Data Quality Plan

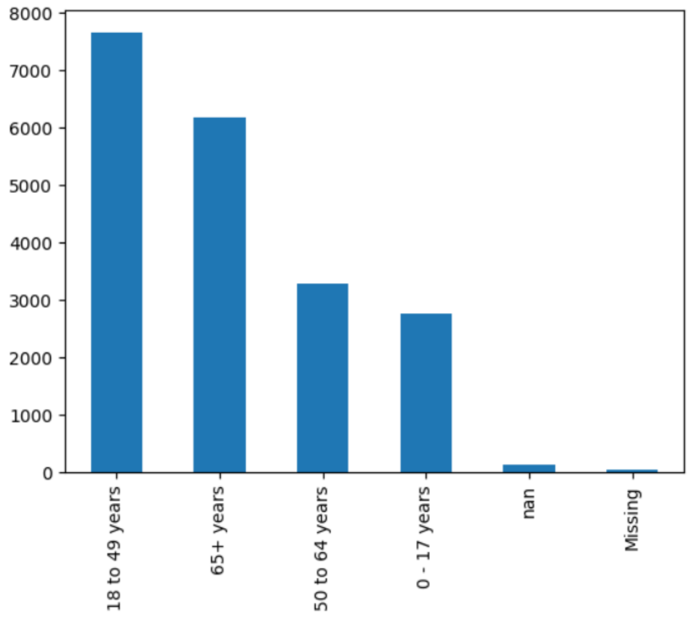
Preparing a data quality plan for the cleaned CSV file.

First I will give a comment on each feature and how useful it will be in our investigation of the effects of COVID on death rates.

First we will start with the categorical data points in order in which they appear in the cleaned CSV file.

1. Case_month:
   - Nothing seems to be out of the ordinary here, there is good quality data that shows the trend of how many cases were reported throughout the three year period of reporting.
2. Age_group:
   - There are some values that need to be cleaned up here as they fall under "Not a Number" or Missing.

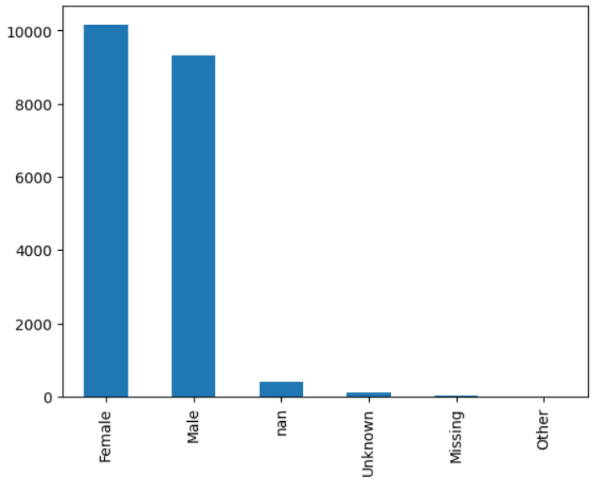
```
In [74]:    df_clean['age_group'].value_counts(dropna = False).plot(kind='bar')
   Out[74]: <AxesSubplot: >
```



   Solution:
   - A simple solution for this problem to be to drop the rows that contain 'nan' or missing values. While we want to keep as much data as possible, there are so few rows that contain these values overall that it will have a very minimal effect on the dataset

3. sex:
   - As with age_group, there are some missing values but once again the majority of the data contains the value of either 'male' or 'female'

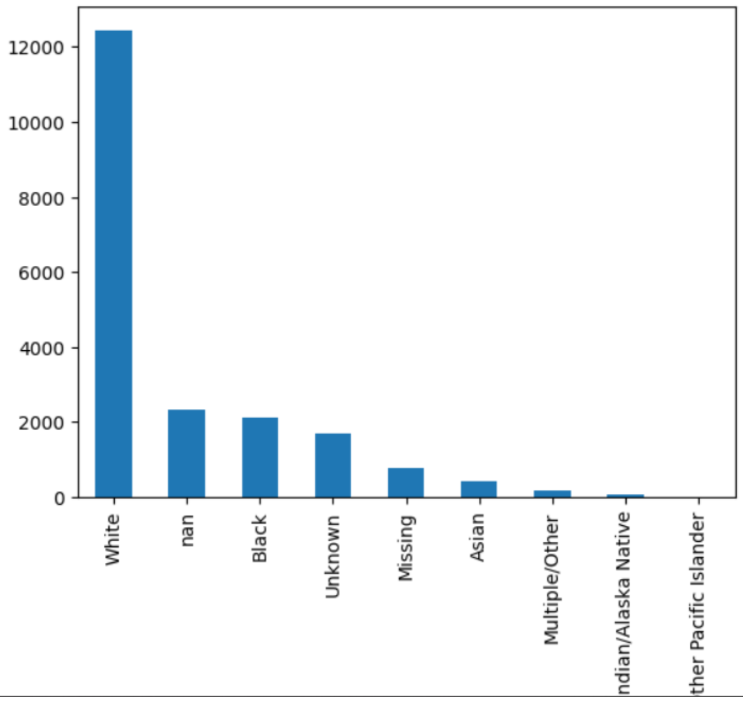
```
In [75]:    df_clean['sex'].value_counts(dropna = False).plot(kind='bar')
   Out[75]: <AxesSubplot: >
```



Solution:
We can use the same solution for age_group and can drop the rows that contain 'nan', 'unknown', 'missing', and 'other'

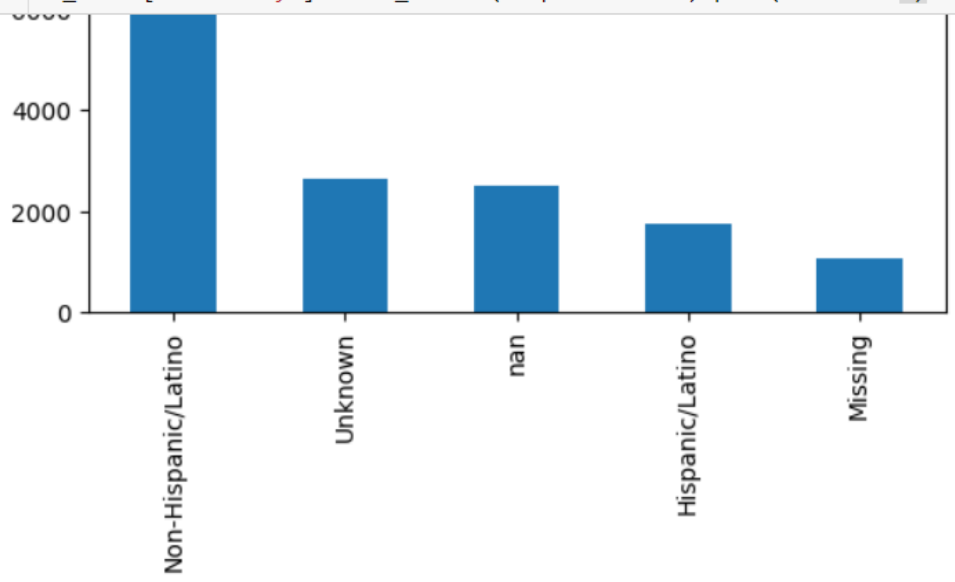
4. race:
   - Most things seem to be in order however there are some 'nan' values, Missing values, and Unknown values.

```
In [76]:    df_clean['race'].value_counts(dropna = False).plot(kind='bar')
   Out[76]: <AxesSubplot: >
```



5. ethnicity:
   - Once again this is similar to the above race attribute, and once again we have the same problem areas of 'nan', Unknown, and Missing values.

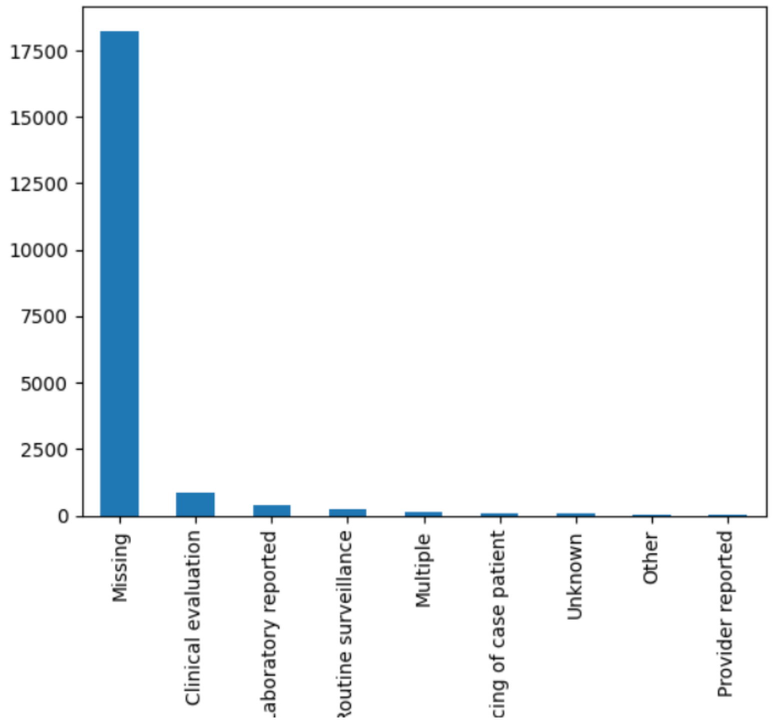
```
In [89]: df_clean['ethnicity'].value_counts(dropna = False).plot(kind='bar')
```



6. process:
   - We see a huge lack of information in relation to this attribute in the dataset, to the point where it is not useful to us.

```
In [78]: df_clean['process'].value_counts(dropna = False).plot(kind='bar')

Out[78]: <AxesSubplot: >
```

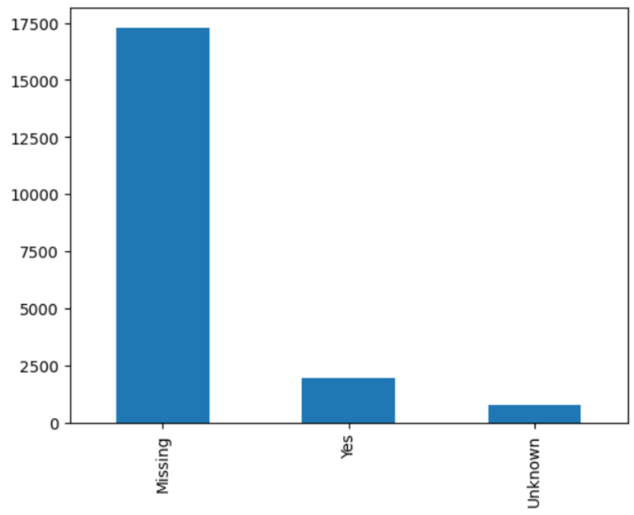


Solution:

I think the right course of action here is to drop this attribute, as the majority of the data could not be used in any profitable way for data analyses

7. exposure_yn:
   - Much like the previous attribute, we have a huge amount of missing data for this attribute

```
In [79]:    df_clean['exposure_yn'].value_counts(dropna = False).plot(kind='bar')
   Out[79]: <AxesSubplot: >
```

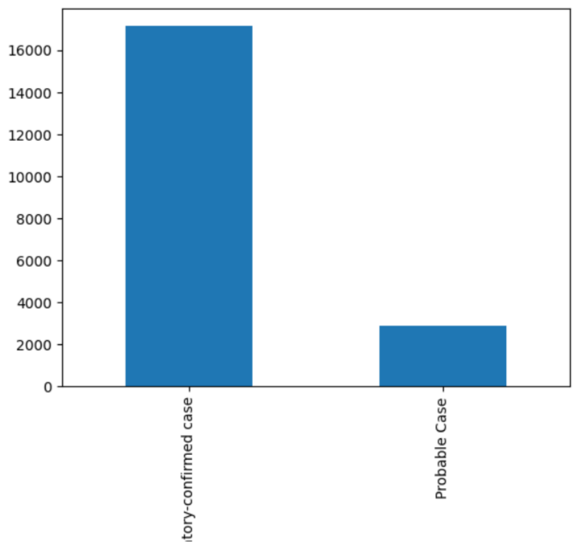


Solution:

Once again I feel as though drop this attribute as a whole would be the best action as the data one would get from rows containing ‘Yes’ is insignificant compared to the amount of data available

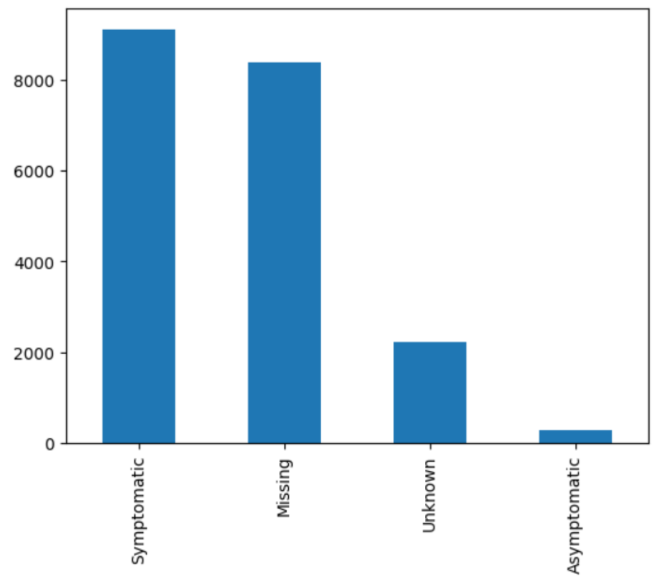
8. current_status:
   - The current_status attribute contains perfect data, not needing to be cleaned at all. No modifications will need to be made for this, which is excellent

```
In [80]:    df_clean['current_status'].value_counts(dropna = False).plot(kind='bar')
   Out[80]: <AxesSubplot: >
```



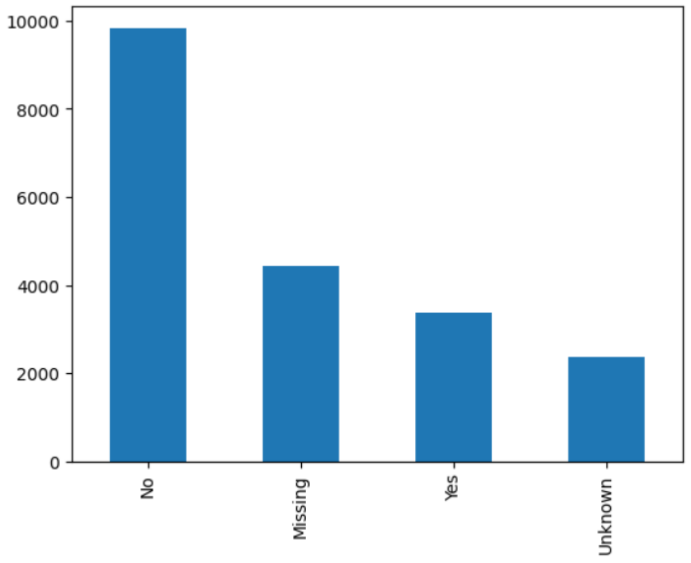
9. symptom_status:
   - This is a little awkward as half of the data contained in this attribute is legitimate, however the other half does not give us a huge amount of information

```
In [81]:   df_clean['symptom_status'].value_counts(dropna = False).plot(kind='bar')
   Out[81]: <AxesSubplot: >
```



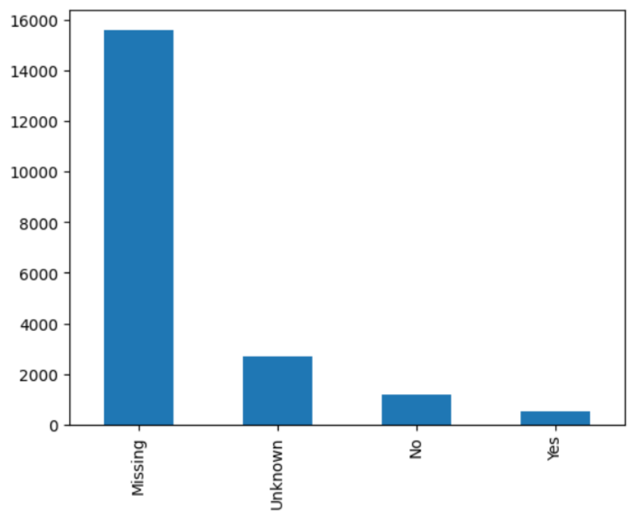
10. hosp_yn:
    - Same as the previous attribute, a lot of missing data but the data is legitimate therefore there is an argument to leave this alone

```
In [82]:   df_clean['hosp_yn'].value_counts(dropna = False).plot(kind='bar')
   Out[82]: <AxesSubplot: >
```



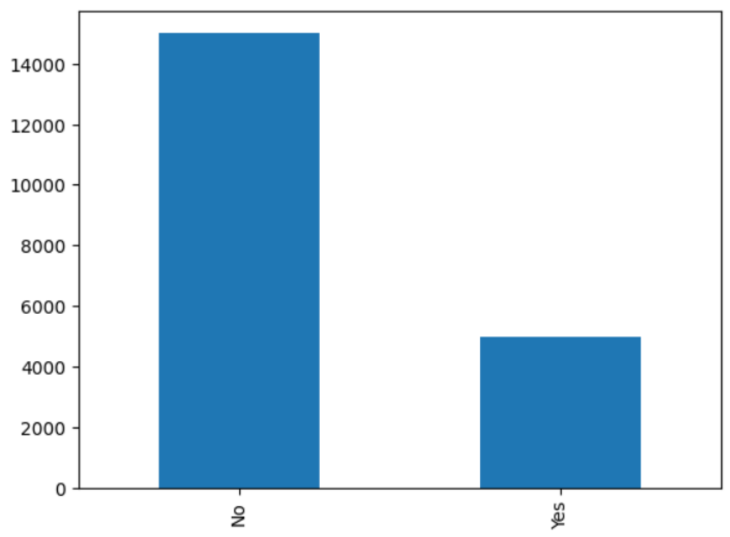
11. icu_yn:
    - Once again, we have a lot of

```
In [83]:  df_clean['icu_yn'].value_counts(dropna = False).plot(kind='bar')
Out[83]: <AxesSubplot: >
```



12. death_yn:
    - Perfect, nothing needs to be altered

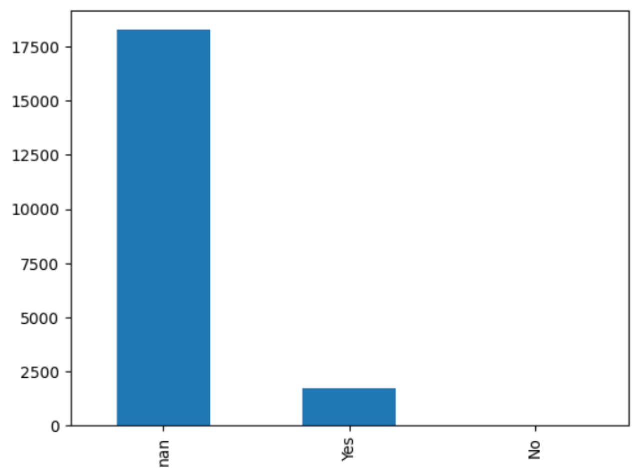
```
In [84]:  df_clean['death_yn'].value_counts(dropna = False).plot(kind='bar')
Out[84]: <AxesSubplot: >
```



13. Underlying_conditions_yn:
    - May need to change things here

```
In [86]:    df_clean['underlying_conditions_yn'].value_counts(dropna = False).plot(kind='bar')

  Out[86]: <AxesSubplot: >
```



Continuous Features:

There are only two continuous features in our dataset that we need to examine which are the following:

- case_positive_specimen_interval
- case_onset_interval

case_positive_specimen_interval:

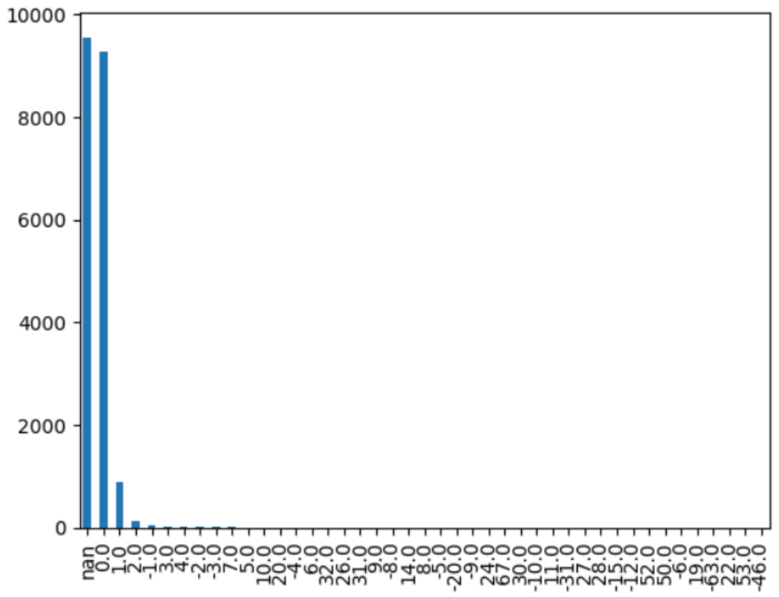
Description:

“Weeks between earliest date and date of first positive specimen collection”

We could describe this feature as when the patient first showed up on records, compared to when it was actually confirmed that the patient has tested positive for COVID.

These are the results when we graph it out

```
In [35]:  df_clean['case_positive_specimen_interval'].value_counts(dropna = False).plot(kind='bar')
   Out[35]: <AxesSubplot: >
```



Problems:

- We see that there are a lot of values that fall under ‘nan’. The second largest value falls under 0.0, therefore we can assume that the date between the earliest record of the patient and when it was confirmed that they had COVID was under a week.

Solution:

- Seeing as the majority of the data is of no use to us, and the second largest piece of data it too broad in scope, there isn’t a lot of value in using this feature in our analysis.

case_onset_interval

Description:

“Weeks between earliest date and date of symptom onset”

This feature is quite similar to that of case_positive, but instead we have the time in weeks from when the patient records were first created to when the patient first felt symptoms of COVID.

Here is what it looks like in terms of the graph