**Data Cleaning:** Data cleansing or data cleaning is the process of detecting and correcting corrupt or inaccurate records from a recordset, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

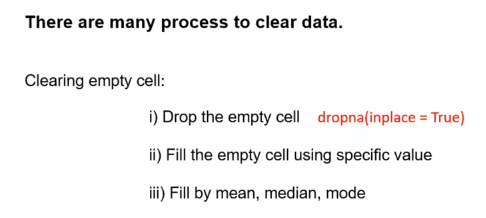
**Bad data could be:**

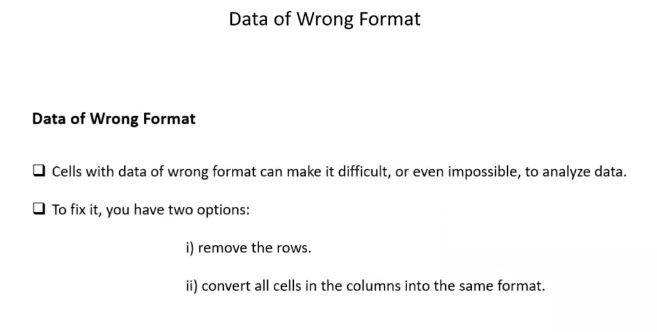
* Empty cells
* Data in the wrong format
* Wrong data
* Duplicates

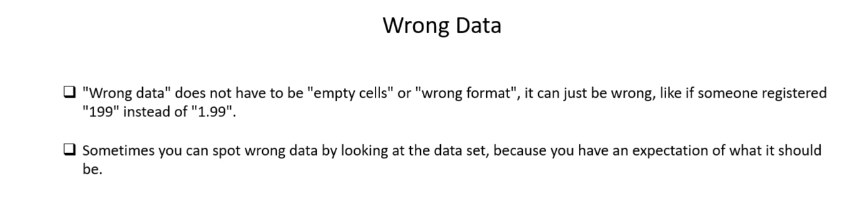
If we encounter such bad data in any of the columns, We will dismiss the whole row from the file. Especially, for duplicate values.

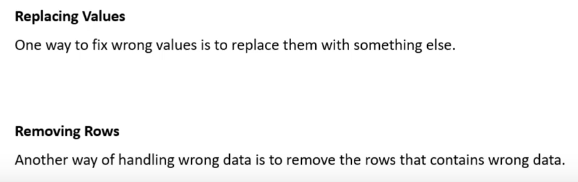
Or, we can replace bad data by taking the mean/median/mode of the other data of that column.

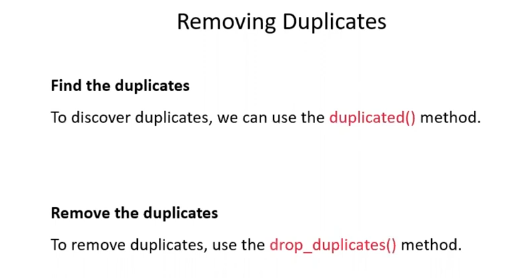
**Data cleaning process**

****

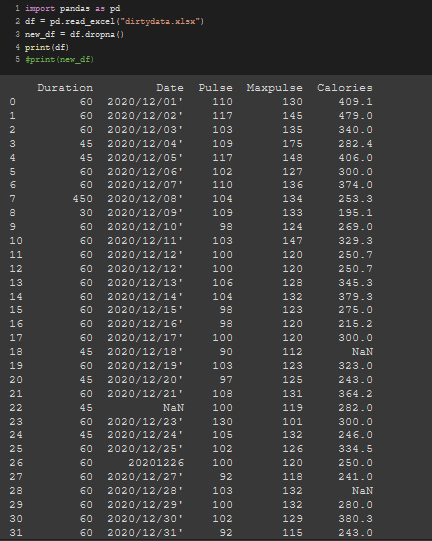




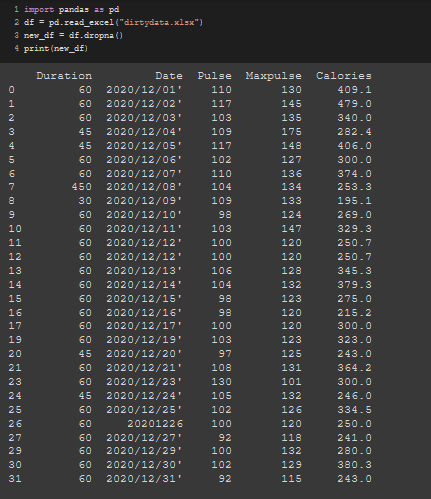




Let a dataframe be,

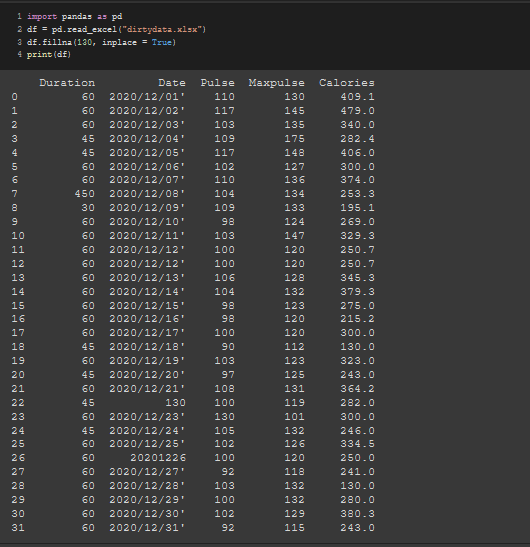


* To drop entire rows with null values, We can use ***.dropna()*** method.

Note that there are several *“Nan”* values in lines like 18,22 and etc. 

* To replace a null value in the dataframe, we can use the ***“.fillna(“value”, inplace = True)”*** method.

Note that there are several *“Nan”* values in lines like 18,22 and etc. And now, They are replaced with 130.

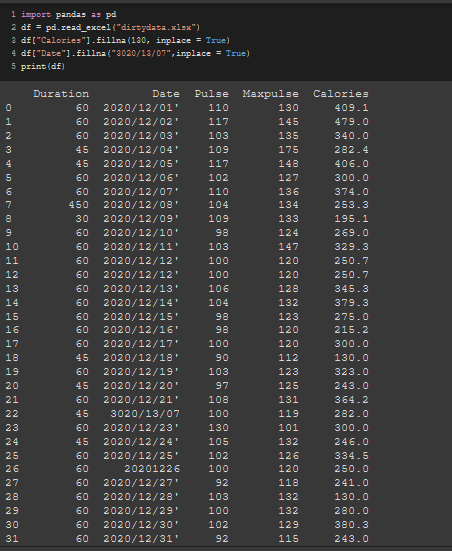


The problem with setting specific values is that it does not abide by the types of data that is in the columns.

* That is why, When we set any specific value, we should specify the columns also.

We can do it using

***dataframe\_name***[“***Column\_name***"].fillna("***Replace\_value***",inplace = True)



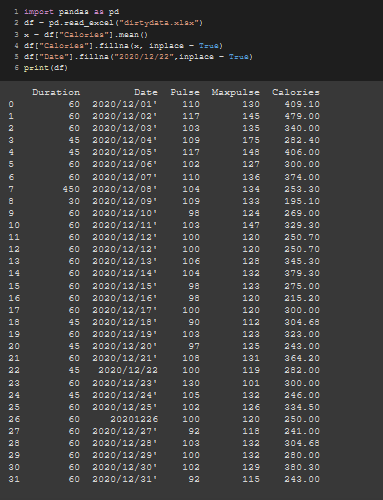
Note that there are several *“Nan”* values in lines like 18 and 28 in the calories column which is now replaced with 130.

And for date, 22th row of the date column had null value before but now it has a new date “30/13/07”

* Now, the problem with this technique is the specific value might not sync or blend in the other values of that column. If we insert a value that can be obtained using mean/median/mode on other values of that column, that value will sync in or blend in relatively better.

We need to calculate the mean/median/mode first using this format.

***variabel \_name*** = ***dataframe***["***Column\_name***"].***technique***()



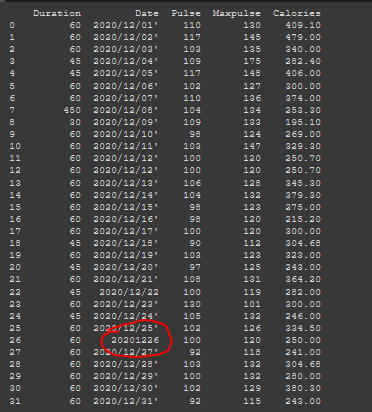
Note that there are several *“Nan”* values in lines like 18,22 and etc. And now, They are replaced with the mean value of the column.

We need to do all mean/median/mode operations and select the best fit value.

**Formatting wrong data**

To fix the format of the wrong data, we need to convert the whole column into the appropriate format.

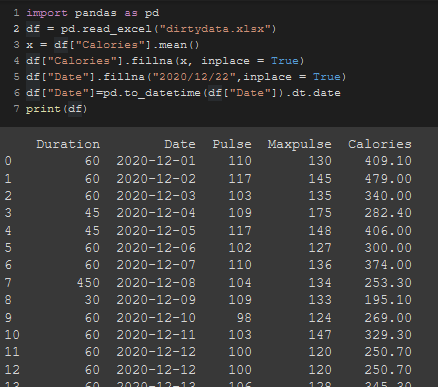
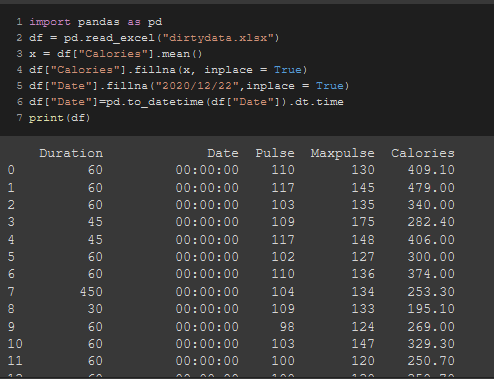
Eg: We have a date here but in the wrong format. We need to format it in a DateTime format.



We can do using this syntax,

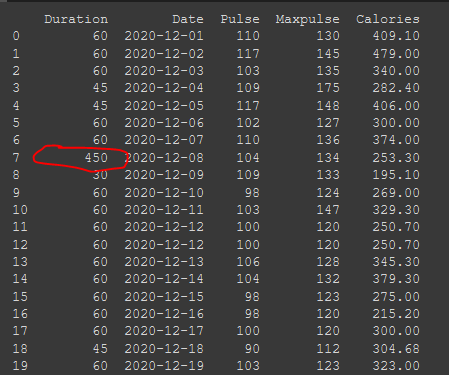
***new\_dataframe***["***Column\_name***"] = pd.to\_***NewFormat***(***old\_dataframe***["***Column\_name***"] ]).dt.***date/time***  [Optional based on actually what we want to see]

Later, We can convert it into a string and print it.

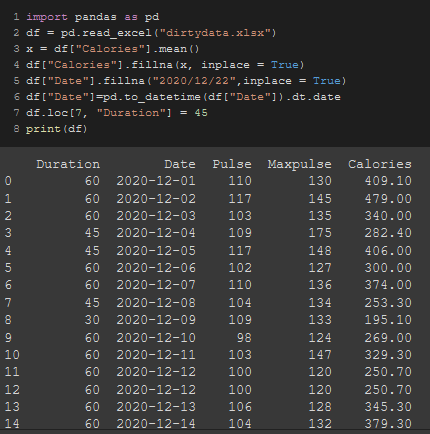
 

* If we want to replace a specific value within a row/column, We can do that using this format,

***data\_frame***.loc[***"row\_number", "column\_name"***] = ***new\_value***



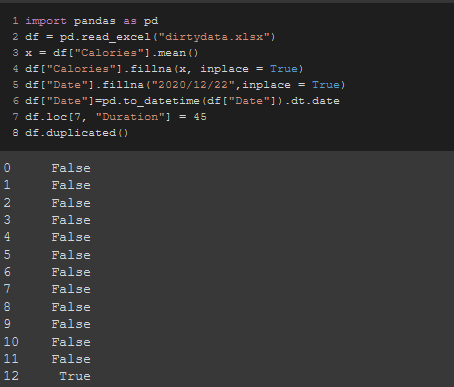
Lets say, We want to change this ***450*** into ***45***. We need to set the exact row number (7 Here) & Column name (“Duration”)



* If we have duplicate data, the Entire row that is. We need to drop those duplicate rows and keep only one because of no point in having the same data over and over again.

Before dismissing the rows, We need to identify those first. We can do that using this format,

***Dataframe***.duplicated()



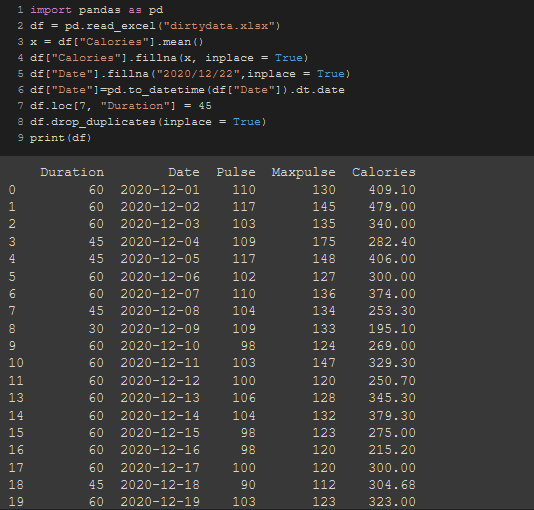
All of them are unique except the 12th column. We don’t know which column’s duplicate this one is but this one is duplicate for sure

But in this case, It is the duplicate of 11th number column.



* We can drop these duplicate values using

***data\_frame***.drop\_duplicates(inplace = True)



We can now see there’s no longer any 12th number row.