# Merging the ATO Dataset with the Postcode Data

In this group exercise, we will merge the ATO dataset (28 columns) with the Postcode dataset (150 columns) to get a richer dataset with an increased number of columns.

Note

The Australian Taxation Office (ATO) dataset can be found in the Fenago GitHub repository: <https://raw.githubusercontent.com/fenago/data_wrangling/main/taxstats2015.csv>.

The Postcode dataset can be found here: <https://github.com/fenago/data_wrangling/blob/main/taxstats2016individual06taxablestatusstateterritorypostcodetaxableincome.xlsx?raw=true>.

The sources of the dataset are as follows:

The **Australian Taxation Office** (**ATO**): https://data.gov.au/dataset/ds-dga-5c99cfed-254d-40a6-af1c-47412b7de6fe/details.

The Postcode dataset: <https://github.com/fenago/data_wrangling/blob/main/taxstats2016individual06taxablestatusstateterritorypostcodetaxableincome.xlsx?raw=true>.

The following steps will help you complete the exercise:

1. Open up a new Colab notebook.
2. Now, begin with the import of the pandas package:

import pandas as pd

1. Assign the link to the ATO dataset to a variable called file\_url:
2. file\_url = “<https://raw.githubusercontent.com/fenago/data_wrangling/main/taxstats2015.csv>”
3. Using the .read\_csv() method from the pandas package, load the dataset into a new DataFrame called df:

df = pd.read\_csv(file\_url)

1. Display the dimensions of this DataFrame using the .shape attribute:

df.shape

You should get the following output:

(2473, 28)

The ATO dataset contains 2471 rows and 28 columns.

1. Display the first five rows of the ATO DataFrame using the .head() method:

df.head()

You should get the following output:

Figure 12.10: First five rows of the ATO dataset


Figure 12.10: First five rows of the ATO dataset

Both DataFrames have a column called Postcode containing postcodes, so we will use it to merge them together.

Note

Postcode is the name used in Australia for zip code. It is an identifier for postal areas.

We are interested in learning more about each of these postcodes. Let's make sure they are all unique in this dataset.

1. Display the number of unique values for the Postcode variable using the .nunique() method:

df['Postcode'].nunique()

You should get the following output:

2473

There are 2473 unique values in this column and the DataFrame has 2473 rows, so we are sure the Postcode variable contains only unique values.

1. Now, assign the link to the second Postcode dataset to a variable called postcode\_df:
2. postcode\_url = “<https://github.com/fenago/data_wrangling/blob/main/taxstats2016individual06taxablestatusstateterritorypostcodetaxableincome.xlsx?raw=true>”
3. Load the second Postcode dataset into a new DataFrame called postcode\_df using the .read\_excel() method.

We will only load the *Individuals Table 6B*sheet as this is where the data is located so we need to provide this name to the sheet\_name parameter. Also, the header row (containing the name of the variables) in this spreadsheet is located on the third row so we need to specify it to the header parameter.

Note

Don't forget the index starts with 0 in Python.

Have a look at the following code snippet:

postcode\_df = pd.read\_excel(postcode\_url, \

                            sheet\_name='Individuals Table 6B', \

                            header=2)

1. Print the dimensions of postcode\_df using the .shape attribute:

postcode\_df.shape

You should get the following output:

(2567, 150)

This DataFrame contains 2567 rows for 150 columns. By merging it with the ATO dataset, we will get additional information for each postcode.

1. Print the first five rows of postcode\_df using the .head() method:

postcode\_df.head()

You should get the following output:

Figure 12.11: First five rows of the Postcode dataset


Figure 12.11: First five rows of the Postcode dataset

We can see that the second column contains the postcode value, and this is the one we will use to merge on with the ATO dataset. Let's check if they are unique.

1. Print the number of unique values in this column using the .nunique() method as shown in the following code snippet:

postcode\_df['Postcode'].nunique()

You should get the following output:

2567

There are 2567 unique values, and this corresponds exactly to the number of rows of this DataFrame, so we're absolutely sure this column contains unique values. This also means that after merging the two tables, there will be only one-to-one matches. We won't have a case where we get multiple rows from one of the datasets matching with only one row of the other one. For instance, postcode 2029 from the ATO DataFrame will have exactly one match in the second Postcode DataFrame.

1. Perform a left join on the two DataFrames using the .merge() method and save the results into a new DataFrame called merged\_df. Specify the how='left' and on='Postcode' parameters:
2. merged\_df = pd.merge(df, postcode\_df, \

                     how='left', on='Postcode')

1. Print the dimensions of the new merged DataFrame using the .shape attribute:

merged\_df.shape

You should get the following output:

(2473, 177)

We got exactly 2473 rows after merging, which is what we expect as we used a left join and there was a one-to-one match on the Postcode column from both original DataFrames. Also, we now have 177 columns, which is the objective of this exercise. But before concluding it, we want to see whether there are any postcodes that didn't match between the two datasets. To do so, we will be looking at one column from the right-hand side DataFrame (the Postcode dataset) and see if there are any missing values.

1. Print the total number of missing values from the 'State/Territory1' column by combining the .isna() and .sum() methods:

merged\_df['State/ Territory1'].isna().sum()

You should get the following output:

4

There are four postcodes from the ATO dataset that didn't match the Postcode code.

Let's see which ones they are.

1. Print the missing postcodes using the .iloc() method, as shown in the following code snippet:
2. merged\_df.loc[merged\_df['State/ Territory1'].isna(), \

              'Postcode']

You should get the following output:

Figure 12.12: List of unmatched postcodes


Figure 12.12: List of unmatched postcodes

The missing postcodes from the Postcode dataset are 3010, 4462, 6068, and 6758. In a real project, you would have to get in touch with your stakeholders or the data team to see if you are able to get this data.

We have successfully merged the two datasets of interest and have expanded the number of features from 28 to 177. We now have a much richer dataset and will be able to perform a more detailed analysis of it.