In [1]:

```python
import pandas as pd
import numpy as np
pd.set_option("display.max_rows", None)
data = pd.read_csv('cleaning_final_task.csv')
data.head()
```

Out[1]:

	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_gain	capital_loss
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0

AGE: ALL VALUES MUST BE NUMERIC

In [2]:

```python
data['age'].value_counts()
# All values in the column are numeric, so there is no cleaning to be done here.
```

Out[2]:

```
36    898
31    888
34    886
23    877
35    876
33    875
28    867
30    861
37    858
25    841
27    835
32    828
38    827
39    816
29    813
41    808
24    798
40    794
26    785
42    780
43    770
22    765
20    753
46    737
45    734
44    724
21    720
19    712
47    708
50    602
51    595
49    577
18    550
48    543
52    478
53    464
55    419
54    415
17    395
56    366
58    366
57    358
59    355
60    312
61    300
62    258
63    230
64    208
65    178
67    151
66    150
68    120
69    108
70     89
71     72
72     67
73     64
74     51
76     46
75     45
90     43
77     29
78     23
79     22
80     22
81     20
82     12
84     10
83      6
88      3
85      3
86      1
87      1
Name: age, dtype: int64
```

WORKCLASS: SOME COLUMNS ARE WEIRD, LETS FIX THEM

In [3]:

```python
data.update(data['workclass'].replace({' Private':'Private',' Self-emp-not-inc':'Self-emp-not-inc',' Local-gov':'Local-gov',' ?':np.NaN,' State-gov':'State-gov',' Self-emp-inc':'Self-emp-inc',' Federal-gov':'Federal-gov',
                                       ' Without-pay':'Without-pay',' Never-worked':'Never-worked',' Privaate':'Private',' Privatea':'Private',' Privateas':'Private','1':np.NaN,'Unknow':np.NaN}, inplace=True))
data['workclass'].value_counts()
# Unknown values have been replaced with np.NaN, which should be equal to NaN im pretty sure.
```

Out[3]:

```
Private             22694
Self-emp-not-inc     2540
Local-gov            2093
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

EDUCATION: VALUES NEED REPLACING (WORD WISE)

In [4]:

```python
data['education'] = data['education'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['education'].replace({'HS-grads':'HS-grad','Baachelors':'Bachelors','HS-grades':'HS-grad','Bacahelors':'Bachelors','10ths':'10th',
                                        'Bacheloors':'Bachelors','Proof-school':'Prof-school','Soome-college':'Some-college'}, inplace=True))


data['education'].value_counts()
```

Out[4]:

```
HS-grad         10501
Some-college     7291
Bachelors        5355
Masters          1723
Assoc-voc        1382
11th             1175
Assoc-acdm       1067
10th              933
7th-8th           646
Prof-school       576
9th               514
12th              433
Doctorate         413
5th-6th           333
1st-4th           168
Preschool          51
Name: education, dtype: int64
```

EDUCATION_NUM: THERE ARE 2 EMPTY VALUES, FROM OTHERS IM NOT SURE WHAT TO CHANGE

In [5]:

```python
#data['education_num'] = data['education_num'].astype(str) + 'L'
# I used this to check how many spaces the empty values have, its pretty self explanatory.

data.update(data['education_num'].replace({'    ':np.NaN,'      ':np.NaN}, inplace=True))
data['education_num'].value_counts()
```

Out[5]:

```
9     10499
10     7288
13     5353
14     1723
11     1382
7      1175
12     1067
6       933
4       646
15      576
5       514
8       433
16      413
3       333
2       168
1        51
Name: education_num, dtype: int64
```

MARITAL_STATUS: VALUES NEED FIXING (TYPOS) AND SOME CHANGING TO NANs

In [6]:

```python
data['marital_status'] = data['marital_status'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['marital_status'].replace({'Mararied-civ-spouse':'Married-civ-spouse','a':np.NaN,'Never-marrs':'Never-married','Marriedsspouse':'Married-spouse-absent','Marriaed-civ-spouse':'Married-civ-spouse',
                                            'Marraied-civ-spouse':'Married-civ-spouse','Married-civ':'Married-civ-spouse','Sepsarated':'Separated','Divorcsed':'Divorced','Marr':np.NaN}, inplace=True))
# 'Marr' was replaced with NaN, even though you can see that it should be 'Married' we still dont know which one
```

OCCUPATION: FIXING TYPOS

In [7]:

```python
data['occupation'] = data['occupation'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['occupation'].replace({'Machine-op-inspct':'Machine-op-inspect','?':np.NaN,'Protective-serv':'Protective-service','Priv-house-serv':'Priv-house-service','Armed-Forces':'Armed-forces',
                                        'Execs-managerial':'Exec-managerial','Exec':'Exec-managerial',}, inplace=True))
data['occupation'].value_counts()
```

Out[7]:

```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspect   2002
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-service    649
Priv-house-service    149
Armed-forces            9
Name: occupation, dtype: int64
```

RELATIONSHIP: FIXING TYPOS

In [8]:

```python
data['relationship'] = data['relationship'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['relationship'].replace({'Wife-':'Wife','Own-schild':'Own-child','Husbsand':'Husband','Others-relative':'Other-relative','Unsmarried':'Unmarried','Hussband':'Husband',
                                          'Nots-in-family':'Not-in-family','Nost-in-family':'Not-in-family'}, inplace=True))
data['relationship'].value_counts()
```

Out[8]:

```
Husband           13193
Not-in-family      8305
Own-child          5068
Unmarried          3446
Wife               1568
Other-relative      981
Name: relationship, dtype: int64
```

RACE: FIXING TYPOS AND CHANGING TO NANs

In [9]:

```python
data.update(data['race'].replace({'w':' w'}, inplace=True))
# I gave this record one space so that the next line of code works without removing it

data['race'] = data['race'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['race'].replace({'Amer-Indian-Eskimo':'American-Indian-Eskimo','w':np.NaN,'Whitea':'White','Wite':'White','Bl':'Black','Wh':'White','Whra':np.NaN}, inplace=True))
# Some values like 'w' and 'Whra' I put as NaN because there is not enough evidence to put it was 'White'. Yes it begins with W but thats not enough

data['race'].value_counts()
```

Out[9]:

```
White                   27808
Black                    3124
Asian-Pac-Islander       1039
American-Indian-Eskimo    311
Other                     271
Name: race, dtype: int64
```

GENDER: FIXING TYPOS

In [10]:

```python
data['gender'] = data['gender'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['gender'].replace({'F':'Female','Feemale':'Female','Mmale':'Male','Femmale':'Female','Maale':'Male','M':'Male','Fem':'Female','Femas':'Female'}, inplace=True))

data['gender'].value_counts()
```

Out[10]:

```
Male      21790
Female    10771
Name: gender, dtype: int64
```

CAPITAL_GAIN: NOTHING, ALL VALUES ARE 'NORMAL' VALUES FOR CAPITAL GAIN, NOTHING IS TOO MUCH OR TOO LESS ACCORDING TO THE INTERNET

#

CAPITAL_LOSS: SAME AS CAPITAL_GAIN IN MY EYES

#

HOURS_PER_WEEK: CHANGING HOURS OVER 119 (EXPLAINED IN THE NEXT CELL)

In [11]:

```python
# Here is my thought process:
# 168 Hours in a week. A human needs 7 hours per day of sleep. 168 - 7 * 7 = 119. Anything over 119 is getting "NaN:ed"
data.loc[data["hours_per_week"] > 119] = np.NaN
# This function basically does what was explained in the first 2 lines
```

NATIVE_COUNTRY: FIXING TYPOS

In [13]:

```python
data.update(data['native_country'].replace({'USA':' USA','?':' ?'}, inplace=True))
# Adding a space so that the next line works better

data['native_country'] = data['native_country'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['native_country'].replace({'Unitsed-States':'United-States','?':np.NaN,'Trinadad&Tobago':'Trinadad & Tobago','United-   States':'United-States','Mex ico':'Mexico',' -States':'United-States',
                                            'Jap an':'Japan','El-Salvadore':'El-Salvador','United-Stastes':'United-States','Columbiaas':'Columbia','USA':'United-States','United-Ses':'United-States',
                                            'United States':'United-States','United':'United-States','Mexicos':'Mexico','Germany-':'Germany','Irans':'Iran','United.  States':'United-States',
                                            'Philippiness':'Philippines','Tai wan':'Taiwan','Me xico':'Mexico'}, inplace=True))

data['native_country'].value_counts()
```

Out[13]:

```
United-States                 29164
Mexico                          643
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        64
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
France                           29
Greece                           29
Ecuador                          28
Ireland                          24
Hong                             20
Cambodia                         19
Trinadad & Tobago                19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Honduras                         13
Hungary                          13
Scotland                         12
Holand-Netherlands                1
Name: native_country, dtype: int64
```

INCOME_BRACKET: FIXING TYPOS; ALTHOUGH I AM NOT SURE IF THIS IS CORRECTLY DONE

In [16]:

```python
data['income_bracket'] = data['income_bracket'].str[1:]
# This removes the first character of each value, this helps me ALOT in the sense that I dont have to rename each value in .replace()

data.update(data['income_bracket'].replace({'>50K':'>=50K','<50K':'<=50K'}, inplace=True))

data['income_bracket'].value_counts()
```

Out[16]:

```
<=50K    24718
>=50K     7840
Name: income_bracket, dtype: int64
```

In [ ]:

```python
### There it is, I am pretty sure all columns are cleaned in the right way (well atleast a beginner level cleaning way), its not beautiful but it should atleast be passable.
```