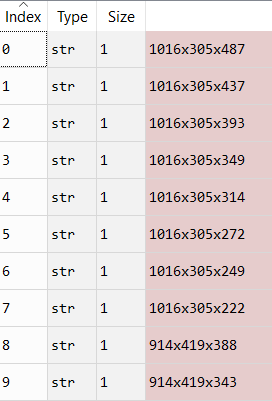
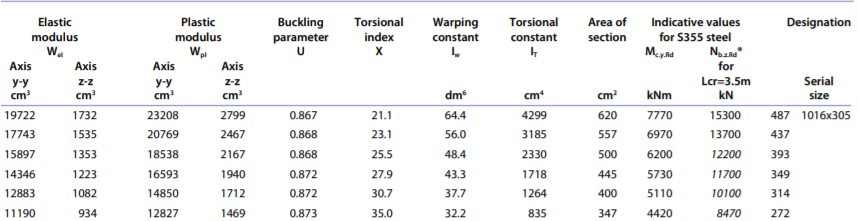
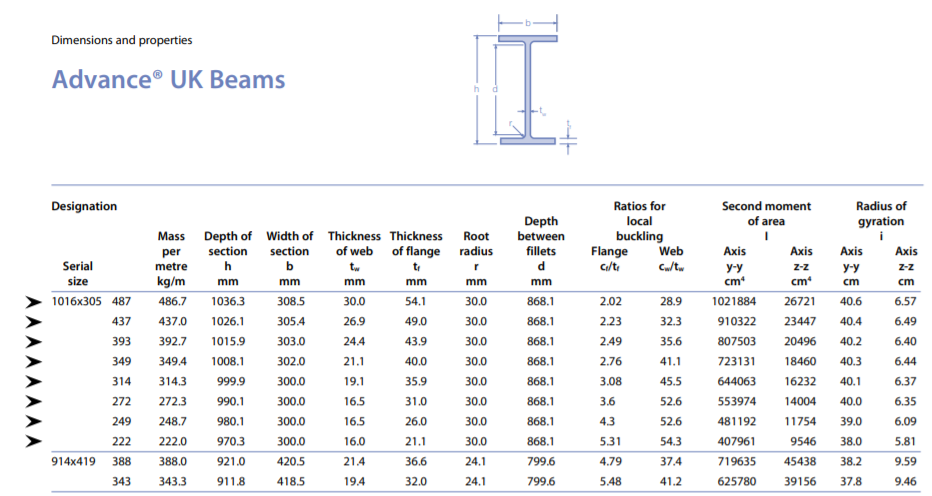


Figure 5: Changing fields of ‘Designation Serial Size’ column to distinct row identifiers for first 10 rows of table in page 16

## Step 3: Joining Tables (where applicable)

* Due to the sheer number of structural member properties, some tables require two pages (see figure 6 for clarification)
* As a result, some tables need to be joined by the ‘Designation Serial size’ (akin to a SQL join)



Page 16

Page 17

Figure 6: Slices from the tables in pages 16 and 17 of the source PDF. Note: the tables join together to display the full properties of a given beam.

* The script produces the following dataframe for pages 16 and 17 after all the previous steps

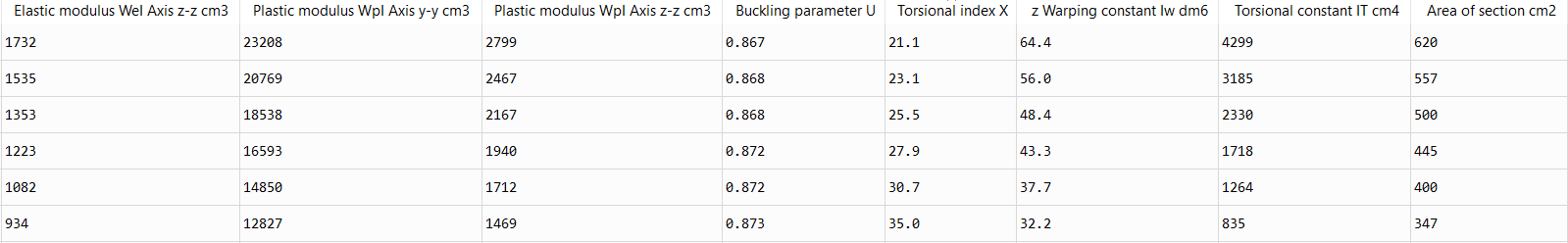
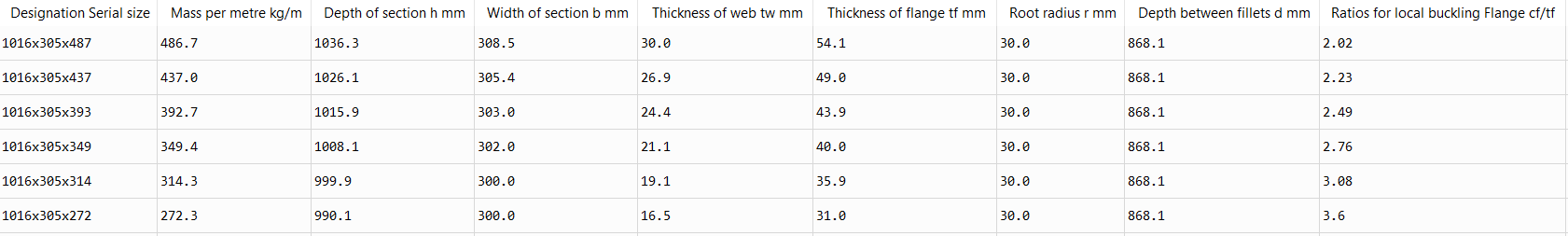


Figure 7: A slice of the dataframe produced by the script after all the cleaning processes have been conducted for page 16 and 17 of the source PDF. Note: this figure does not show all the columns in the dataframe dues to the sheer number of columns

# How to use the PDF scraping script

OPEN FILE: “pdf\_scrape\_class.py

Scroll down to Line 242: #Testing#

Is the table of you member properties separated into two pages?

yes

No

Write relevant pages in a list object:

i.e. [Page1, Page 2]

e.g. < page\_list = [16, 17]>

This extracts tables in pages 16 and 17 and concatenates them

Write relevant pages in a list object:

i.e. [Page1]

e.g. < page\_list = [16]>

This extracts tables from pages 16

Specify the following:

1. Name of Excel file:

* <Excel\_file = 'filename.xlsx'> #

1. Header inclusion

* If you want to include the columns headings :

<header = 'No'>

* If you don’t want to include column heading:

<header = 'No'>

**Run Script**

Would you like the data to be stored in an excel file?

No

Yes

Write ‘No’ as a string:

e.g. <write = ‘No’>

This extracts data for 2019

Write ‘Yes’ as a string:

e.g. <write = ‘Yes’>

This extracts data for 2019