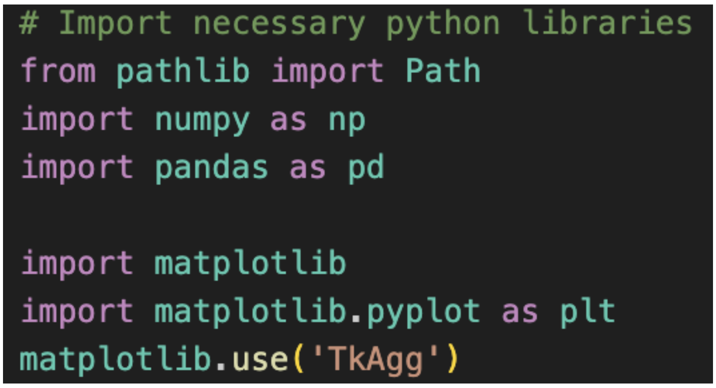
1. Data preparation and understanding

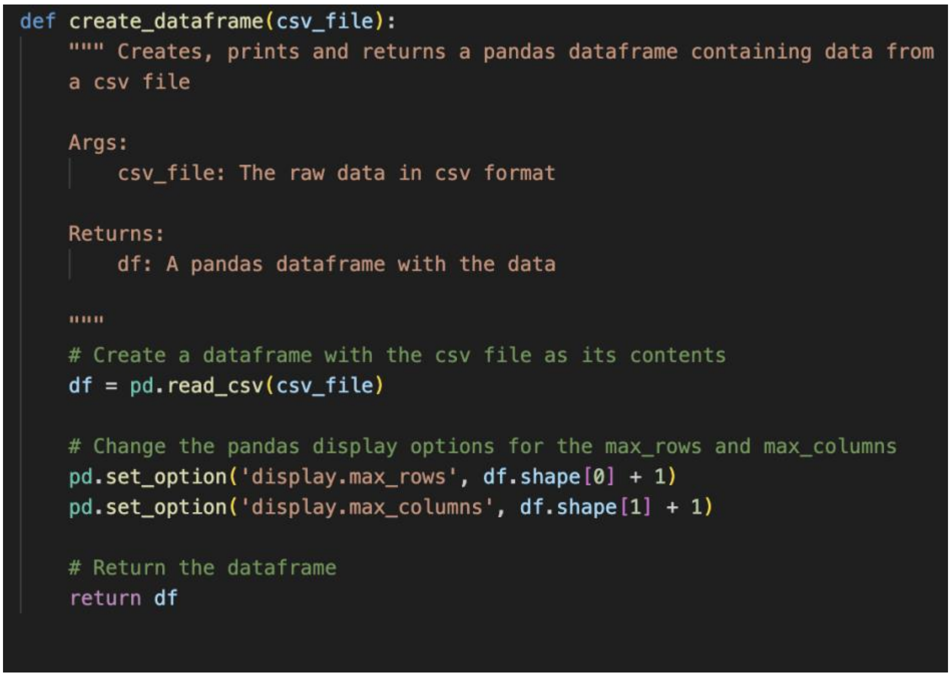
1.4. Explanation of code preparation & understanding

First of all, we need to activate .venv and pip install all the libraries listed in the requirements.txt. Then we can import libraries which will be used such as numpy and pandas. The pathlib module for Python offers a simpler way to work with the filesystem. NumPy is a Python library used for working with arrays. Pandas is mostly used to analyse data and manipulate tabular data in DataFrames. Matplotlib is used for constructing static, animated, and interactive Python visualizations.



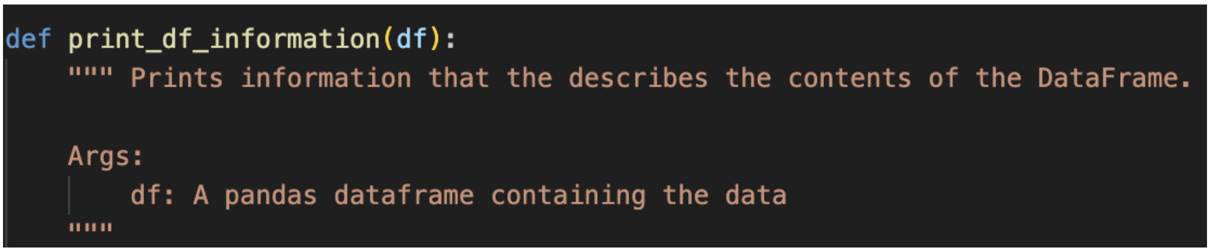
```python
# Import necessary python libraries
from pathlib import Path
import numpy as np
import pandas as pd

import matplotlib
import matplotlib.pyplot as plt
matplotlib.use('TkAgg')
```

Then I define a function which creates, prints and returns a pandas Dataframe containing data from a csv file. In the function ‘create_dataframe(csv_file)’, we can change the pandas display options for the max rows and columns.



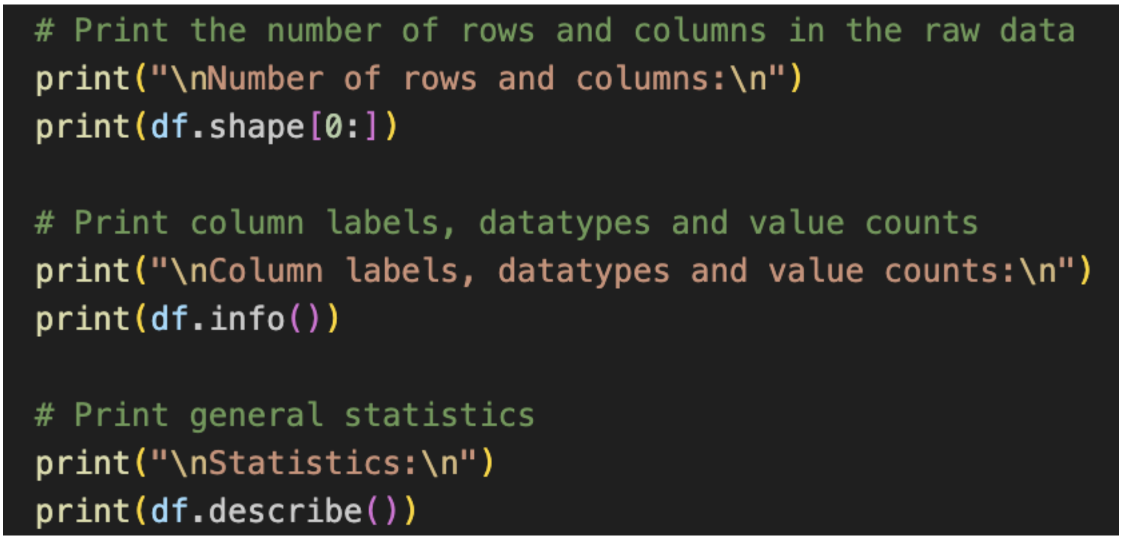
```python
def create_dataframe(csv_file):
    """ Creates, prints and returns a pandas dataframe containing data from
    a csv file

    Args:
        csv_file: The raw data in csv format

    Returns:
        df: A pandas dataframe with the data

    """
    # Create a dataframe with the csv file as its contents
    df = pd.read_csv(csv_file)

    # Change the pandas display options for the max_rows and max_columns
    pd.set_option('display.max_rows', df.shape[0] + 1)
    pd.set_option('display.max_columns', df.shape[1] + 1)

    # Return the dataframe
    return df
```

Before we prepare the data, we need to know what is the raw data. The next step is to define a function to print information which describes the contents of the raw Dataframe.



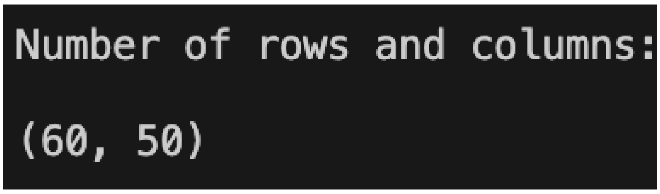
```
def print_df_information(df):
    """ Prints information that the describes the contents of the DataFrame.

    Args:
        df: A pandas dataframe containing the data
    """
```

In the function 'print_df_information(df)', we can print out the number of rows and columns of the raw data. We can also print columns labels, data types and value counts by the raw data. After that, we can also print the general statistics to help us understand the data more.



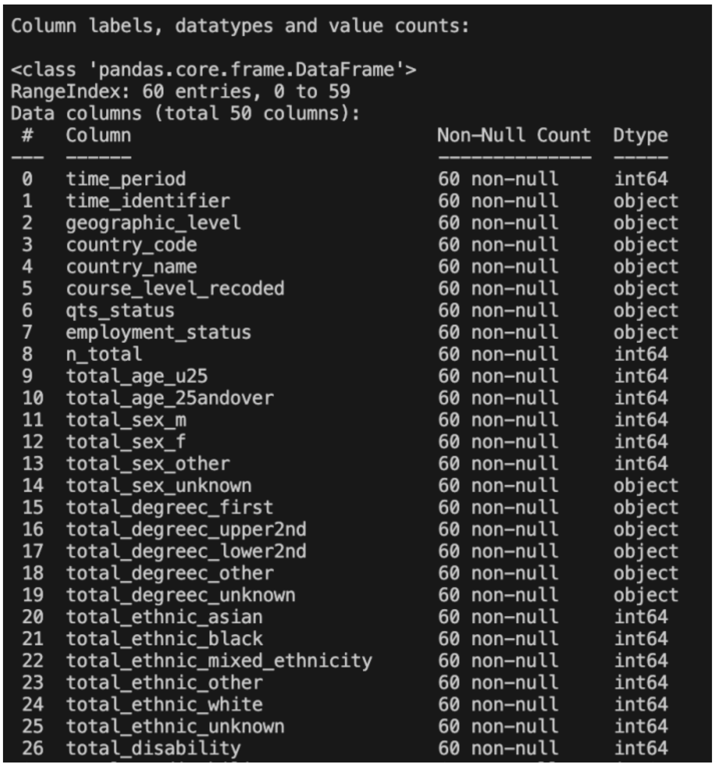
```
# Print the number of rows and columns in the raw data
print("\nNumber of rows and columns:\n")
print(df.shape[0:])

# Print column labels, datatypes and value counts
print("\nColumn labels, datatypes and value counts:\n")
print(df.info())

# Print general statistics
print("\nStatistics:\n")
print(df.describe())
```

In the terminal when we run the code in the final, we can clearly see there are 60 rows and 50 columns in the raw data.



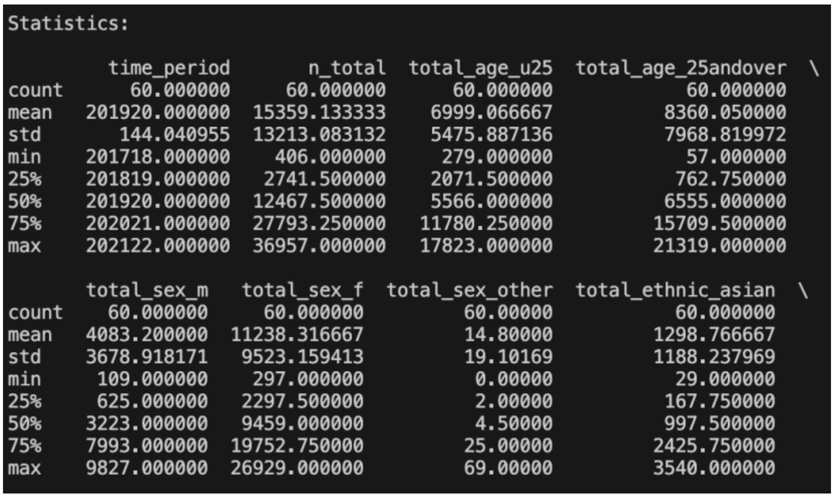
```
Number of rows and columns:

(60, 50)
```

We can also know the columns labels for each column and recognise whether the data type is int64 (A 64-bit signed integer) or object. The non-null count indicates how many rows contain non-null values for each column. As the data is 60 × 50, so the diagram below doesn’t show every column.



```
Column labels, datatypes and value counts:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 50 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   time_period                     60 non-null     int64
 1   time_identifier                 60 non-null     object
 2   geographic_level                60 non-null     object
 3   country_code                    60 non-null     object
 4   country_name                    60 non-null     object
 5   course_level_recoded            60 non-null     object
 6   qts_status                      60 non-null     object
 7   employment_status               60 non-null     object
 8   n_total                         60 non-null     int64
 9   total_age_u25                   60 non-null     int64
 10  total_age_25andover             60 non-null     int64
 11  total_sex_m                     60 non-null     int64
 12  total_sex_f                     60 non-null     int64
 13  total_sex_other                 60 non-null     int64
 14  total_sex_unknown               60 non-null     object
 15  total_degreec_first             60 non-null     object
 16  total_degreec_upper2nd          60 non-null     object
 17  total_degreec_lower2nd          60 non-null     object
 18  total_degreec_other             60 non-null     object
 19  total_degreec_unknown           60 non-null     object
 20  total_ethnic_asian              60 non-null     int64
 21  total_ethnic_black              60 non-null     int64
 22  total_ethnic_mixed_ethnicity    60 non-null     int64
 23  total_ethnic_other              60 non-null     int64
 24  total_ethnic_white              60 non-null     int64
 25  total_ethnic_unknown            60 non-null     int64
 26  total_disability                60 non-null     int64
```

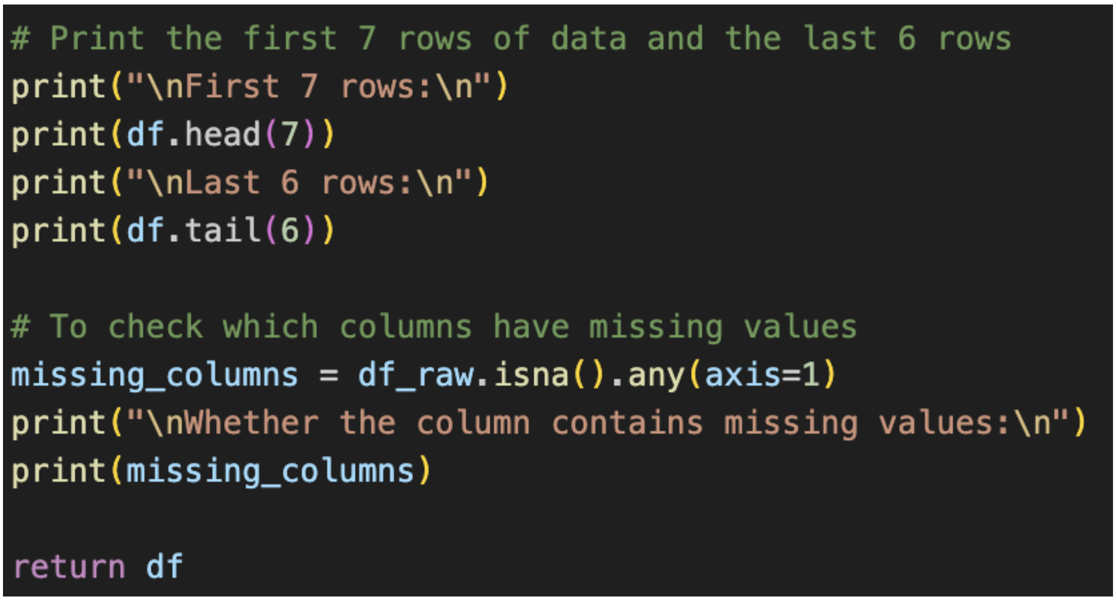
The function returns the raw data’s statistics such as count, mean, standard deviation, minimum, and maximum values for each column. It helps us understands the data in a mathematic way.



```
Statistics:

         time_period       n_total  total_age_u25  total_age_25andover  \
count      60.000000     60.000000      60.000000            60.000000
mean   201920.000000  15359.133333    6999.066667          8360.050000
std       144.040955  13213.083132    5475.887136          7968.819972
min    201718.000000    406.000000     279.000000            57.000000
25%    201819.000000   2741.500000    2071.500000           762.750000
50%    201920.000000  12467.500000    5566.000000          6555.000000
75%    202021.000000  27793.250000   11780.250000         15709.500000
max    202122.000000  36957.000000   17823.000000         21319.000000

       total_sex_m   total_sex_f  total_sex_other  total_ethnic_asian  \
count    60.000000     60.000000         60.00000           60.000000
mean   4083.200000  11238.316667         14.80000         1298.766667
std    3678.918171   9523.159413         19.10169         1188.237969
min     109.000000    297.000000          0.00000           29.000000
25%     625.000000   2297.500000          2.00000          167.750000
50%    3223.000000   9459.000000          4.50000          997.500000
75%    7993.000000  19752.750000         25.00000         2425.750000
max    9827.000000  26929.000000         69.00000         3540.000000
```

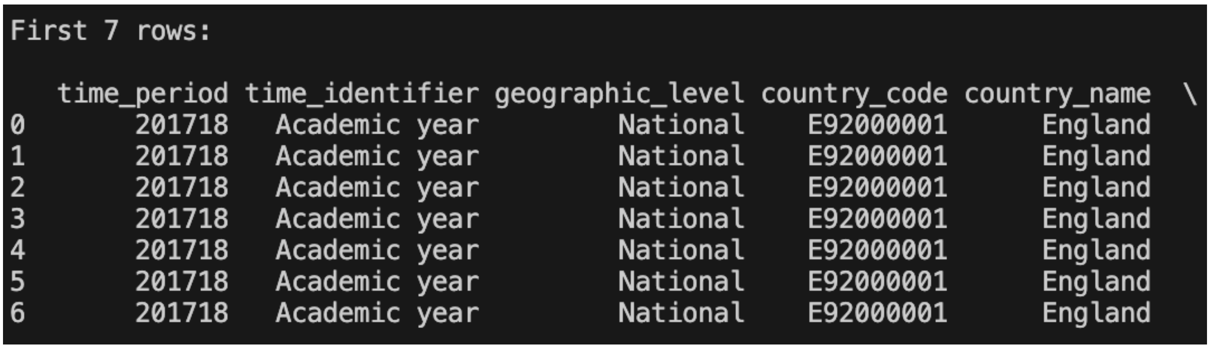
In the same function we defined, I also wanted to print the first 7 rows of data and the last 6 rows, and then check which columns have missing values by

‘.isna()’. That’s because if we observe the data directly, we can understand more about it. For a data analyst, missing data might be problematic as most statistical processes need a value for every variable. We need to find the location of missing values and then decide how to cope with them.

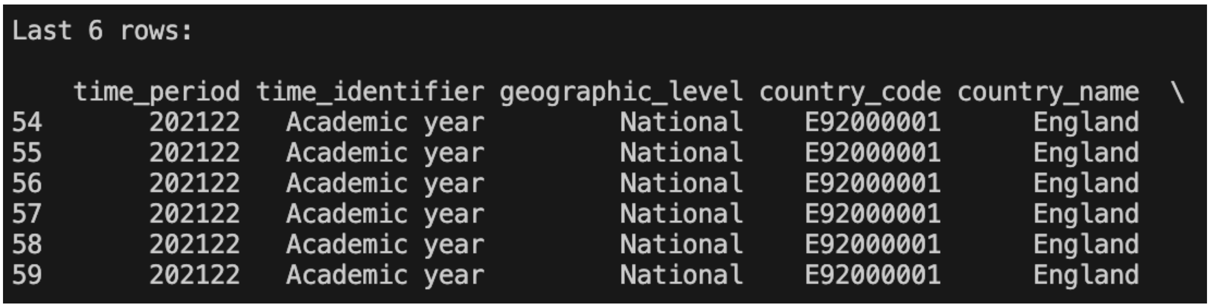


```python
# Print the first 7 rows of data and the last 6 rows
print("\nFirst 7 rows:\n")
print(df.head(7))
print("\nLast 6 rows:\n")
print(df.tail(6))

# To check which columns have missing values
missing_columns = df_raw.isna().any(axis=1)
print("\nWhether the column contains missing values:\n")
print(missing_columns)

return df
```

When we run the function, the first 7 rows and last 6 rows are shown below. It’s not the full results as there are 50 columns. We can clearly know the time period, geographic level and etc…

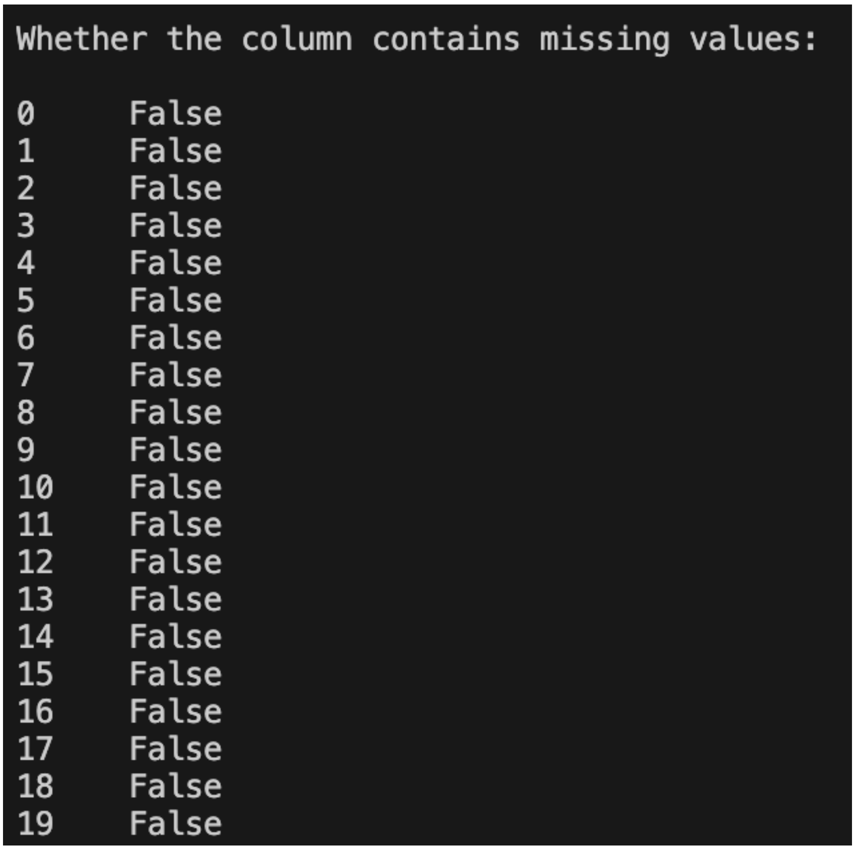


```
First 7 rows:

   time_period time_identifier geographic_level country_code country_name  \
0       201718   Academic year         National    E92000001      England
1       201718   Academic year         National    E92000001      England
2       201718   Academic year         National    E92000001      England
3       201718   Academic year         National    E92000001      England
4       201718   Academic year         National    E92000001      England
5       201718   Academic year         National    E92000001      England
6       201718   Academic year         National    E92000001      England
```



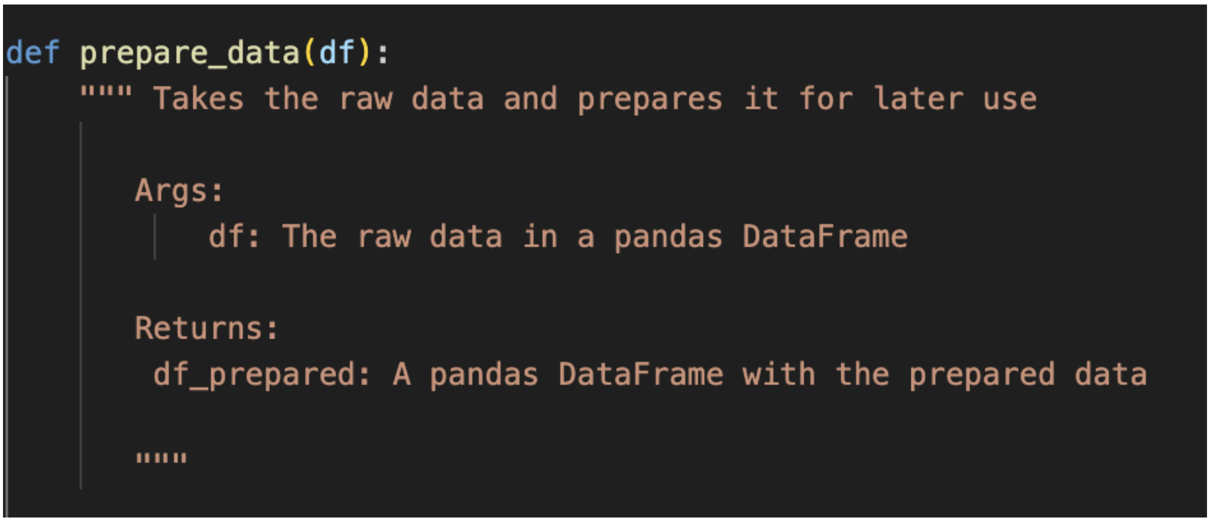
```
Last 6 rows:

    time_period time_identifier geographic_level country_code country_name  \
54       202122   Academic year         National    E92000001      England
55       202122   Academic year         National    E92000001      England
56       202122   Academic year         National    E92000001      England
57       202122   Academic year         National    E92000001      England
58       202122   Academic year         National    E92000001      England
59       202122   Academic year         National    E92000001      England
```

We can clearly know there is no columns containing missing values which is shown by the results.



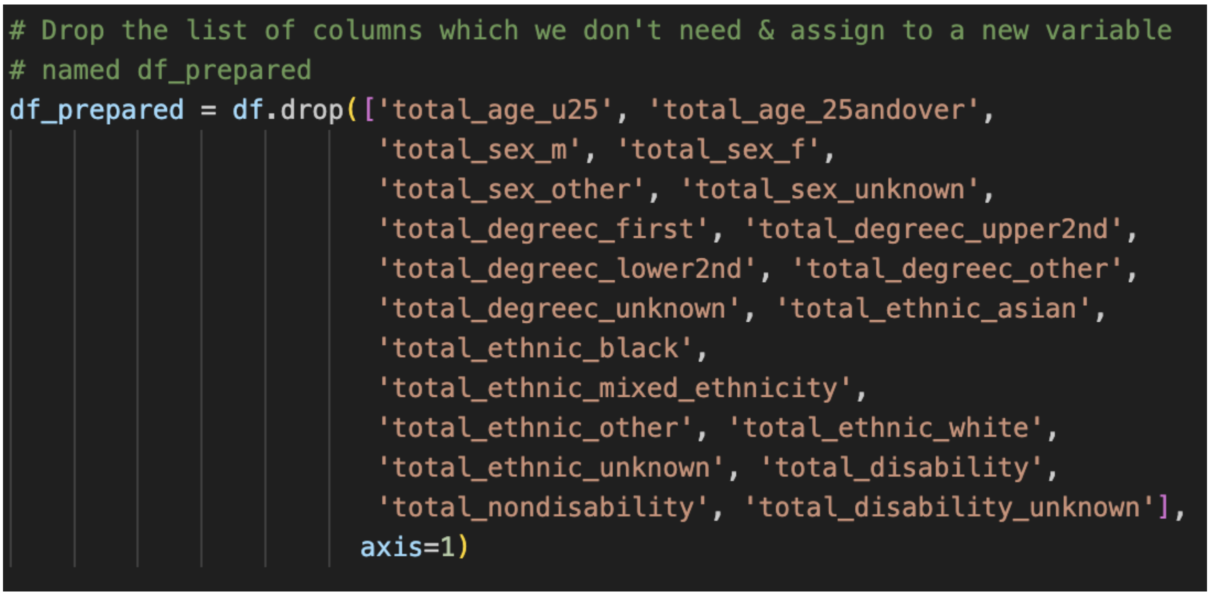
```
Whether the column contains missing values:

0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
```

When we understand what's included in the dataframe, we can define a function 'prepare_data(df)' to prepare and create a new dataframe.



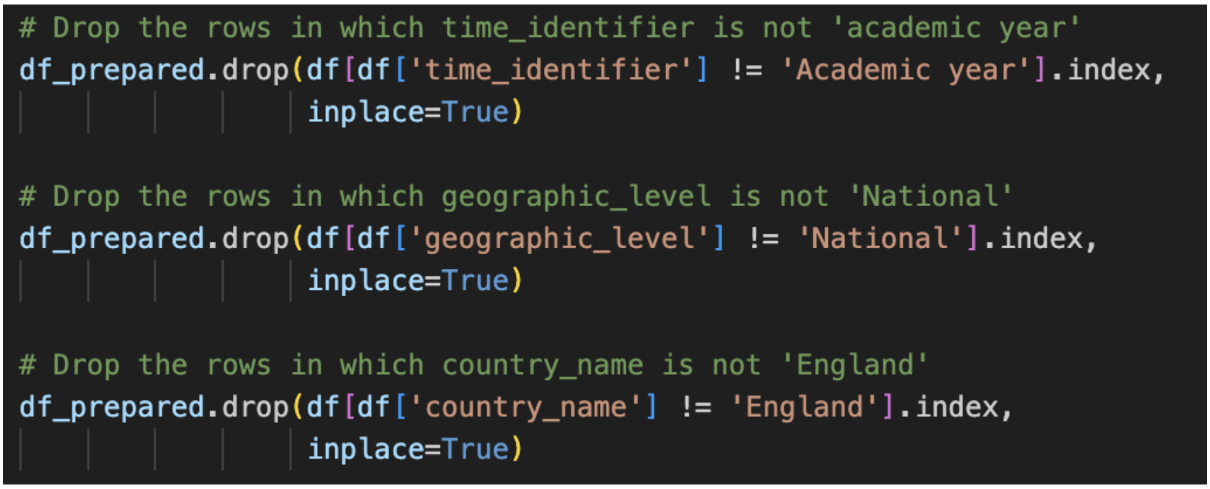
```python
def prepare_data(df):
    """ Takes the raw data and prepares it for later use

       Args:
           df: The raw data in a pandas DataFrame

       Returns:
        df_prepared: A pandas DataFrame with the prepared data

       """
```

In the function, as I only need necessary data such as percentage data to analyse and make data visualisation, I decided to delete all the columns which includes total something like total number of male teachers. Then I assigned it to a new variable named df_prepared.



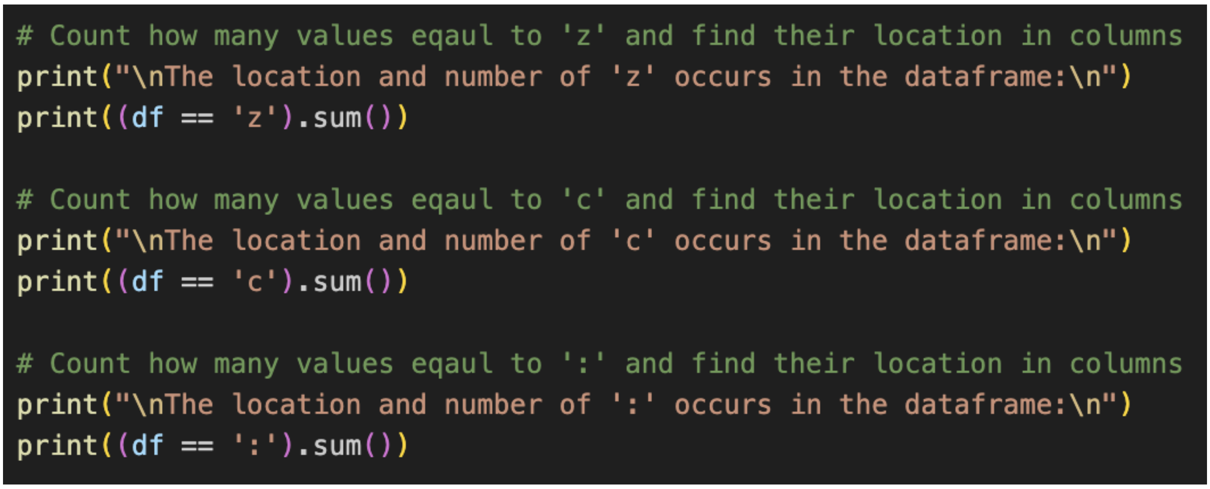
```
# Drop the list of columns which we don't need & assign to a new variable
# named df_prepared
df_prepared = df.drop(['total_age_u25', 'total_age_25andover',
                        'total_sex_m', 'total_sex_f',
                        'total_sex_other', 'total_sex_unknown',
                        'total_degreec_first', 'total_degreec_upper2nd',
                        'total_degreec_lower2nd', 'total_degreec_other',
                        'total_degreec_unknown', 'total_ethnic_asian',
                        'total_ethnic_black',
                        'total_ethnic_mixed_ethnicity',
                        'total_ethnic_other', 'total_ethnic_white',
                        'total_ethnic_unknown', 'total_disability',
                        'total_nondisability', 'total_disability_unknown'],
                       axis=1)
```

The raw data is provided by initial teacher training performance profiles which provide national and provider-level information about the outcomes for teacher trainees in England in the academic year. So I made a condition to delete unwanted rows. For the ‘time_identifier’ column, I only need the data which is in Academic year, so I deleted all rows which is not in Academic year. It’s the same process for the ‘geographic_level’ column and ‘country_name’ column. I deleted all the rows which are not ‘National’ and ‘England’.



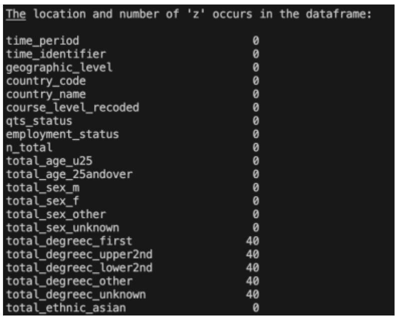
```
# Drop the rows in which time_identifier is not 'academic year'
df_prepared.drop(df[df['time_identifier'] != 'Academic year'].index,
                 inplace=True)

# Drop the rows in which geographic_level is not 'National'
df_prepared.drop(df[df['geographic_level'] != 'National'].index,
                 inplace=True)

# Drop the rows in which country_name is not 'England'
df_prepared.drop(df[df['country_name'] != 'England'].index,
                 inplace=True)
```

The data guidance states that some specific symbols are defined by different meanings. ‘c’ means small number suppressed to preserve confidentiality, ‘z’ means not applicable and ‘:’ means not available. As we already know that there is no missing values in the raw data, there may be some specific symbols. So I decided to find how many values equal to ‘z’, ‘c’ and ‘:’ and their location in columns.

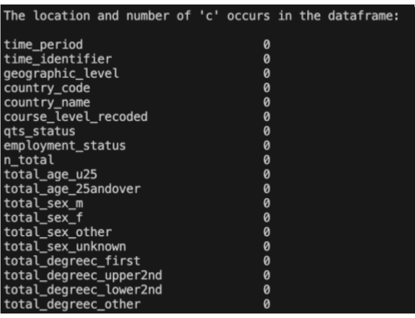


```
# Count how many values eqaul to 'z' and find their location in columns
print("\nThe location and number of 'z' occurs in the dataframe:\n")
print((df == 'z').sum())

# Count how many values eqaul to 'c' and find their location in columns
print("\nThe location and number of 'c' occurs in the dataframe:\n")
print((df == 'c').sum())

# Count how many values eqaul to ':' and find their location in columns
print("\nThe location and number of ':' occurs in the dataframe:\n")
print((df == ':').sum())
```

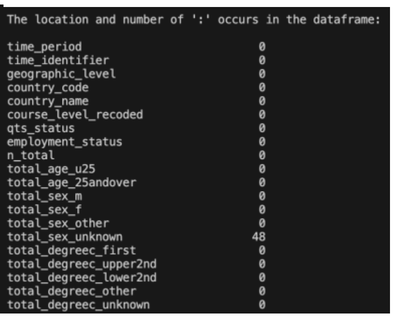
The results are shown below which we can find which columns contain ‘z’ or ‘c’ or ‘:’.



```
The location and number of 'z' occurs in the dataframe:

time_period                        0
time_identifier                    0
geographic_level                   0
country_code                       0
country_name                       0
course_level_recoded               0
qts_status                         0
employment_status                  0
n_total                            0
total_age_u25                      0
total_age_25andover                0
total_sex_m                        0
total_sex_f                        0
total_sex_other                    0
total_sex_unknown                  0
total_degreec_first               40
total_degreec_upper2nd            40
total_degreec_lower2nd            40
total_degreec_other               40
total_degreec_unknown             40
total_ethnic_asian                 0
```

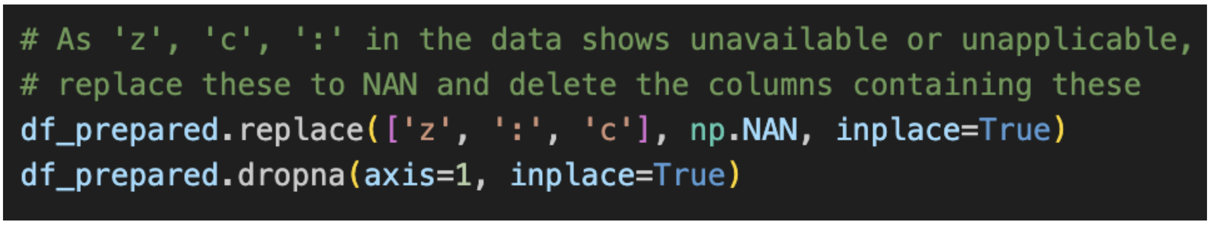


```
The location and number of 'c' occurs in the dataframe:

time_period                        0
time_identifier                    0
geographic_level                   0
country_code                       0
country_name                       0
course_level_recoded               0
qts_status                         0
employment_status                  0
n_total                            0
total_age_u25                      0
total_age_25andover                0
total_sex_m                        0
total_sex_f                        0
total_sex_other                    0
total_sex_unknown                  0
total_degreec_first                0
total_degreec_upper2nd             0
total_degreec_lower2nd             0
total_degreec_other                0
```



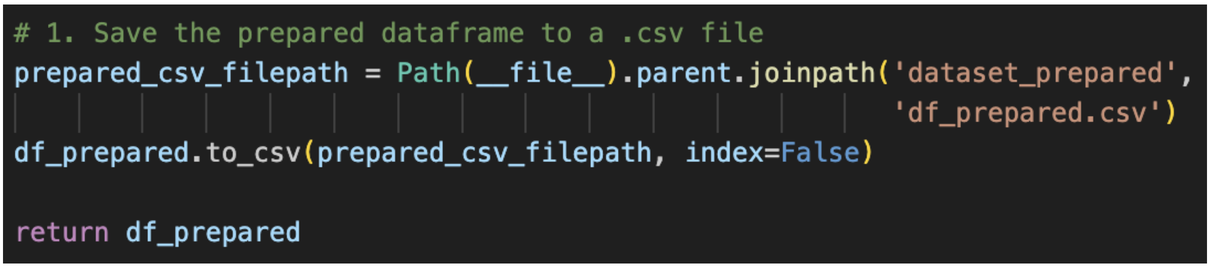
```
The location and number of ':' occurs in the dataframe:

time_period                        0
time_identifier                    0
geographic_level                   0
country_code                       0
country_name                       0
course_level_recoded               0
qts_status                         0
employment_status                  0
n_total                            0
total_age_u25                      0
total_age_25andover                0
total_sex_m                        0
total_sex_f                        0
total_sex_other                    0
total_sex_unknown                 48
total_degreec_first                0
total_degreec_upper2nd             0
total_degreec_lower2nd             0
total_degreec_other                0
total_degreec_unknown              0
```

There are only several columns containing these symbols and either there isn't any, or there's a lot. It makes the column containing ‘z’ and ‘c’ nearly useless. As a result, I decided to replace all the ‘z’, ‘:’ and ‘c’ to NAN values and then delete the columns containning them by .dropna.



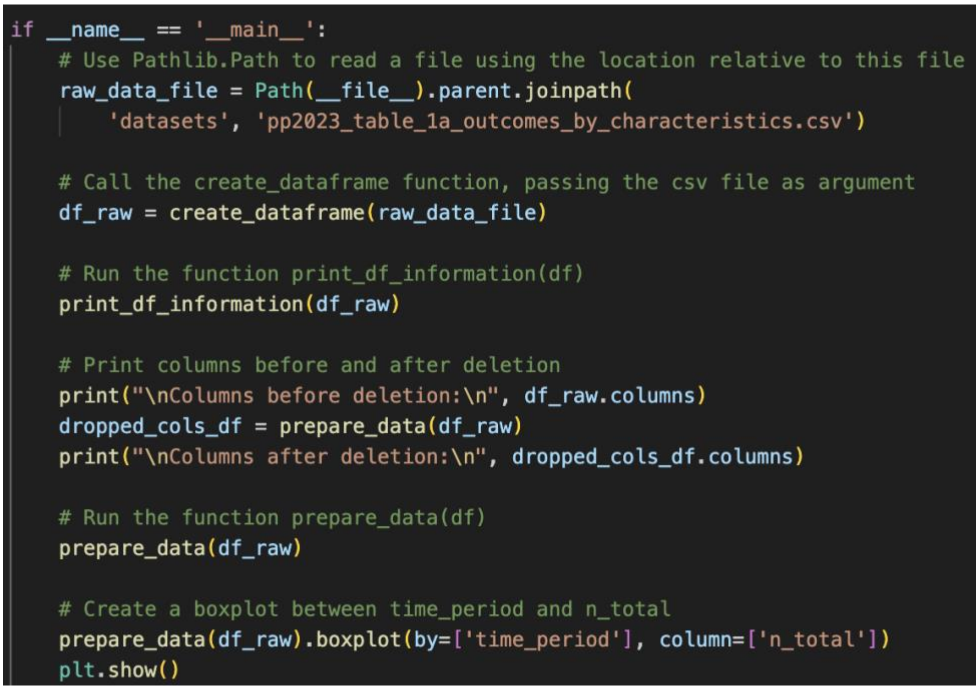
```
# As 'z', 'c', ':' in the data shows unavailable or unapplicable,
# replace these to NAN and delete the columns containing these
df_prepared.replace(['z', ':', 'c'], np.NAN, inplace=True)
df_prepared.dropna(axis=1, inplace=True)
```

When we already prepared the dataframe, we can save the prepared dataframe to a new .csv file named as ‘df_prepared.csv’ and then return the prepared dataframe.



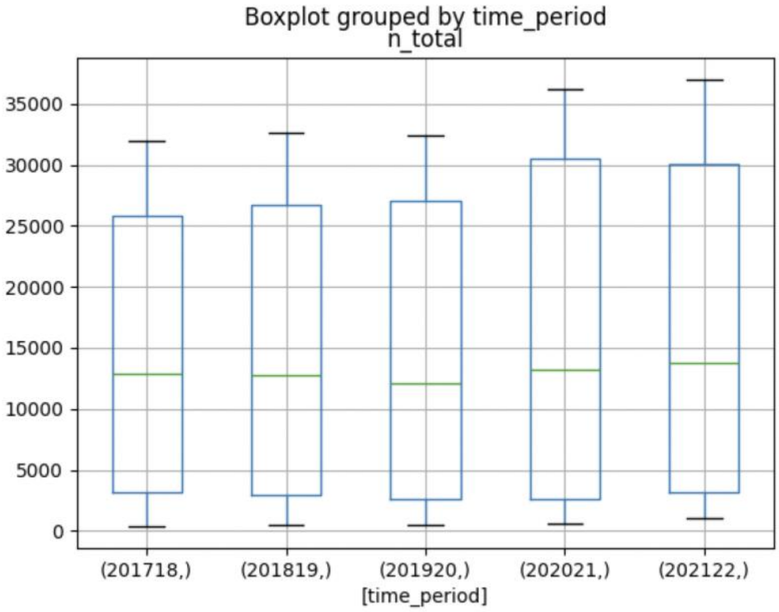
```
# 1. Save the prepared dataframe to a .csv file
prepared_csv_filepath = Path(__file__).parent.joinpath('dataset_prepared',
                                                      'df_prepared.csv')
df_prepared.to_csv(prepared_csv_filepath, index=False)

return df_prepared
```

In the end, if the __name__ == "__main__" expression is True, then the indented code following the conditional statement executes. Set the path of raw data and name it as raw_data_file. Next, run the functions we defined before. I created a boxplot between time_period and n_total. Boxplots can aid in providing us with a better understanding of the data distribution, which in turn facilitates the easier identification of outliers.



```
if __name__ == '__main__':
    # Use Pathlib.Path to read a file using the location relative to this file
    raw_data_file = Path(__file__).parent.joinpath(
        'datasets', 'pp2023_table_1a_outcomes_by_characteristics.csv')

    # Call the create_dataframe function, passing the csv file as argument
    df_raw = create_dataframe(raw_data_file)

    # Run the function print_df_information(df)
    print_df_information(df_raw)

    # Print columns before and after deletion
    print("\nColumns before deletion:\n", df_raw.columns)
    dropped_cols_df = prepare_data(df_raw)
    print("\nColumns after deletion:\n", dropped_cols_df.columns)

    # Run the function prepare_data(df)
    prepare_data(df_raw)

    # Create a boxplot between time_period and n_total
    prepare_data(df_raw).boxplot(by=['time_period'], column=['n_total'])
    plt.show()
```

The diagram of boxplot is shown below, which have box from lower quartile to the upper quartile, with the median marked.


Boxplot grouped by time_period
n_total
35000
30000
25000
20000
15000
10000
5000
0
(201718,)
(201819,)
(201920,)
(202021,)
(202122,)
[time_period]

2. Product and project definition

2.1. Problem statement

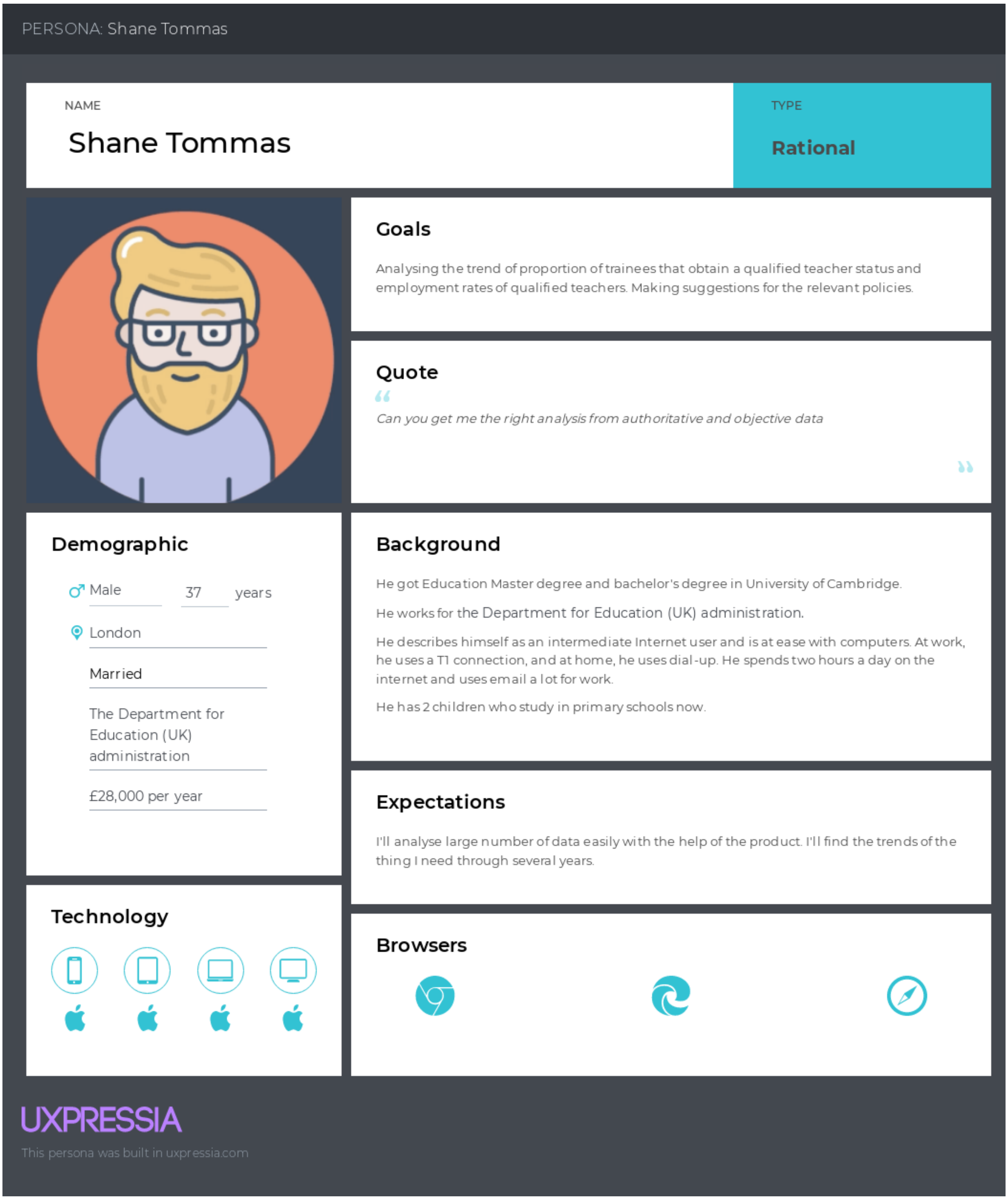
It’s an individual project, so no need for problem statement

2.2. Product overview

Project vision statement:

For	government staff in the education sector in England
Who	want analyze basic data about national teachers and then implement relevant policies in education
The Educator App	is a data visualisation app
That	analyze the proportion of trainees that obtain a qualified teacher status and the employment rates of these qualified teachers
Our product	will provide authoritative and objective data

2.3. Persona



PERSONA: Shane Tommas

NAME

Shane Tommas

TYPE

Rational

Goals

Analysing the trend of proportion of trainees that obtain a qualified teacher status and employment rates of qualified teachers. Making suggestions for the relevant policies.

Quote

“

Can you get me the right analysis from authoritative and objective data

”

Demographic

Male 37 years

London

Married

The Department for Education (UK) administration

£28,000 per year

Background

He got Education Master degree and bachelor's degree in University of Cambridge.

He works for the Department for Education (UK) administration.

He describes himself as an intermediate Internet user and is at ease with computers. At work, he uses a T1 connection, and at home, he uses dial-up. He spends two hours a day on the internet and uses email a lot for work.

He has 2 children who study in primary schools now.

Expectations

I'll analyse large number of data easily with the help of the product. I'll find the trends of the thing I need through several years.

Technology

Browsers

UXPRESSIA

This persona was built in uxpressia.com

2.4. Project goal & objectives / Questions

It’s an individual project, so no need for Project goal & objectives / Questions

3. Tools & techniques

3.1. Source code control

The URL of my repository is shown below:

https://github.com/ucl-comp0035/comp0035-cwi-SHOX1ie.git

3.2. Linting

I installed Pylint and Flake8 in the VS code. Then I followed the instructions and improved my code quality.

3.3. Project planning and tracking

It’s an individual project, so no need for Project planning and tracking

3.4. Use of AI

AI not used.

4. Methodology

4.1. Methodology selection

It’s an individual project, so no need for methodology part.

5. References

This is the website of raw dataset:

https://explore-education-statistics.service.gov.uk/find-statistics/initial-teacher-training-performance-profiles#releaseHeadlines-tables