Introduction :-

we got this data set from UCI site (adults)

COLUMNS :-

age : The age of each person

workclass : The type of sector this person works in

fnlwgt : Final Weight of the person after the decrease and increase

education : The last education stage

educationN : Number of education years

MaritalStatus

Occupation

Relationship

Rece : white , black , etc..

Sex : male , female

capitalGain : Weight gain

capitalloss : Weight loss

hoursPerWeek : Working hours

country

salary : Greater or less than 50K or equal

The issues in adults data set is divided into two parts :

1) **Quality**

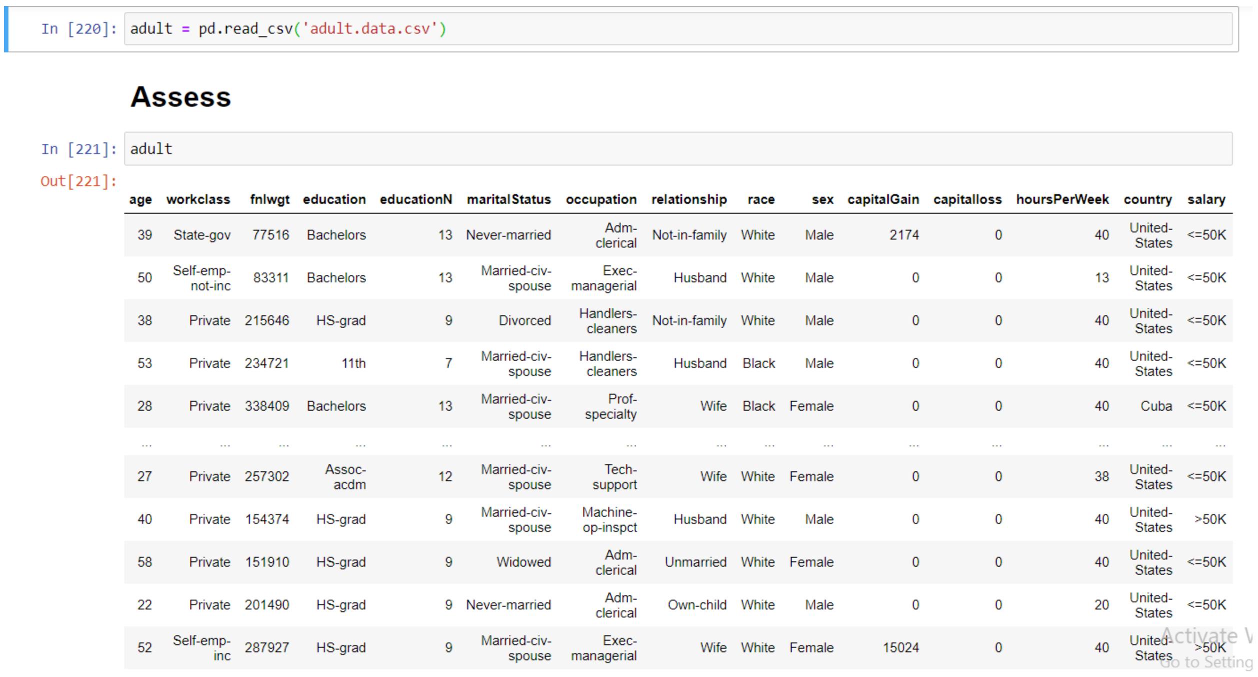
- **Noisy data in workclass column**
- **Adjust wrong values in relationship column into single or taken based on their Marital Status**
- **Sex and relationship is category not object**
- **Fnlwgt is float not int**
- **Fnlwgt is kg not g**
- **New column called timework based on hoursPerWeek**
- **Unnecessary column (hoursPerWeek)**
- **Abbreviation values in country**
- **Missing values in country**
- **Noisy data in fnlwgt**
- **Noisy data in occupation**
- **Knowing poor or rich based on salary**

2) **Tidiness**

- **Duplicated rows**
- **Merge capitalloss and capitalgain in one column**

NOW LETS START TO TALK ABOUT OUR EFFORTS IN THIS DATA SET

- view the data before cleaning



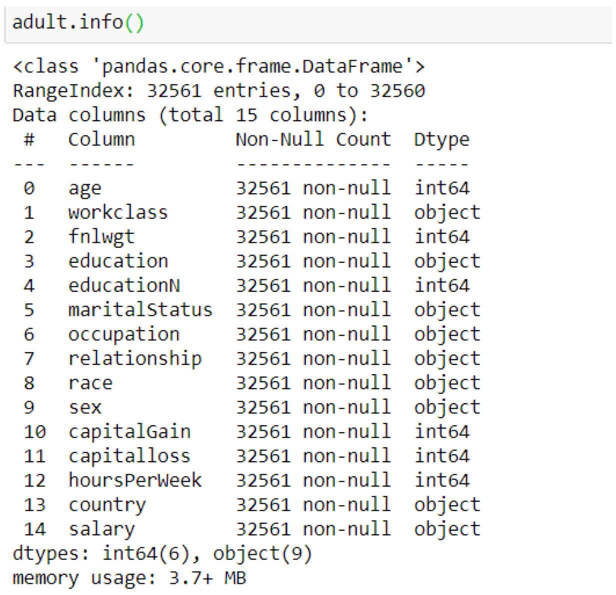
```
In [220]: adult = pd.read_csv('adult.data.csv')
```

Assess

```
In [221]: adult
Out[221]:
```

age	workclass	fnlwgt	education	educationN	maritalStatus	occupation	relationship	race	sex	capitalGain	capitalloss	hoursPerWeek	country	salary
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

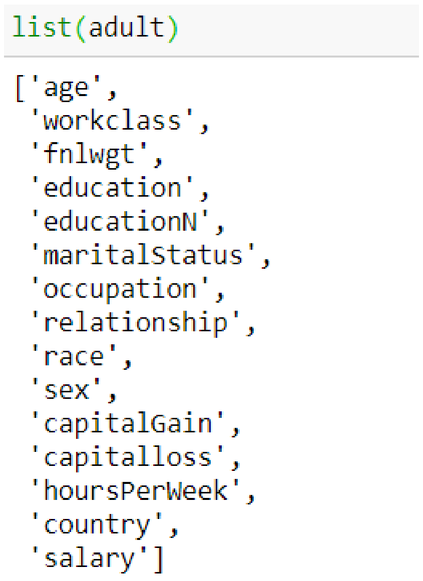
- Knowing some information about the data , data type for each variable and number of null counts



```
adult.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            32561 non-null  int64 
 1   workclass      32561 non-null  object
 2   fnlwgt         32561 non-null  int64 
 3   education      32561 non-null  object
 4   educationN     32561 non-null  int64 
 5   maritalStatus  32561 non-null  object
 6   occupation     32561 non-null  object
 7   relationship   32561 non-null  object
 8   race           32561 non-null  object
 9   sex            32561 non-null  object
 10  capitalGain    32561 non-null  int64 
 11  capitalloss    32561 non-null  int64 
 12  hoursPerWeek   32561 non-null  int64 
 13  country        32561 non-null  object
 14  salary         32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

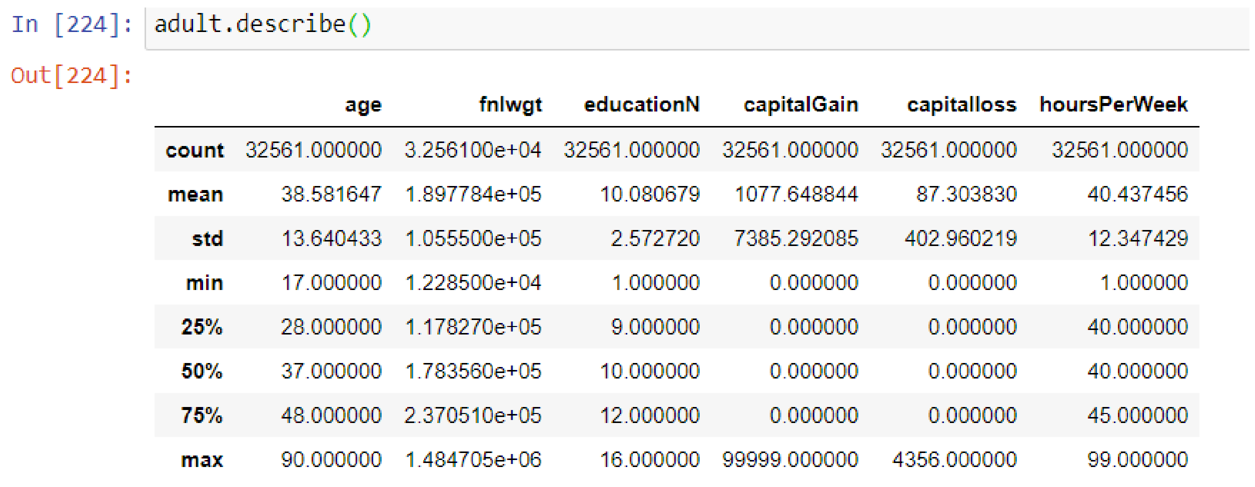
- View the list of attribute



```
list(adult)
```

```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'educationN',
 'maritalStatus',
 'occupation',
 'relationship',
 'race',
 'sex',
 'capitalGain',
 'capitalloss',
 'hoursPerWeek',
 'country',
 'salary']
```

- Describe the data : counts , standard division , mean , min and max etc..

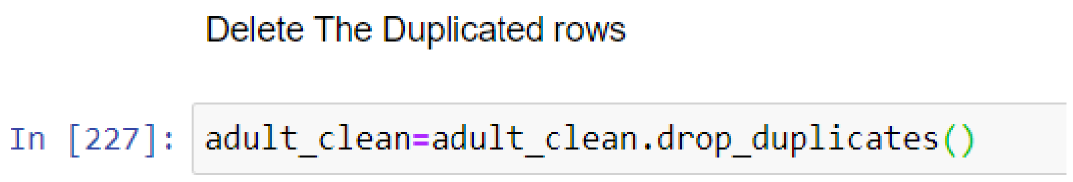


```
In [224]: adult.describe()
Out[224]:
```

	age	fnlwgt	educationN	capitalGain	capitalloss	hoursPerWeek
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

To start the cleaning in data we should get a copy from a data to working it

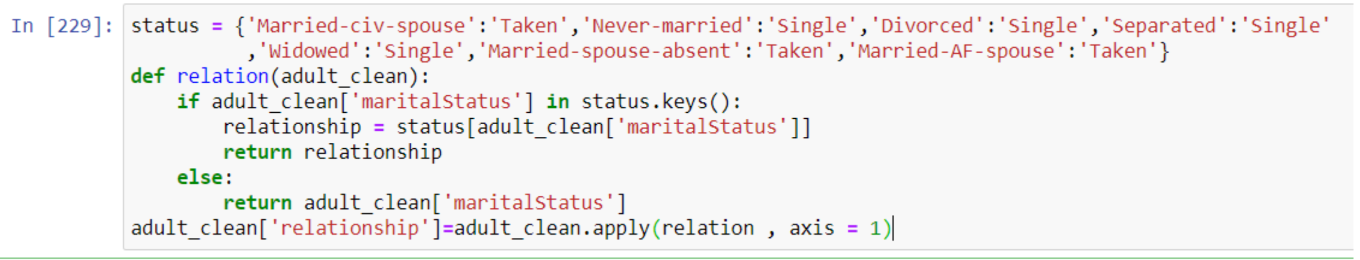
- Notice that there are repeated rows with the same values in each column and this causes data problems so should drop all duplicated rows



Delete The Duplicated rows

```
In [227]: adult_clean=adult_clean.drop_duplicates()
```

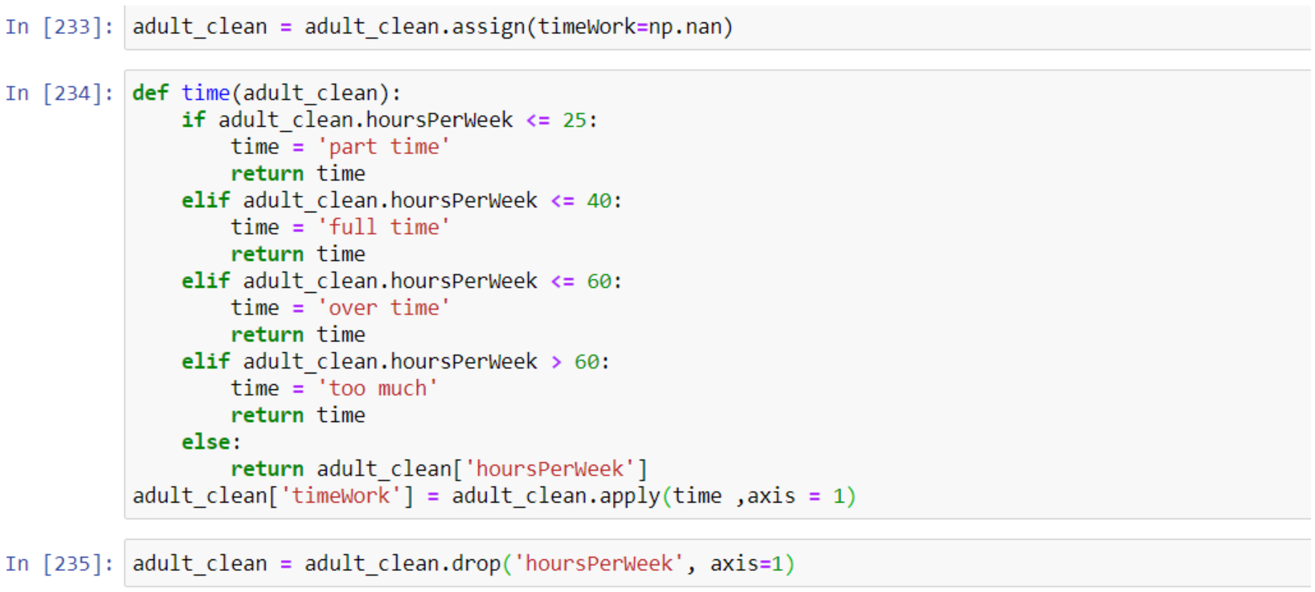
- All rows that contain unknown value in workclass also contain unknown value in occupation and this meaning that we have a noisy data so delete all rows that contain unknown value in workclass
- Notice that the data set contain a invalid values in relationship column , in some rows the marital status to someone is never-married and in relationship is own – child ?? so we fill this column by two values (single or taken) based on their marital status



```
In [229]: status = {'Married-civ-spouse':'Taken','Never-married':'Single','Divorced':'Single','Separated':'Single'
                    ,'Widowed':'Single','Married-spouse-absent':'Taken','Married-AF-spouse':'Taken'}
          def relation(adult_clean):
              if adult_clean['maritalStatus'] in status.keys():
                  relationship = status[adult_clean['maritalStatus']]
                  return relationship
              else:
                  return adult_clean['maritalStatus']
          adult_clean['relationship']=adult_clean.apply(relation , axis = 1)
```

- Then we convert the data type of sex and relationship to category and fnlwgt to float by astype() function and divide fnlwgt column into 1000 (kg)

- Create new column called timework and fill it 4 values (part time = 0 to 25 , full time = 25 to 40 , over time = 40 to 60 , too much = greater than 60) hours per week and delete hours per week column

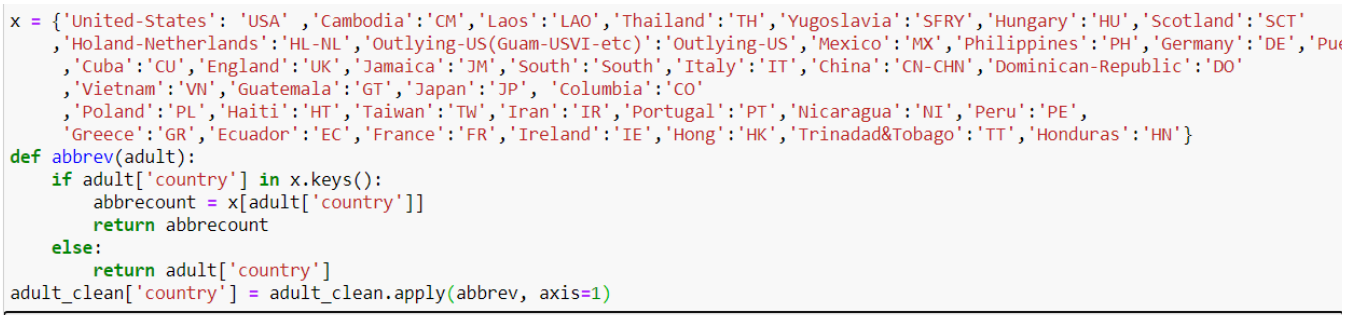


```
In [233]: adult_clean = adult_clean.assign(timeWork=np.nan)
```

```
In [234]: def time(adult_clean):
              if adult_clean.hoursPerWeek <= 25:
                  time = 'part time'
                  return time
              elif adult_clean.hoursPerWeek <= 40:
                  time = 'full time'
                  return time
              elif adult_clean.hoursPerWeek <= 60:
                  time = 'over time'
                  return time
              elif adult_clean.hoursPerWeek > 60:
                  time = 'too much'
                  return time
              else:
                  return adult_clean['hoursPerWeek']
          adult_clean['timeWork'] = adult_clean.apply(time ,axis = 1)
```

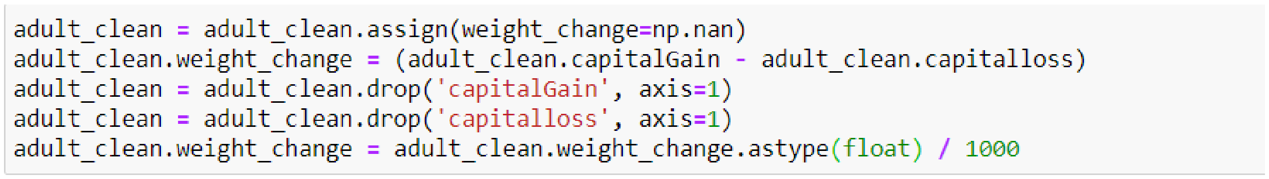
```
In [235]: adult_clean = adult_clean.drop('hoursPerWeek', axis=1)
```

- Set abbreviation to all country to be easier to read



```
x = {'United-States': 'USA' ,'Cambodia':'CM','Laos':'LAO','Thailand':'TH','Yugoslavia':'SFRY','Hungary':'HU','Scotland':'SCT'
    ,'Holand-Netherlands':'HL-NL','Outlying-US(Guam-USVI-etc)':'Outlying-US','Mexico':'MX','Philippines':'PH','Germany':'DE','Pue
     ,'Cuba':'CU','England':'UK','Jamaica':'JM','South':'South','Italy':'IT','China':'CN-CHN','Dominican-Republic':'DO'
     ,'Vietnam':'VN','Guatemala':'GT','Japan':'JP', 'Columbia':'CO'
     ,'Poland':'PL','Haiti':'HT','Taiwan':'TW','Iran':'IR','Portugal':'PT','Nicaragua':'NI','Peru':'PE',
     'Greece':'GR','Ecuador':'EC','France':'FR','Ireland':'IE','Hong':'HK','Trinadad&Tobago':'TT','Honduras':'HN'}
def abbrev(adult):
    if adult['country'] in x.keys():
        abbrecount = x[adult['country']]
        return abbrecount
    else:
        return adult['country']
adult_clean['country'] = adult_clean.apply(abbrev, axis=1)
```

- Merge capitalGain and capitalloss in one column called weight_change by subtracting capitalGain from capitalloss and if this value is negative that mean this person lost his weight , if positive then gain weight and divide into 1000 (kg)



```
adult_clean = adult_clean.assign(weight_change=np.nan)
adult_clean.weight_change = (adult_clean.capitalGain - adult_clean.capitalloss)
adult_clean = adult_clean.drop('capitalGain', axis=1)
adult_clean = adult_clean.drop('capitalloss', axis=1)
adult_clean.weight_change = adult_clean.weight_change.astype(float) / 1000
```

- From the value counts of each country we notice that :-
  1) The largest percentage of white is from USA and largest in USA
  2) JM contains black only

3) The percentage of Asian-Pac-Islander in PH is greater
4) The number of Amer-Indian-Eskimo are few and most of them are in USA

- Therefore the missing values are filled in based in This statistic

```
def CountryRace(adult_clean):
    if adult_clean.country == '?':

        if adult_clean.race == 'White':
            new_country = 'USA'
            return new_country

        elif adult_clean.race == 'Black':
            new_country = 'JM'
            return new_country

        elif adult_clean.race == 'Asian-Pac-Islander':
            new_country = 'PH'
            return new_country

        elif adult_clean.race == 'Amer-Indian-Eskimo':
            new_country = 'USA'
            return new_country

        else:
            new_country = 'USA'
            return new_country
    else:
        return adult_clean['country']
adult_clean['country'] = adult_clean.apply(CountryRace ,axis = 1)
```

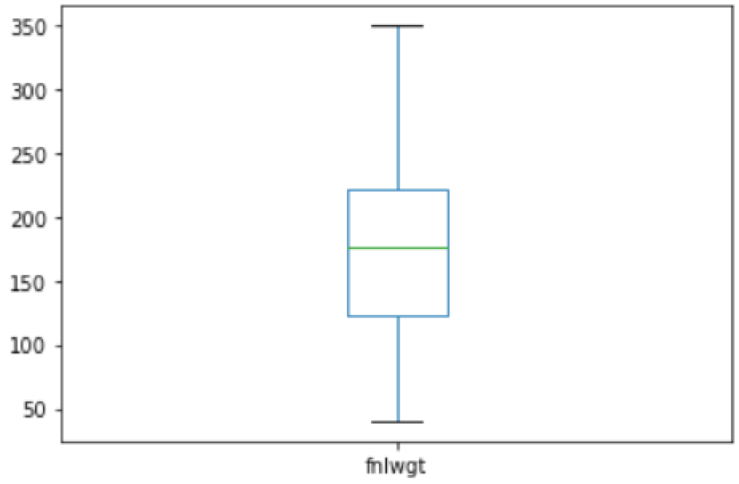
- The largest human weight in the world was 350 Kg and it does not make sense that an adult weighs less than 40 Kg so we delete this invalid values

```
adult_clean = adult_clean[adult_clean.fnlwgt >= 40]
adult_clean = adult_clean[adult_clean.fnlwgt <= 350]
```

- Notice that no outliers in fnlwgt

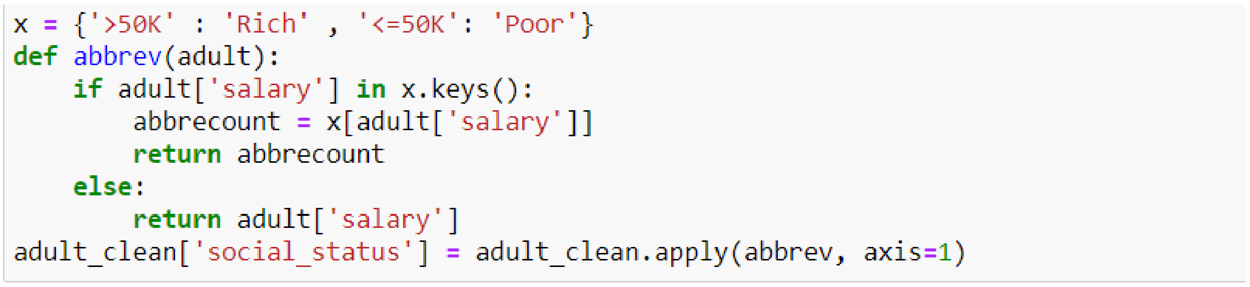
```
adult_clean.fnlwgt.plot(kind = "box")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x159260db100>
```



-

- Delete the missing values in occupation
- Create new column called social_status and fills it two values poor and rich based on their salary (less than or equal 50k = poor , greater than 50k = rich)
-



```
x = {'>50K' : 'Rich' , '<=50K': 'Poor'}
def abbrev(adult):
    if adult['salary'] in x.keys():
        abbrecount = x[adult['salary']]
        return abbrecount
    else:
        return adult['salary']
adult_clean['social_status'] = adult_clean.apply(abbrev, axis=1)
```

- Finally we store the clean data as csv file