Question 1.1: We have noticed that there are some missing values in the data (scroll all the way to the right of the dataframe to see some of the NaNs). What are the different ways of handling missing data? What are their advantages? What are their disadvantages?

There are multiple ways of handling missing data. First, we could use `ipums_raw.isna().any(axis = 0)` to look at which columns are affected – though there are NaN, it might not be relevant for our analysis. Next if we find that we will use affected columns in our analysis we could use `ipums_raw.isna().any(axis = 1)` to look at the affected rows. Here, we might find a pattern for the missing values, which might make us capable of replacing the NaN with an appropriate value, e.g., 0 or the mean value in the existing data – this, however, could affect one analysis going forward. Most often, the appropriate thing to do, therefore, is dropping the affected rows, though this reduces the number of rows we are working with, which can be unfortunate when running regressions, etc.

Question 3.5: Comment on the results shown above. Do they match your expectations? Could there be problems with the way this dataset is constructed?

We see an interesting trend on how wages for nuclear and petrolium engineering have fallen back whilst wages in the tech-sector have increased this could follow an expected trend with more high paid jobs comming in the IT-sector. However, if this were the case one could argue that Applied Mathematics should see an increase as well. Furthermore from my understanding of the table in the section "Selecting Relevant Variables" numbers are not adjusted for inflation which in turn would make me expect that all salaries should have gone up, not down. Additionally, there is a problem with this data. It is based on a rather small dataset, e.g. where there only 1 observation for Actuarial Science in 2009 and 4 in 2019 – this is simply to little data to conclude that salaries have gone down in this field.

```
In [24]: display(pd.concat([ipums_2009[ipums_2009['DEGFIELDD'] == 'Actuarial Science'],ipums_2019[ipums_
```

```
    index  YEAR  AGE  SCHOOL  EDUCD  DEGFIELD          DEGFIELDD  EMPSTAT  \
0   53446  2009   38       1    114  Business  Actuarial Science        1
1   96068  2019   26       1    101  Business  Actuarial Science        3
2  115804  2019   37       2    114  Business  Actuarial Science        1
3  117109  2019   23       1    101  Business  Actuarial Science        1
4  127231  2019   47       1    114  Business  Actuarial Science        1

    OCC  INCTOT  INCWAGE  INCWAGE_CPIU_2010
0  1200  350700   356000           361839.0
1  9130       0        0                0.0
2  1006    1300     1300             1109.0
3  2545    5000     5000             4265.0
4   860  627100   626000           533930.0
```

Question 4.2: Identify the parameters and their values you must use if you want to obtain year-over-year real GDP percentage changes (i.e. percent change from year ago values).

Hint: You can look up the `series_id` of real GDP by searching it on Fred's website. It should be listed in parentheses next to the name of the series.

To find the YoY real change in GDP no more than just the real GDP, from here the rest can be calculated in python. Thus the only necessary dataset is the Real GDP GDPC1, and the only paramters needed to specify is my `api_key` and `series_id = GDPC1`.

Though not necesary one could specify `units = pc1` but this is not stricly necessary as I could just do the same calculation in python instead. Additionally filetype could be specified as either `file_type = json` or `file_type = xls`, but we could also just use the defaul `xml` output. The data is already aggregated on a quaterly level hence there is no need to change this unless we want something different and not specifying dates would just give us the full dataset.

Thus as stated the *only necessary* parameters would be my `api_key` and `series_id = GDPC1`.