Project name - Census_income_data

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')                                          # importing python librarie
```

In [2]:

```
df = pd.read_csv("C:/Users/harshitagups/Desktop/project/census_income_data.csv")  # importing dataset
```

In [3]:

```
print('Rows: {} Columns: {}'.format(df.shape[0], df.shape[1]))             # defines (rows,columns)
```

```
Rows: 32561 Columns: 15
```

In [4]:

```
df.head(11)                                                                #first 10 data entry from d
```

Out[4]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.g
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	
7	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	
8	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	
9	41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	
10	45	Private	172274	Doctorate	16	Divorced	Prof-specialty	Unmarried	Black	Female	

In [5]:

```
df.tail(15)                                                           #last 15 data entry from da
```

Out[5]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	ca
32546	31	Private	199655	Masters	14	Divorced	Other-service	Not-in-family	Other	Female	
32547	39	Local-gov	111499	Assoc-acdm	12	Married-civ-spouse	Adm-clerical	Wife	White	Female	
32548	37	Private	198216	Assoc-acdm	12	Divorced	Tech-support	Not-in-family	White	Female	
32549	43	Private	260761	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	
32550	43	State-gov	255835	Some-college	10	Divorced	Adm-clerical	Other-relative	White	Female	
32551	43	Self-emp-not-inc	27242	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	
32552	32	Private	34066	10th	6	Married-civ-spouse	Handlers-cleaners	Husband	Amer-Indian-Eskimo	Male	
32553	43	Private	84661	Assoc-voc	11	Married-civ-spouse	Sales	Husband	White	Male	
32554	32	Private	116138	Masters	14	Never-married	Tech-support	Not-in-family	Asian-Pac-Islander	Male	
32555	53	Private	321865	Masters	14	Married-civ-spouse	Exec-managerial	Husband	White	Male	
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Male	
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	

In [6]:

```
df.info()                                                   #all information regarding dataset like datatypes,
#Observations:
#1.There are in total 32561 samples in the census_income data set
#2.There are both categorical and numerical attributes in the dataset
```

```
<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   education.num   32561 non-null  int64 
 5   marital.status  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  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [7]:

```
df.nunique()                                                        #finding out no. of unique values in part
```

Out[7]:

```
age                  73
workclass             9
fnlwgt            21648
education            16
education.num        16
marital.status        7
occupation           15
relationship          6
race                  5
sex                   2
capital.gain        119
capital.loss         92
hours.per.week       94
native.country       42
income                2
dtype: int64
```

In [8]:

```
for i, col in enumerate(df.columns):
    print(df.columns[i],":", df[str(col)].unique(), '\n')                     #Unique values in e
```

```
age : [90 82 66 54 41 34 38 74 68 45 52 32 51 46 57 22 37 29 61 21 33 49 23 59
 60 63 53 44 43 71 48 73 67 40 50 42 39 55 47 31 58 62 36 72 78 83 26 70
 27 35 81 65 25 28 56 69 20 30 24 64 75 19 77 80 18 17 76 79 88 84 85 86
 87]

workclass : ['?' 'Private' 'State-gov' 'Federal-gov' 'Self-emp-not-inc' 'Self-emp-inc'
 'Local-gov' 'Without-pay' 'Never-worked']

fnlwgt : [ 77053 132870 186061 ...  34066  84661 257302]

education : ['HS-grad' 'Some-college' '7th-8th' '10th' 'Doctorate' 'Prof-school'
 'Bachelors' 'Masters' '11th' 'Assoc-acdm' 'Assoc-voc' '1st-4th' '5th-6th'
 '12th' '9th' 'Preschool']

education.num : [ 9 10  4  6 16 15 13 14  7 12 11  2  3  8  5  1]

marital.status : ['Widowed' 'Divorced' 'Separated' 'Never-married' 'Married-civ-spouse'
 'Married-spouse-absent' 'Married-AF-spouse']

occupation : ['?' 'Exec-managerial' 'Machine-op-inspct' 'Prof-specialty'
 'Other-service' 'Adm-clerical' 'Craft-repair' 'Transport-moving'
 'Handlers-cleaners' 'Sales' 'Farming-fishing' 'Tech-support'
 'Protective-serv' 'Armed-Forces' 'Priv-house-serv']

relationship : ['Not-in-family' 'Unmarried' 'Own-child' 'Other-relative' 'Husband' 'Wife']

race : ['White' 'Black' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']

sex : ['Female' 'Male']

capital.gain : [    0 99999 41310 34095 27828 25236 25124 22040 20051 18481 15831 15024
 15020 14344 14084 13550 11678 10605 10566 10520  9562  9386  8614  7978
  7896  7688  7443  7430  7298  6849  6767  6723  6514  6497  6418  6360
  6097  5721  5556  5455  5178  5060  5013  4934  4931  4865  4787  4687
  4650  4508  4416  4386  4101  4064  3942  3908  3887  3818  3781  3674
  3471  3464  3456  3432  3418  3411  3325  3273  3137  3103  2993  2977
  2964  2961  2936  2907  2885  2829  2653  2635  2597  2580  2538  2463
  2414  2407  2387  2354  2346  2329  2290  2228  2202  2176  2174  2105
  2062  2050  2036  2009  1848  1831  1797  1639  1506  1471  1455  1424
  1409  1173  1151  1111  1086  1055   991   914   594   401   114]

capital.loss : [4356 3900 3770 3683 3004 2824 2754 2603 2559 2547 2489 2472 2467 2457
 2444 2415 2392 2377 2352 2339 2282 2267 2258 2246 2238 2231 2206 2205
 2201 2179 2174 2163 2149 2129 2080 2057 2051 2042 2002 2001 1980 1977
 1974 1944 1902 1887 1876 1848 1844 1825 1816 1762 1755 1741 1740 1735
 1726 1721 1719 1672 1669 1668 1651 1648 1628 1617 1602 1594 1590 1579
 1573 1564 1539 1504 1485 1411 1408 1380 1340 1258 1138 1092  974  880
  810  653  625  419  323  213  155    0]

hours.per.week : [40 18 45 20 60 35 55 76 50 42 25 32 90 48 15 70 52 72 39  6 65 12 80 67
 99 30 75 26 36 10 84 38 62 44  8 28 59  5 24 57 34 37 46 56 41 98 43 63
  1 47 68 54  2 16  9  3  4 33 23 22 64 51 19 58 53 96 66 21  7 13 27 11
 14 77 31 78 49 17 85 87 88 73 89 97 94 29 82 86 91 81 92 61 74 95]

native.country : ['United-States' '?' 'Mexico' 'Greece' 'Vietnam' 'China' 'Taiwan' 'India'
 'Philippines' 'Trinadad&Tobago' 'Canada' 'South' 'Holand-Netherlands'
 'Puerto-Rico' 'Poland' 'Iran' 'England' 'Germany' 'Italy' 'Japan' 'Hong'
 'Honduras' 'Cuba' 'Ireland' 'Cambodia' 'Peru' 'Nicaragua'
 'Dominican-Republic' 'Haiti' 'El-Salvador' 'Hungary' 'Columbia'
 'Guatemala' 'Jamaica' 'Ecuador' 'France' 'Yugoslavia' 'Scotland'
 'Portugal' 'Laos' 'Thailand' 'Outlying-US(Guam-USVI-etc)']

income : ['<=50K' '>50K']
```

In [9]:

```
pd.isnull(df).sum()                                                    # Check for Null Data
```

Out[9]:

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

numerical attributes

In [10]:

```
num_attributes = df.select_dtypes(include=['int'])
print(num_attributes.columns)                                    # Identify Numeric features
```

```
Index(['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss',
       'hours.per.week'],
      dtype='object')
```

In [11]:

```
num_attributes.describe()                          # describe about the numerical columns( like mean,
```

Out[11]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
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

categorical_attributes

In [12]:

```
categorical_attributes = df.select_dtypes(include=['object'])
print(categorical_attributes.columns)                                  # Identify Categorical
```

```
Index(['workclass', 'education', 'marital.status', 'occupation',
       'relationship', 'race', 'sex', 'native.country', 'income'],
      dtype='object')
```

In [13]:

```
categorical_attributes.describe()
```

Out[13]:

	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
count	32561	32561	32561	32561	32561	32561	32561	32561	32561
unique	9	16	7	15	6	5	2	42	2
top	Private	HS-grad	Married-civ-spouse	Prof-specialty	Husband	White	Male	United-States	<=50K
freq	22696	10501	14976	4140	13193	27816	21790	29170	24720

encoded number for each field

In [14]:

```
df.workclass.value_counts()
```

Out[14]:

```
Private             22696
Self-emp-not-inc     2541
Local-gov            2093
?                    1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

In [15]:

```
df.groupby('education')['education.num'].unique().sort_values()
# Education.num shows the no from 1 to 16 on the education basis, We are getting the same info from the educat
```

Out[15]:

```
education
Preschool        [1]
1st-4th          [2]
5th-6th          [3]
7th-8th          [4]
9th              [5]
10th             [6]
11th             [7]
12th             [8]
HS-grad          [9]
Some-college    [10]
Assoc-voc       [11]
Assoc-acdm      [12]
Bachelors       [13]
Masters         [14]
Prof-school     [15]
Doctorate       [16]
Name: education.num, dtype: object
```

In [16]:

```
df['marital.status'].value_counts()
```

Out[16]:

```
Married-civ-spouse       14976
Never-married            10683
Divorced                  4443
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: marital.status, dtype: int64
```

In [17]:

```
df.occupation.value_counts()
```

Out[17]:

```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: occupation, dtype: int64
```

In [18]:

```
df.relationship.value_counts()
```

Out[18]:

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

In [19]:

```
df.race.value_counts()
```

Out[19]:

```
White                 27816
Black                  3124
Asian-Pac-Islander     1039
Amer-Indian-Eskimo      311
Other                   271
Name: race, dtype: int64
```

In [20]:

```
df.sex.value_counts()
```

Out[20]:

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

In [21]:

```
df['native.country'].value_counts()
```

Out[21]:

```
United-States                 29170
Mexico                          643
?                               583
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
Greece                           29
France                           29
Ecuador                          28
Ireland                          24
Hong                             20
Cambodia                         19
Trinadad&Tobago                  19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Hungary                          13
Honduras                         13
Scotland                         12
Holand-Netherlands                1
Name: native.country, dtype: int64
```

In [22]:

```
df.income.value_counts()
```

Out[22]:

```
<=50K    24720
>50K      7841
Name: income, dtype: int64
```

In [23]:

```
df.isin(['?']).sum(axis=0)                              # finding out total number of "?" sy
```

Out[23]:

```
age                  0
workclass         1836
fnlwgt               0
education            0
education.num        0
marital.status       0
occupation        1843
relationship         0
race                 0
sex                  0
capital.gain         0
capital.loss         0
hours.per.week       0
native.country     583
income               0
dtype: int64
```

In [24]:

```
df.groupby(['income', 'occupation']).size()
```

Out[24]:

```
income  occupation
<=50K   ?                    1652
        Adm-clerical         3263
        Armed-Forces            8
        Craft-repair         3170
        Exec-managerial      2098
        Farming-fishing       879
        Handlers-cleaners    1284
        Machine-op-inspct    1752
        Other-service        3158
        Priv-house-serv       148
        Prof-specialty       2281
        Protective-serv       438
        Sales                2667
        Tech-support          645
        Transport-moving     1277
>50K    ?                     191
        Adm-clerical          507
        Armed-Forces            1
        Craft-repair          929
        Exec-managerial      1968
        Farming-fishing       115
        Handlers-cleaners      86
        Machine-op-inspct     250
        Other-service         137
        Priv-house-serv         1
        Prof-specialty       1859
        Protective-serv       211
        Sales                 983
        Tech-support          283
        Transport-moving      320
dtype: int64
```

Reformating Column

converting dependent col into categor- assuming <=50K - 0, >50K - 1

In [25]:

```
df['income'] = df['income'].map({'<=50K':0, '>50K':1})
#df['income'].replace({'<=50K':0, '>50K':1}, inplace=True)
df.head(10)
```

Out[25]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.ga
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	
7	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	
8	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	
9	41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	

Event rate

In [26]:

```
df['income'].value_counts(normalize=True)*100
```

Out[26]:

```
0    75.919044
1    24.080956
Name: income, dtype: float64
```

In [27]:

```
df.groupby(['income', 'sex']).size()
```

Out[27]:

```
income  sex
0       Female     9592
        Male      15128
1       Female     1179
        Male       6662
dtype: int64
```

Outliers detection

Q1, Q3, IQR, LB, UB

In [28]:

```
df.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   education.num   32561 non-null  int64 
 5   marital.status  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  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  int64 
dtypes: int64(7), object(8)
memory usage: 3.7+ MB
```

In [29]:

```
col_for_outliers=['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss', 'hours.per.week']
```

In [30]:

```
summary_pre_outliers_detection = df[col_for_outliers].describe()
```

In [31]:

```
summary_pre_outliers_detection
```

Out[31]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
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

In [32]:

```
for i in col_for_outliers:
    Q1 = np.percentile(df[i], 25)
    Q3 = np.percentile(df[i], 75)
```

In [33]:

```
IQR = Q3-Q1
```

In [34]:

```
LB = Q1-1.5*IQR
```

In [35]:

```
UB = Q3+1.5*IQR
```

In [36]:

```
df[i] = np.where(df[i] < LB, LB, df[i])
```

In [37]:

```
df[i] = np.where(df[i] > UB, UB, df[i])
```

In [38]:

```
summary_post_outliers_detection = df[col_for_outliers].describe()
```

In [39]:

```
summary_post_outliers_detection
```

Out[39]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
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	41.202451
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	6.187005
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	32.500000
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	52.500000

In [40]:

```
df.head(10)
```

Out[40]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.ga
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	
7	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	
8	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	
9	41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	

Univariate Anlysis

For col workclass and occupation we are replacing ? with never-worked

In [41]:

```
#print(df.replace("?",np.nan, inplace = True))              # this will replace the value which is on the
```

In [42]:

```
#(Univariate analysis)
df['workclass']=df['workclass'].replace(to_replace="?",value="never-worked")
df.workclass.value_counts()                          # this will replace the '?' from workclass to Unemployed
```

Out[42]:

```
Private             22696
Self-emp-not-inc     2541
Local-gov            2093
never-worked         1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

In [43]:

```
#df['workclass'].replace(to_replace = ['?','Self-emp-not-inc','Without-pay','Never-worked'], value = 'no-incor
#df['workclass'].replace(to_replace = ['Local-gov','State-gov','Federal-gov'], value = 'gov',inplace = True)
#df['workclass'].replace(to_replace = 'Self-emp-inc', value = 'Self', inplace = True)
```

In [44]:

```
df.head()
```

Out[44]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.ga
0	90	never-worked	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	
2	66	never-worked	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	

In [45]:

```
df['occupation']=df['occupation'].replace(to_replace="?",
                    value="never-worked")                                    #replacing '?' to Unemploye

#df['occupation'].replace(to_replace = ['?','Other-service'], value = 'Other', inplace = True)
df.occupation.value_counts()
```

Out[45]:

```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
never-worked         1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: occupation, dtype: int64
```

In [46]:

```
df.head()
```

Out[46]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.ga
0	90	never-worked	77053	HS-grad	9	Widowed	never-worked	Not-in-family	White	Female	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	
2	66	never-worked	186061	Some-college	10	Widowed	never-worked	Unmarried	Black	Female	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	

In [47]:

```
df['native.country'].value_counts()
```

Out[47]:

```
United-States                 29170
Mexico                          643
?                               583
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
Greece                           29
France                           29
Ecuador                          28
Ireland                          24
Hong                             20
Cambodia                         19
Trinadad&Tobago                  19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Hungary                          13
Honduras                         13
Scotland                         12
Holand-Netherlands                1
Name: native.country, dtype: int64
```

In native.country we are replacing ? with mode

In [48]:

```
temp_mode = df['native.country'].mode()[0]
df['native.country'] = df['native.country'].replace('?',temp_mode)
```

United states appearing maximum no of times(29k), we can segregate the column in 2 parts United states & others take other countries as one entity.

In [49]:

```
# Loop for seggregate the native.country into US and Other

col = df['native.country']

for i in col:
    print(i)
    if i!= 'United-States':
        df['native.country']= df['native.country'].replace({i:'others'})
```

```
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
United-States
```

In [50]:

```
df['native.country'].value_counts()
```

Out[50]:

```
United-States    29753
others            2808
Name: native.country, dtype: int64
```

In [51]:

```
df.head()
```

Out[51]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.ga
0	90	never-worked	77053	HS-grad	9	Widowed	never-worked	Not-in-family	White	Female	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	
2	66	never-worked	186061	Some-college	10	Widowed	never-worked	Unmarried	Black	Female	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	

In [52]:

```
#pd.crosstab(raw['native.country'], raw['income'],
#values=raw['hours.per.week'], aggfunc=np.mean)
```

In [53]:

```
df.isnull().sum()
```

Out[53]:

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

In [54]:

```
df.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   education.num   32561 non-null  int64  
 5   marital.status  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  capital.gain    32561 non-null  int64  
 11  capital.loss    32561 non-null  int64  
 12  hours.per.week  32561 non-null  float64
 13  native.country  32561 non-null  object 
 14  income          32561 non-null  int64  
dtypes: float64(1), int64(6), object(8)
memory usage: 3.7+ MB
```

In [55]:

```
df.isnull().sum()
```

Out[55]:

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

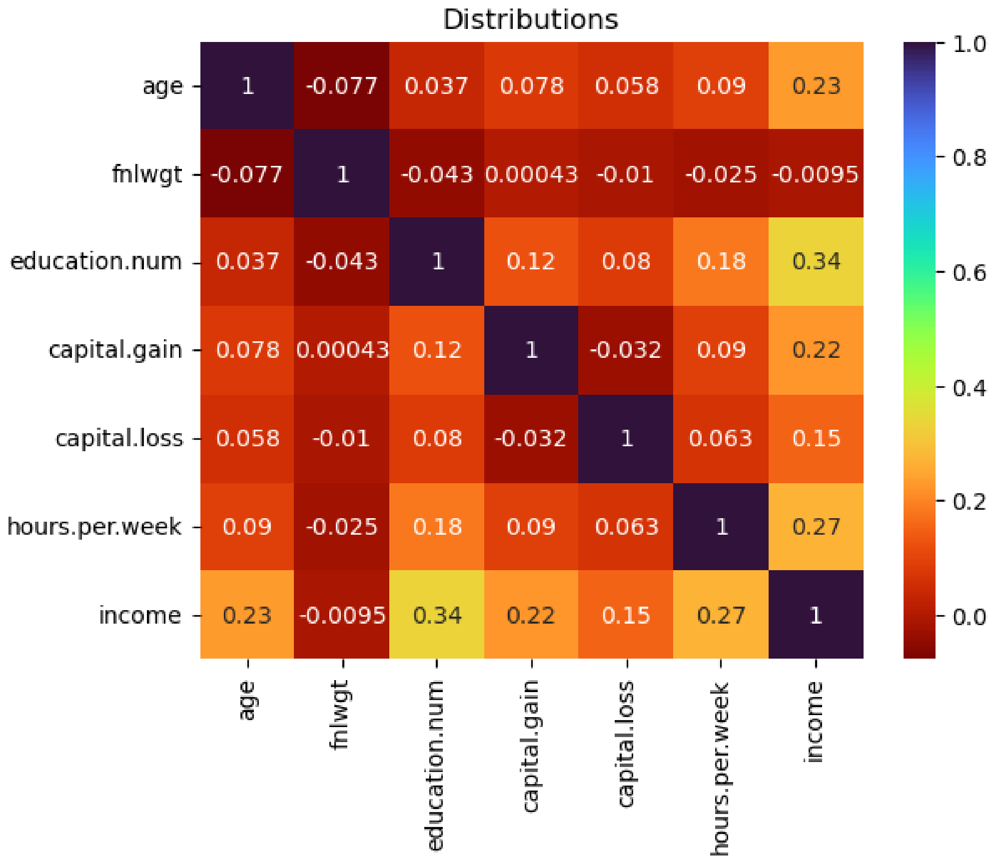
Bivariate analysis for continuous variables

In [56]:

```
corr_df = df.corr()
sns.heatmap(corr_df, xticklabels=corr_df,  yticklabels=corr_df, cmap='turbo_r', annot=True)
plt.title('Distributions')
```

Out[56]:

```
Text(0.5, 1.0, 'Distributions')
```

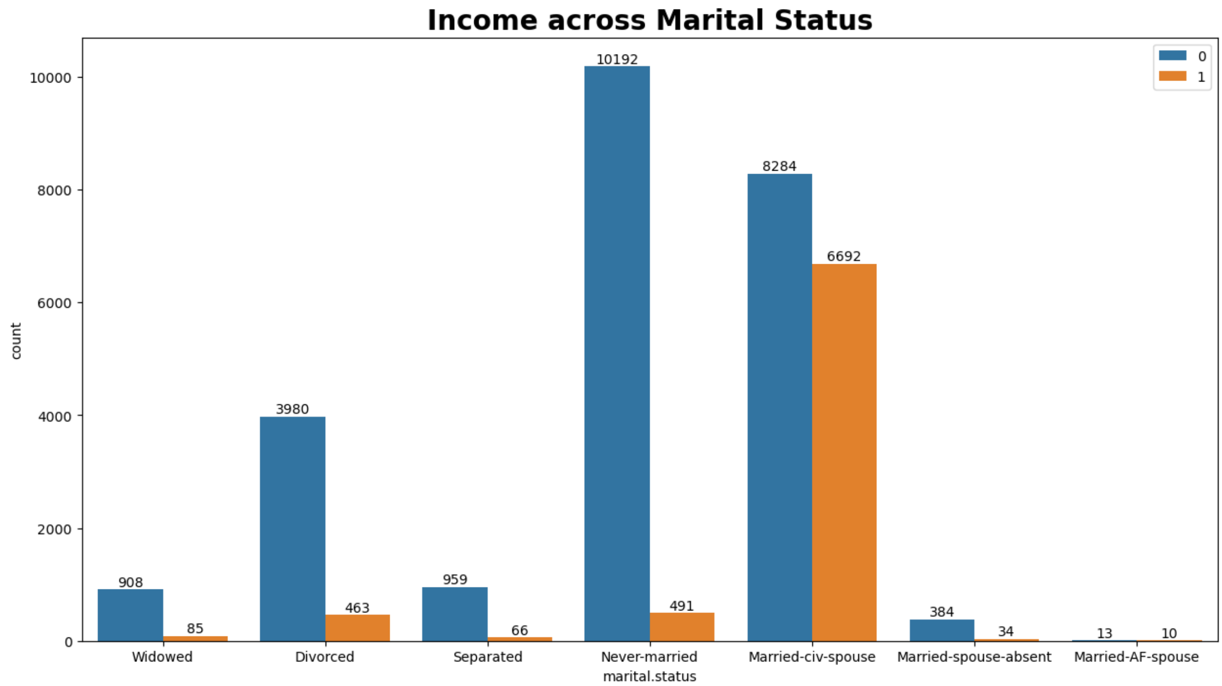


In [57]:

```
# Creating a countplot of income across Marital Status
plt.figure(figsize=(15,8))
graph1=sns.countplot(x='marital.status',hue='income',data=df)

for i in graph1.containers:
    graph1.bar_label(i)
plt.legend(loc='upper right')
plt.title("Income across Marital Status", fontdict={'fontsize': 20, 'fontweight': 'bold'})

#Based on the graph analysis it is clear that Never-married peoples salary is less than 50K but more in total
#and same for marries-civ-spouse
```

Out[57]:

```
Text(0.5, 1.0, 'Income across Marital Status')
```

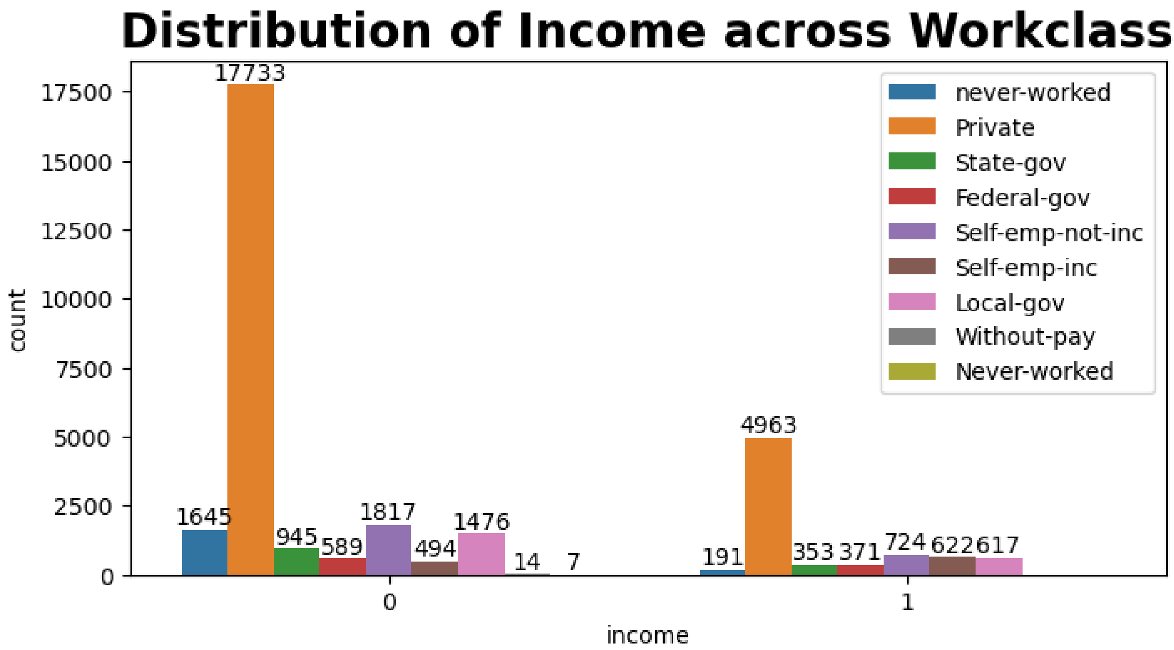


In [58]:

```
# Creating a countplot of income across workclass
plt.figure(figsize=(8,4))
graph3=sns.countplot(x='income',hue='workclass',data=df)
for i in graph3.containers:
    graph3.bar_label(i)
plt.legend(loc='upper right')
plt.title('Distribution of Income across Workclass', fontdict={'fontsize': 20, 'fontweight': 'bold'})
#From the 'Workclass' countplot graph it can be analyse that most of the people are working privately with
                    #less than 50K income.
```

Out[58]:

```
Text(0.5, 1.0, 'Distribution of Income across Workclass')
```

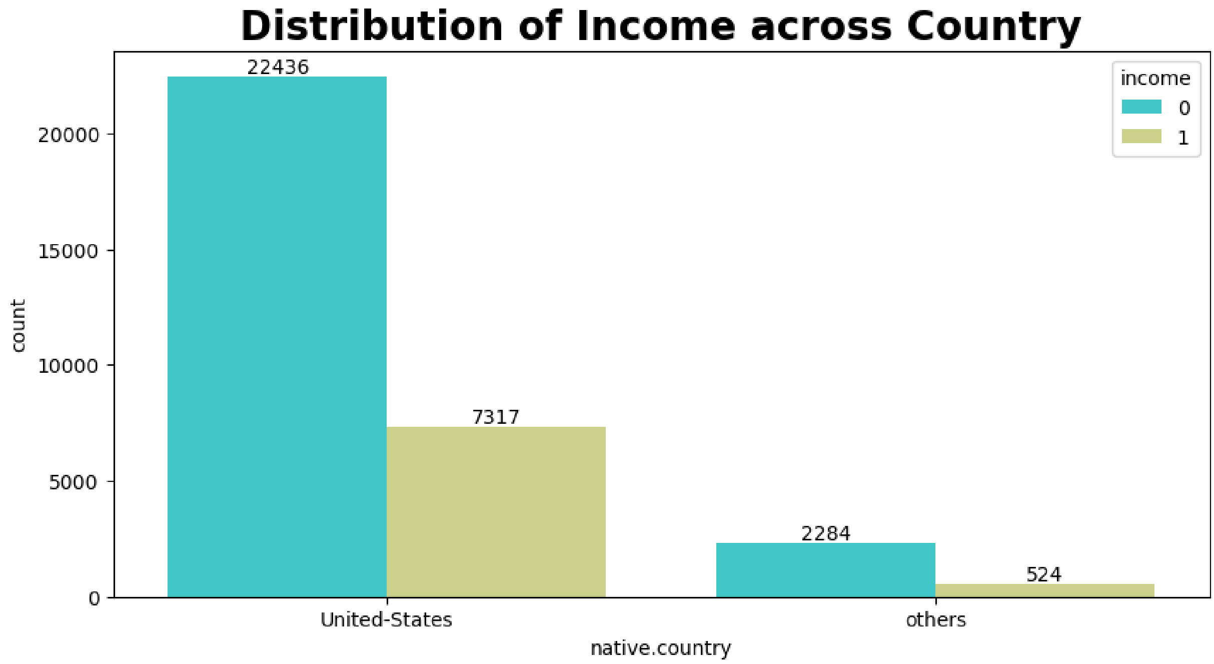


In [59]:

```
# Creating a countplot of income across country
plt.figure(figsize=(10,5))
graph4=sns.countplot(x="native.country", hue="income", data=df,palette='rainbow')
for i in graph4.containers:
    graph4.bar_label(i)
plt.title('Distribution of Income across Country', fontdict={'fontsize': 20, 'fontweight': 'bold'})
#From the 'native country' countplot graph it can be analyse that maximum number of people working in united
                   #and had a income <=50K
```

Out[59]:

```
Text(0.5, 1.0, 'Distribution of Income across Country')
```

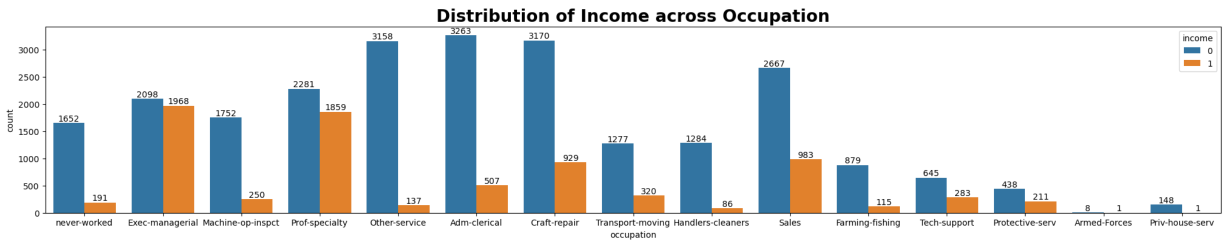


In [60]:

```
# Creating a countplot of income across occupation
plt.figure(figsize=(25,4))
graph=sns.countplot(data=df, x='occupation', hue='income')
for i in graph.containers:
    graph.bar_label(i)
plt.title('Distribution of Income across Occupation', fontdict={'fontsize': 20, 'fontweight': 'bold'})
#From the 'Occupation' countplot graph it can be analyse that most of the peoples work as Adm-Clerical, Craft
```

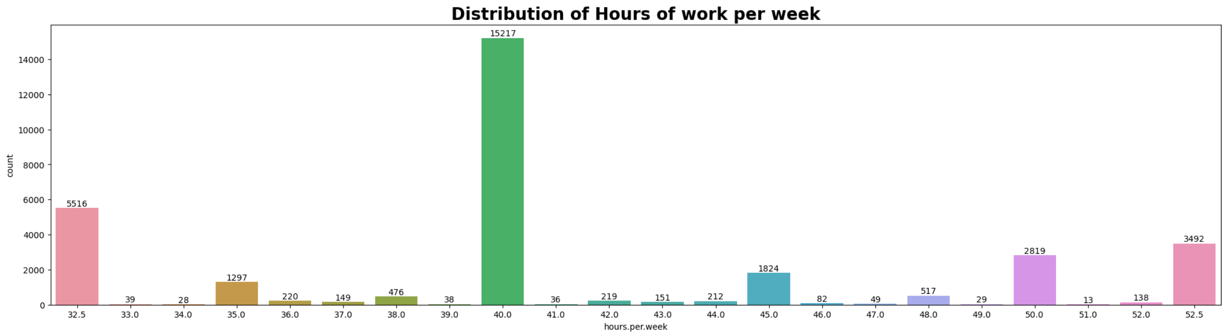
Out[60]:

```
Text(0.5, 1.0, 'Distribution of Income across Occupation')
```



In [61]:

```
# Creating a countplot for 'Hours per week'
plt.figure(figsize=(25,6))
graph=sns.countplot(x="hours.per.week",data=df)
for i in graph.containers:
    graph.bar_label(i)
    plt.title('Distribution of Hours of work per week', fontdict={'fontsize': 20, 'fontweight': 'bold'})
# it is graph from the graph that majority of the people are working 40 hours in a week.
```

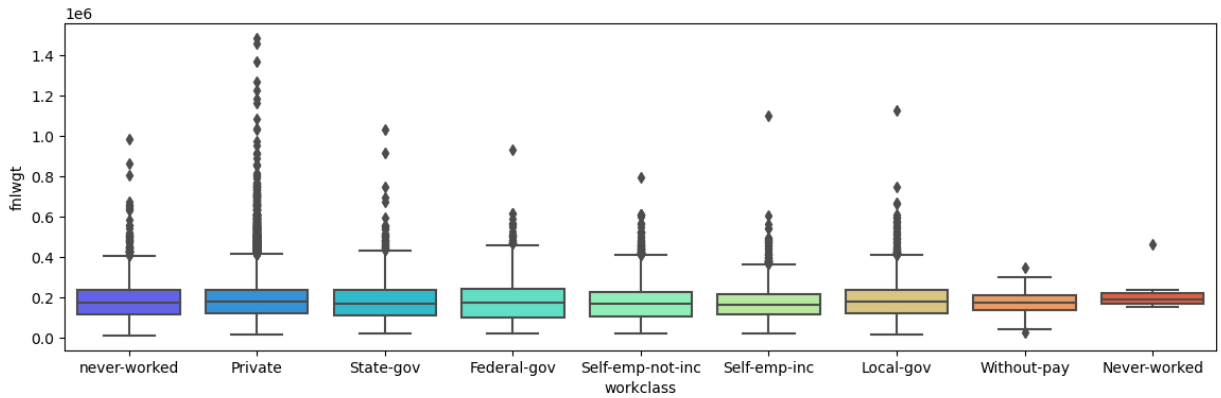


In [62]:

```
# Creating a boxplot for workclass
plt.figure(figsize=(14,4))
sns.boxplot(x="workclass", y="fnlwgt", data=df,palette='rainbow')
#Outliers present in all the workclass w.r.t final weight
```

Out[62]:

<AxesSubplot:xlabel='workclass', ylabel='fnlwgt'>

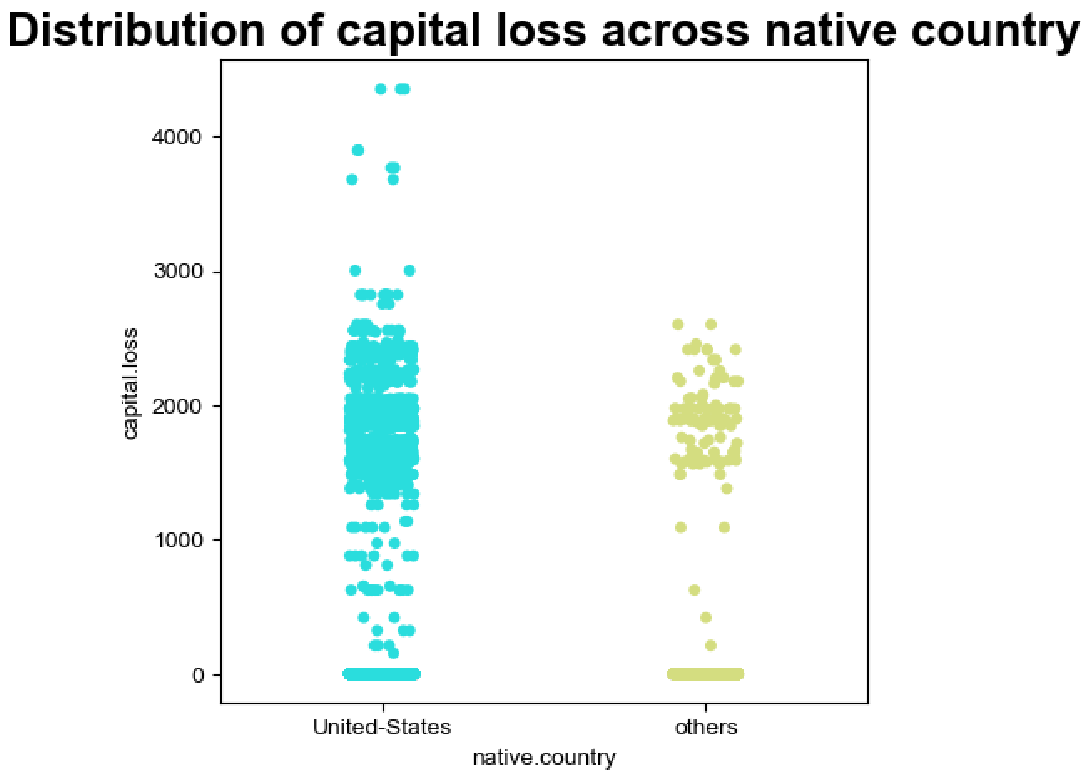


In [63]:

```
# Creating a strip plot of native country accross capital loss
plt.figure(figsize=(5,5))
sns.stripplot(x="native.country", y="capital.loss", data=df,palette='rainbow')
sns.set(rc={'figure.figsize':(30,20)})
plt.title('Distribution of capital loss across native country', fontdict={'fontsize': 20, 'fontweight': 'bold
#capital loss is highest in united states
```

Out[63]:

```
Text(0.5, 1.0, 'Distribution of capital loss across native country')
```

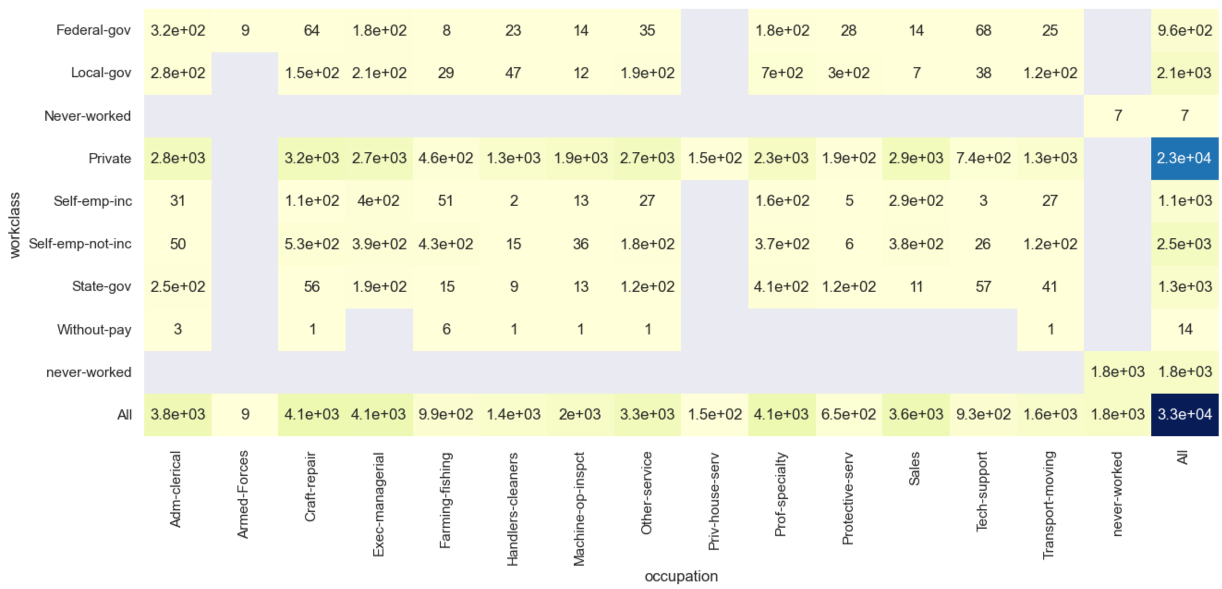


In [64]:

```
# Creating a heat map of occupation accross workclass
plt.figure(figsize=(15,6))
sns.heatmap(pd.crosstab(df.workclass, df.occupation, margins=True, values=df.income, aggfunc=pd.Series.count)
            annot=True, cbar=False)
```

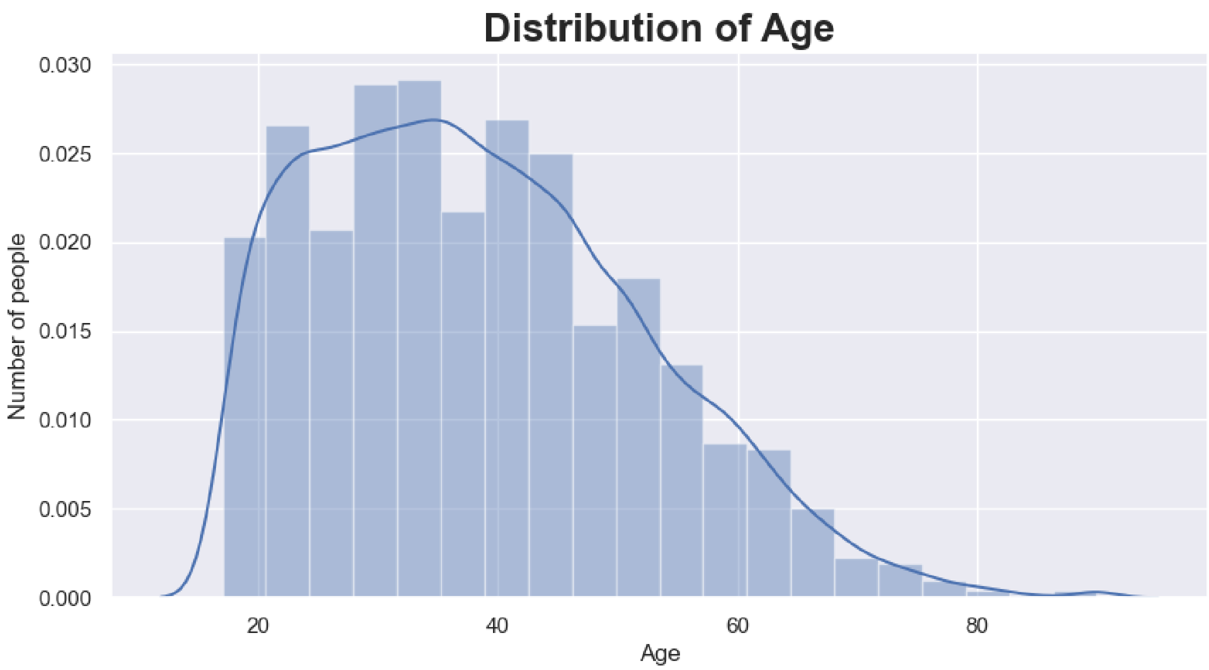
Out[64]:

```
<AxesSubplot:xlabel='occupation', ylabel='workclass'>
```



In [65]:

```
# Creating a distribution plot for 'Age'
age = df['age'].value_counts()
plt.figure(figsize=(10, 5))
sns.distplot(df['age'], bins=20)
plt.title('Distribution of Age', fontdict={'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Age')
plt.ylabel('Number of people')
plt.show()
#From the graph shown below it can be predicted that people of age 25-45 are more in population.
```

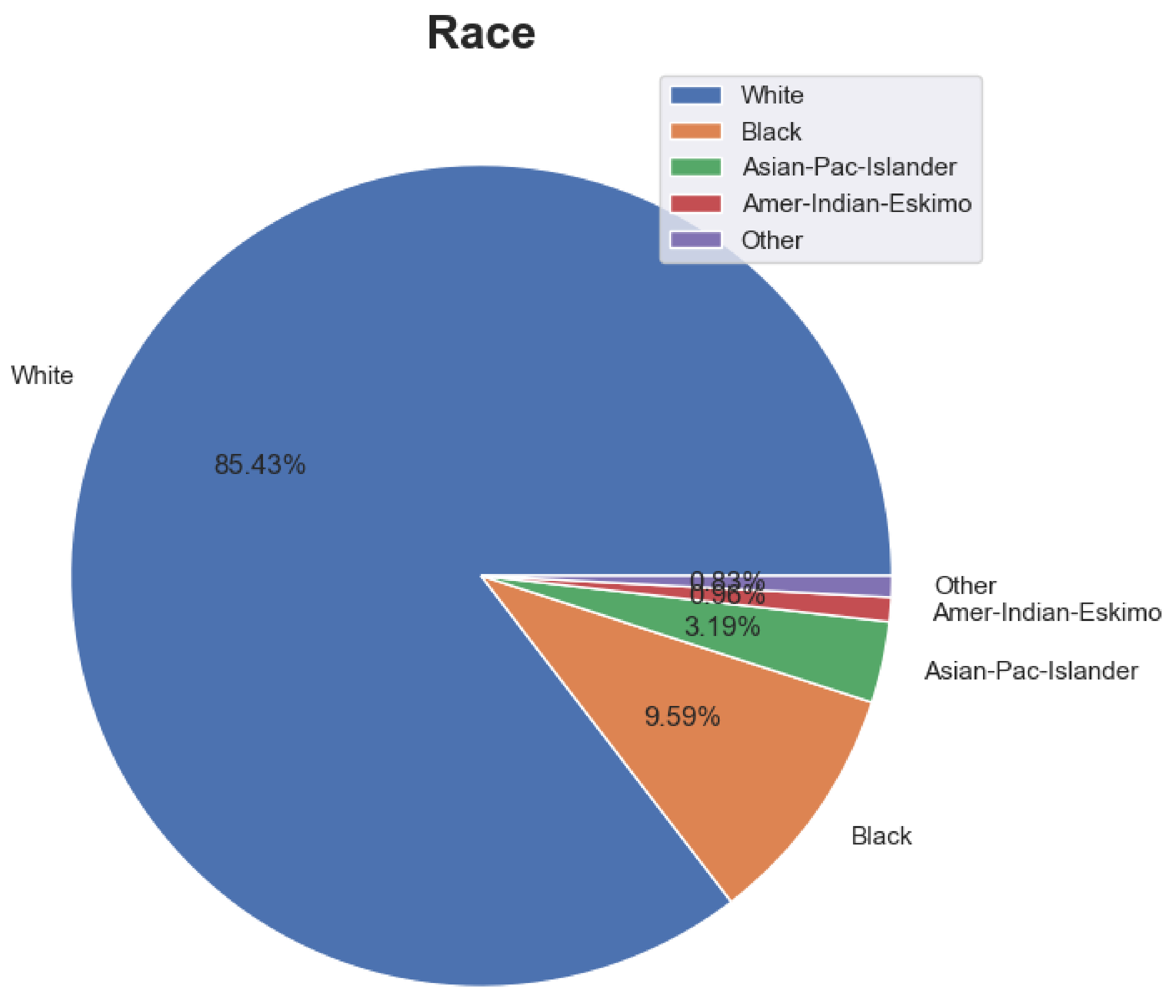


In [66]:

```
# Creating a pie chart for 'Race'
plt.figure(figsize=(8,8))
labels = df['race'].value_counts().index
values = df['race'].value_counts().values
colors = df['race']
plt.pie(values, labels=labels, autopct="%1.2f%%")
plt.title('Race', fontdict={'fontsize': 20, 'fontweight': 'bold'})
plt.legend()
#based on the graph analysis it is clear that 85.43% people are white
```

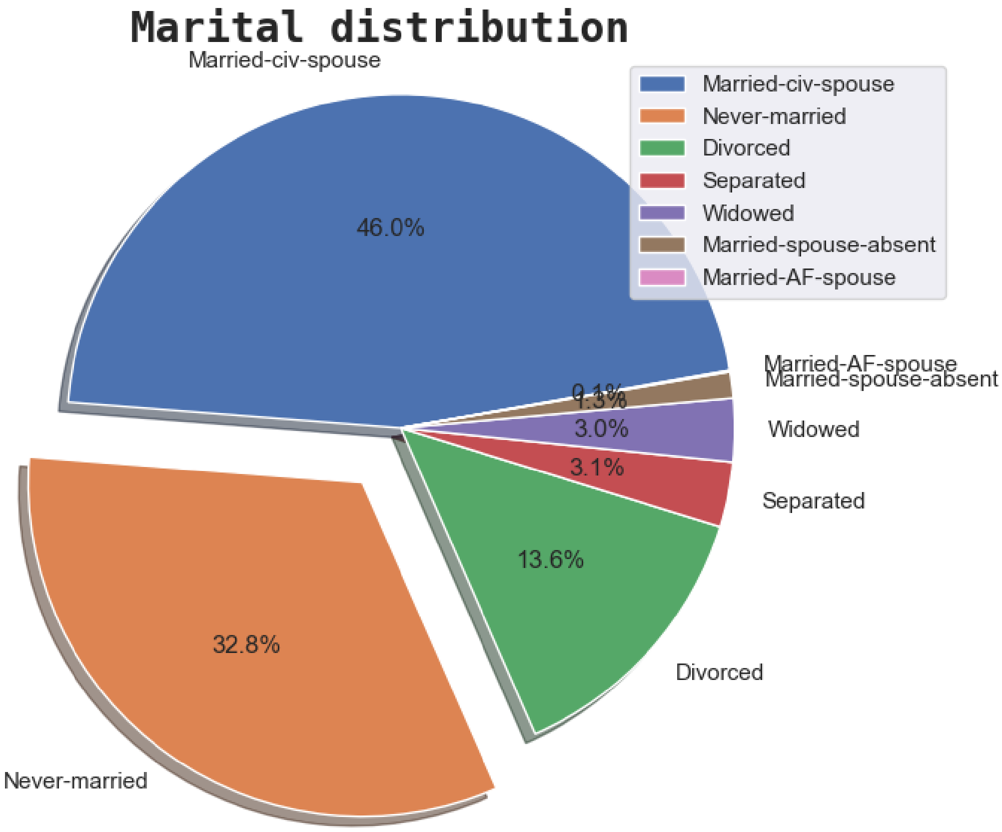
Out[66]:

```
<matplotlib.legend.Legend at 0x172fe911ca0>
```



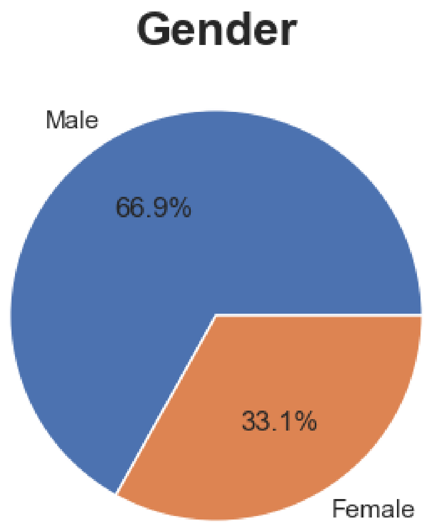
In [67]:

```
# Creating a pie chart for 'Marital status'
marital = df['marital.status'].value_counts()
plt.figure(figsize=(10, 7))
plt.pie(marital.values, labels=marital.index, startangle=10, explode=(0, 0.20, 0, 0, 0, 0, 0), shadow=True, a
plt.title('Marital distribution', fontdict={'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.legend()
plt.axis('equal')
plt.show()
#Based on the graph analysis it can be conclude that Mostly population are Married-civ-spouse
```



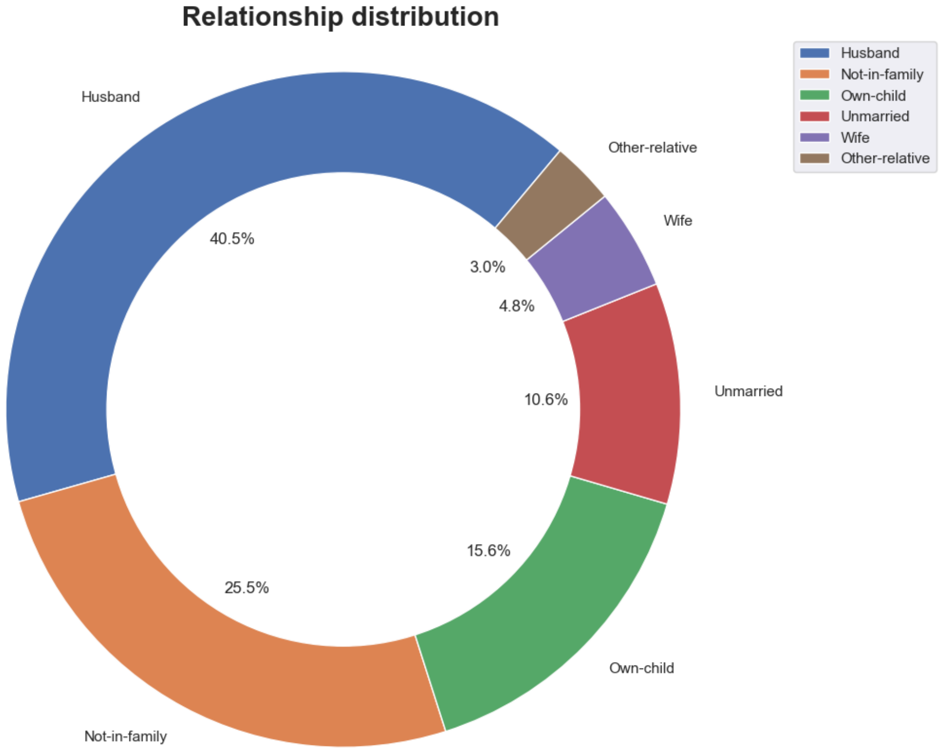
In [68]:

```
# Creating a pie chart for 'Gender'
label=df.sex.value_counts().index
count=df.sex.value_counts().values
plt.figure(1, figsize=(4,4))
plt.pie(count,labels=label,autopct='%1.1f%%')
plt.title('Gender', fontdict={'fontsize': 20, 'fontweight': 'bold'})
plt.show()
#Based on the graph analysis it is clear that males percentage is more then females in short males are more th
```



In [69]:

```
# Creating a donut chart for 'Relationship'
relation = df['relationship'].value_counts()
plt.figure(figsize=(16, 10))
plt.pie(relation.values, labels=relation.index, startangle=50, autopct='%1.1f%%')
centre_circle = plt.Circle((0, 0), 0.7, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('Relationship distribution', fontdict={'fontsize': 20, 'fontweight': 'bold'})
plt.axis('equal')
plt.legend()
plt.show()
#From the 'Relationship Distribution graph' it can be analyse that most of the husbands are working.
```



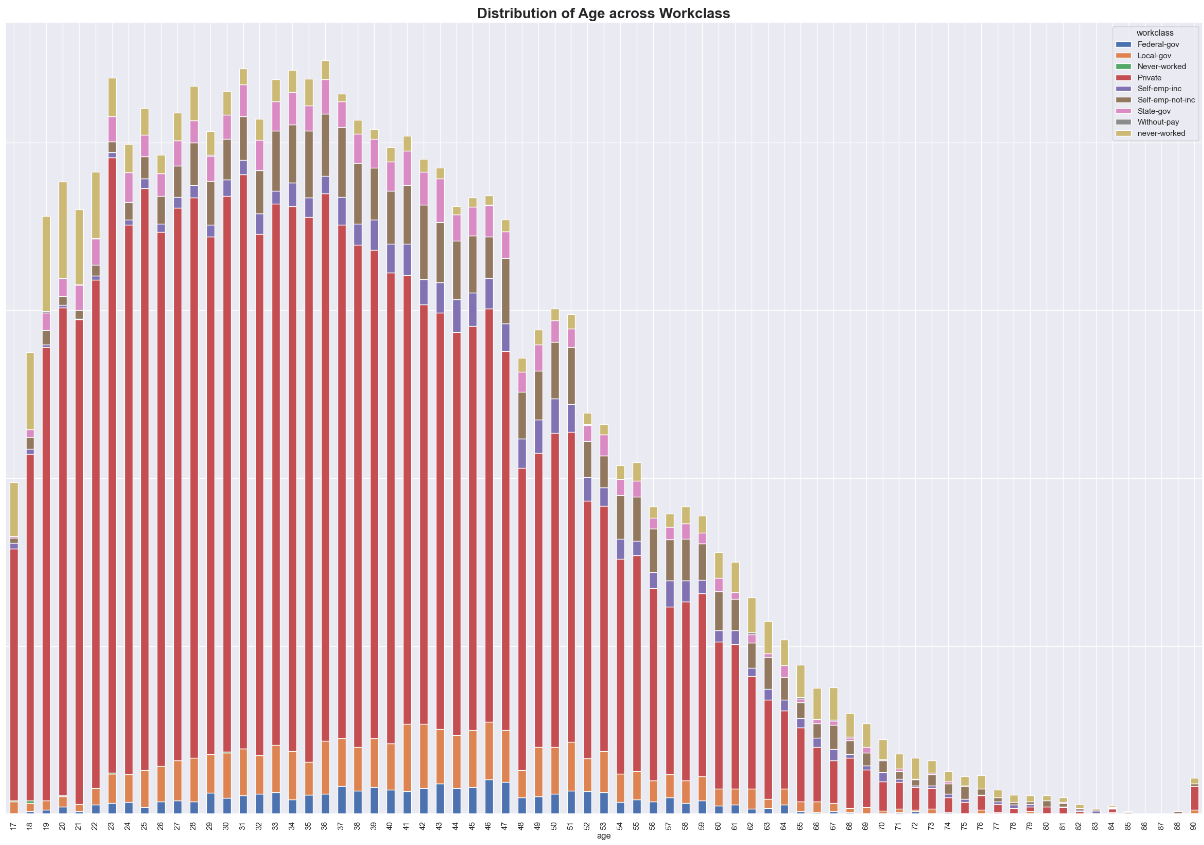
In [70]:

```
# Creating a barplot of Age across Workclass
plt.figure(figsize=(25,18))
df.groupby(['age', 'workclass']).size().unstack().plot(kind='bar', stacked=True)
plt.title('Distribution of Age across Workclass', fontdict={'fontsize': 20, 'fontweight': 'bold'})
#based on the graph analysis it is clear that most of the people are working privately in each and every age.
```

Out[70]:

```
Text(0.5, 1.0, 'Distribution of Age across Workclass')
```

```
<Figure size 2500x1800 with 0 Axes>
```

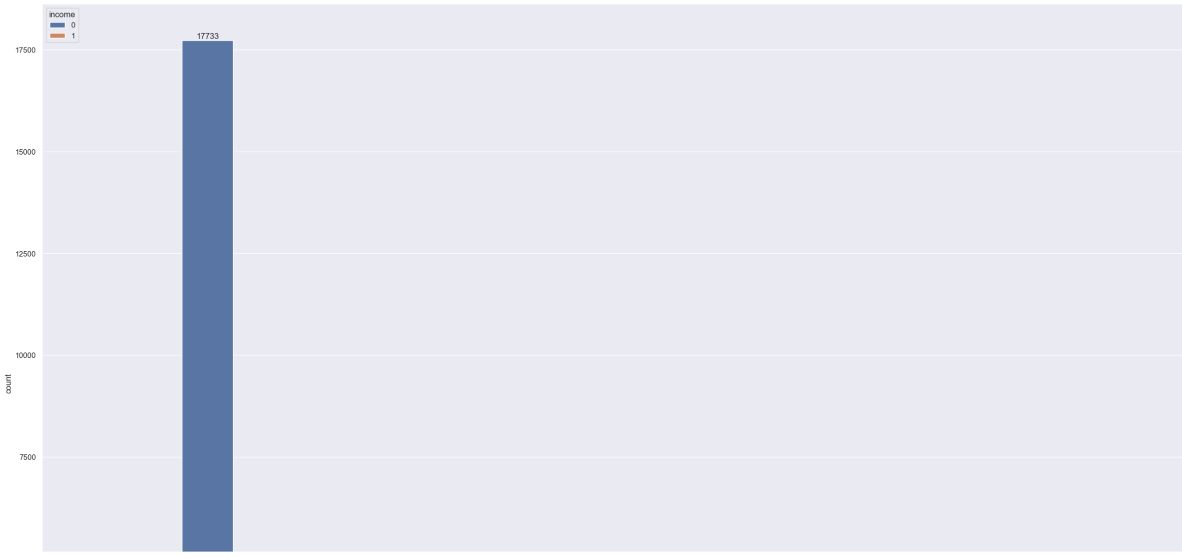


In [71]:

```
#from matplotlib.pyplot import figure



#category_var=['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss', 'hours.per.week','workclass',
             # 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.country']

#for i in category_var:
    #figure()
    #sns.barplot(y=df['income_num'], x=df[i])
    #sns.set(rc={'figure.figsize':(25,10)})
```

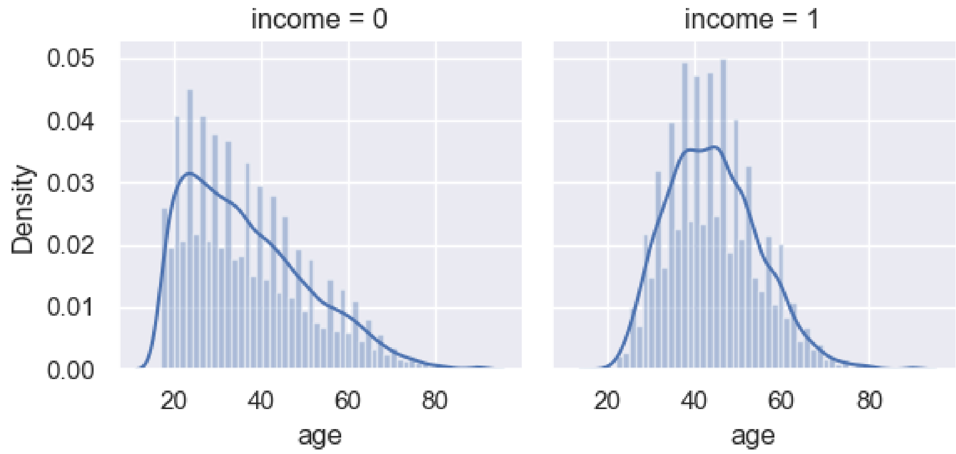
In [72]:

```
from matplotlib.pyplot import figure
#Creating a count plot for the following fields.
category_var=['workclass', 'education', 'occupation', 'relationship', 'race', 'sex', 'native.country',
              'hours.per.week', 'marital.status']
for i in category_var:
    figure()
    graph=sns.countplot(data=df, x=df[i] ,hue='income')
    for a in graph.containers:
        graph.bar_label(a)
    #sns.barplot(y=df['income_num'], x=df[i])
    sns.set(rc={'figure.figsize':(25,10)})
#GRAPH 1 --- From the 'Workclass' countplot graph it can be analyse that most of the people are working privat
                    #less than 50K income.

#GRAPH 2 --- From the 'Education' countplot graph it can be analyse that most of the peoples are High School G
                    #whose income is less than or equal to 50K.

#GRAPH 3 --- From the 'Occupation' countplot graph it can be analyse that most of the peoples work as Adm-Cler
                    #Craft repair.

#GRAPH 4 --- From the 'Relationship' countplot graph it can be analyse that maximum number of people does not
                    #with their family and had a salary less than or equal to 50K.

#GRAPH 5 --- From the 'Race' countplot graph it can be analyse that most of the people are white and have a in

#GRAPH 6 --- From the 'Sex' countplot graph it can be analyse that maximum population who is working is male a
             #<=50K

#Graph 7 --- From the 'native country' countplot graph it can be analyse that maximum number of people working
                   #and had a income <=50K

#Graph 8 --- From the 'hours per week' countplot graph it can be analyse that people mostly work 40 hours a we
       #salary <=50K

#Graph 9 --- From the 'Never married' countplot graph it can be analyse that people who are unmarried are work
                # rather tan others.
```



In [73]:

```
#Creating a plot for the following fields.
g = sns.FacetGrid(df, col='income')
g = g.map(sns.distplot, "age")
plt.show()
#From the graph shown below it can be analyse that people belonging to age 20-40 are more and have there incom
#From the graph shown below it can be analyse that people belonging to age 30-50 are more and have there incom
```

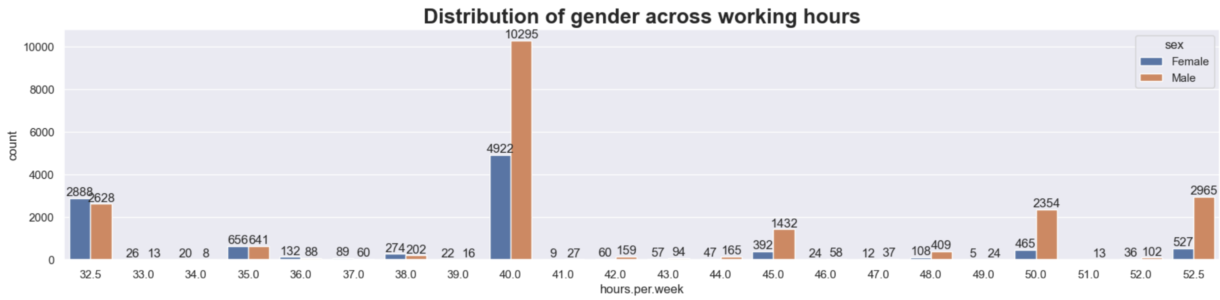


In [74]:

```
plt.figure(figsize=(20,4))
graph=sns.countplot(data=df, x='hours.per.week', hue='sex')
for i in graph.containers:
    graph.bar_label(i)
plt.title('Distribution of gender across working hours', fontdict={'fontsize': 20, 'fontweight': 'bold'})
```

Out[74]:

```
Text(0.5, 1.0, 'Distribution of gender across working hours')
```

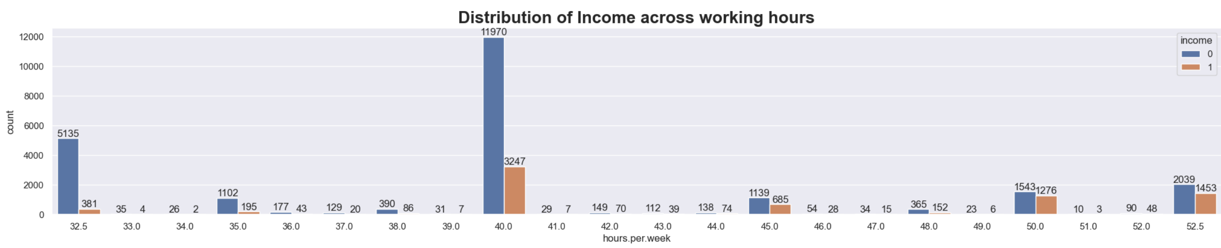


In [75]:

```
plt.figure(figsize=(25,4))
graph=sns.countplot(data=df, x='hours.per.week', hue='income')
for i in graph.containers:
    graph.bar_label(i)
plt.title('Distribution of Income across working hours', fontdict={'fontsize': 20, 'fontweight': 'bold'})
```

Out[75]:

```
Text(0.5, 1.0, 'Distribution of Income across working hours')
```

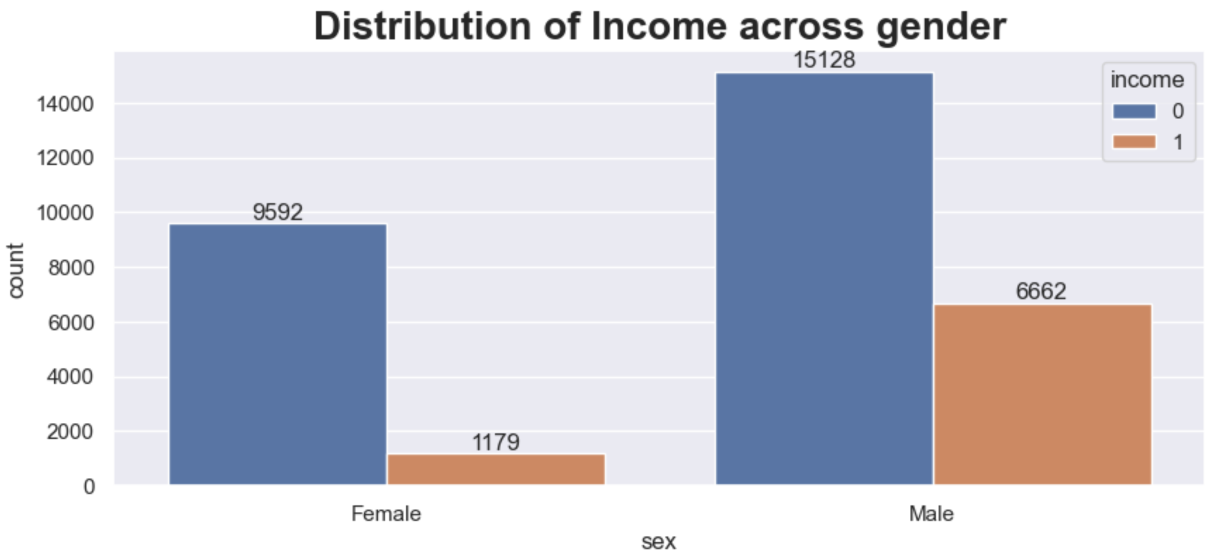


In [76]:

```
plt.figure(figsize=(10,4))
graph=sns.countplot(data=df, x='sex', hue='income')
for i in graph.containers:
    graph.bar_label(i)
plt.title('Distribution of Income across gender', fontdict={'fontsize': 20, 'fontweight': 'bold'})
```

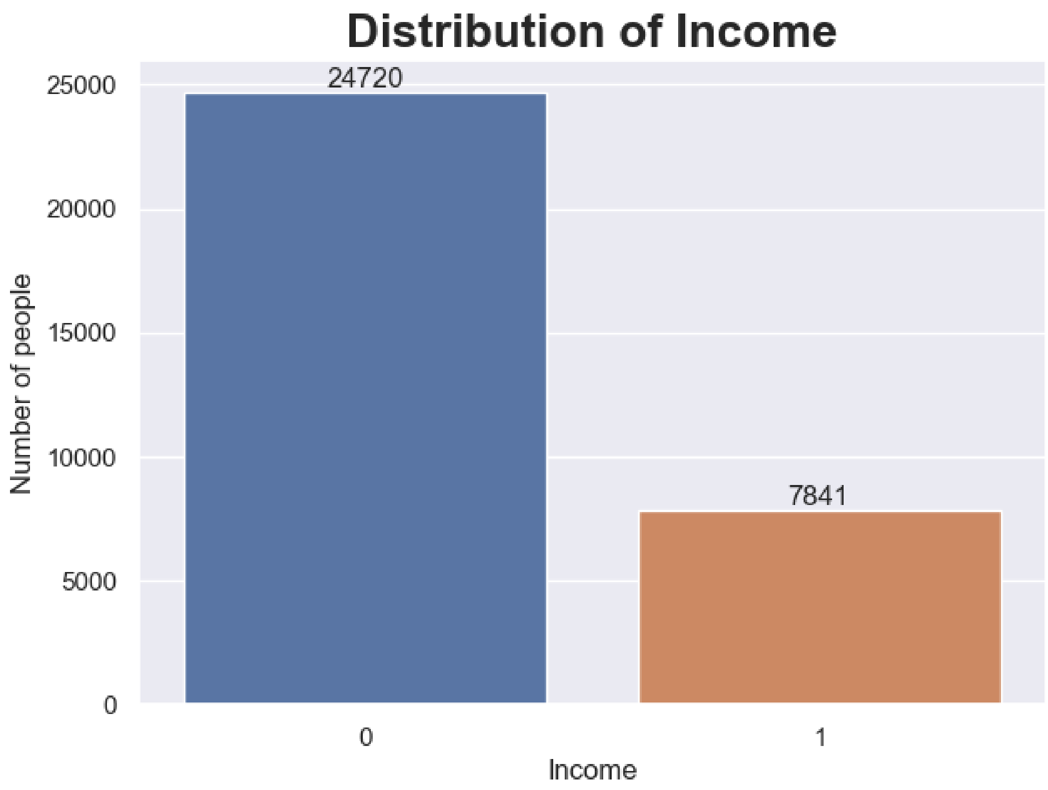
Out[76]:

```
Text(0.5, 1.0, 'Distribution of Income across gender')
```



In [77]:

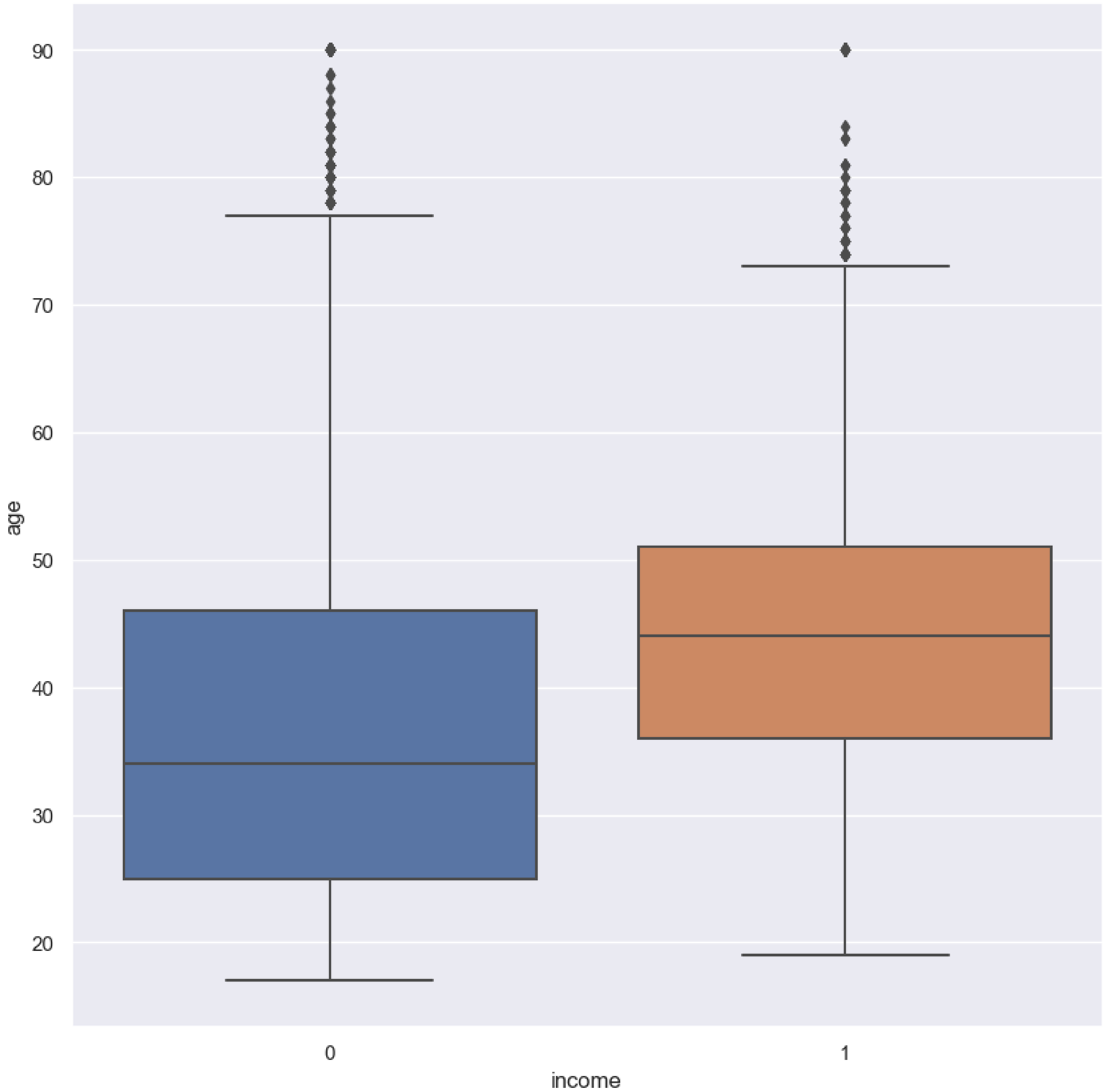
```
# Creating a barplot for 'Income'
income = df['income'].value_counts()
plt.figure(figsize=(7, 5))
graph=sns.barplot(income.index, income.values)
for a in graph.containers:
        graph.bar_label(a)
plt.title('Distribution of Income', fontdict={'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Income')
plt.ylabel('Number of people')
plt.show()
#From the graph shown below it is clear that maximum number of people's salary is less than or equal to 50 tho
```



Statistical Tests

In [78]:

```
#Boxplot analysis between age and income
fig = plt.figure(figsize=(10,10))
sns.boxplot(x="income", y="age", data=df)
plt.show()
#Outliers present in both the income group(<=50k and >50k) wrt "age" attribute.
#Income group(<=50k) has lower median "age"(34 year) than the Income group(>50k) which has median "age"(43 yea
#For Income group(<=50k) , Interquartile range(IQR) is between [25,46] (long range)
#For Income group(>50k) , Interquartile range(IQR) is between [35,50] (shorter range)
```



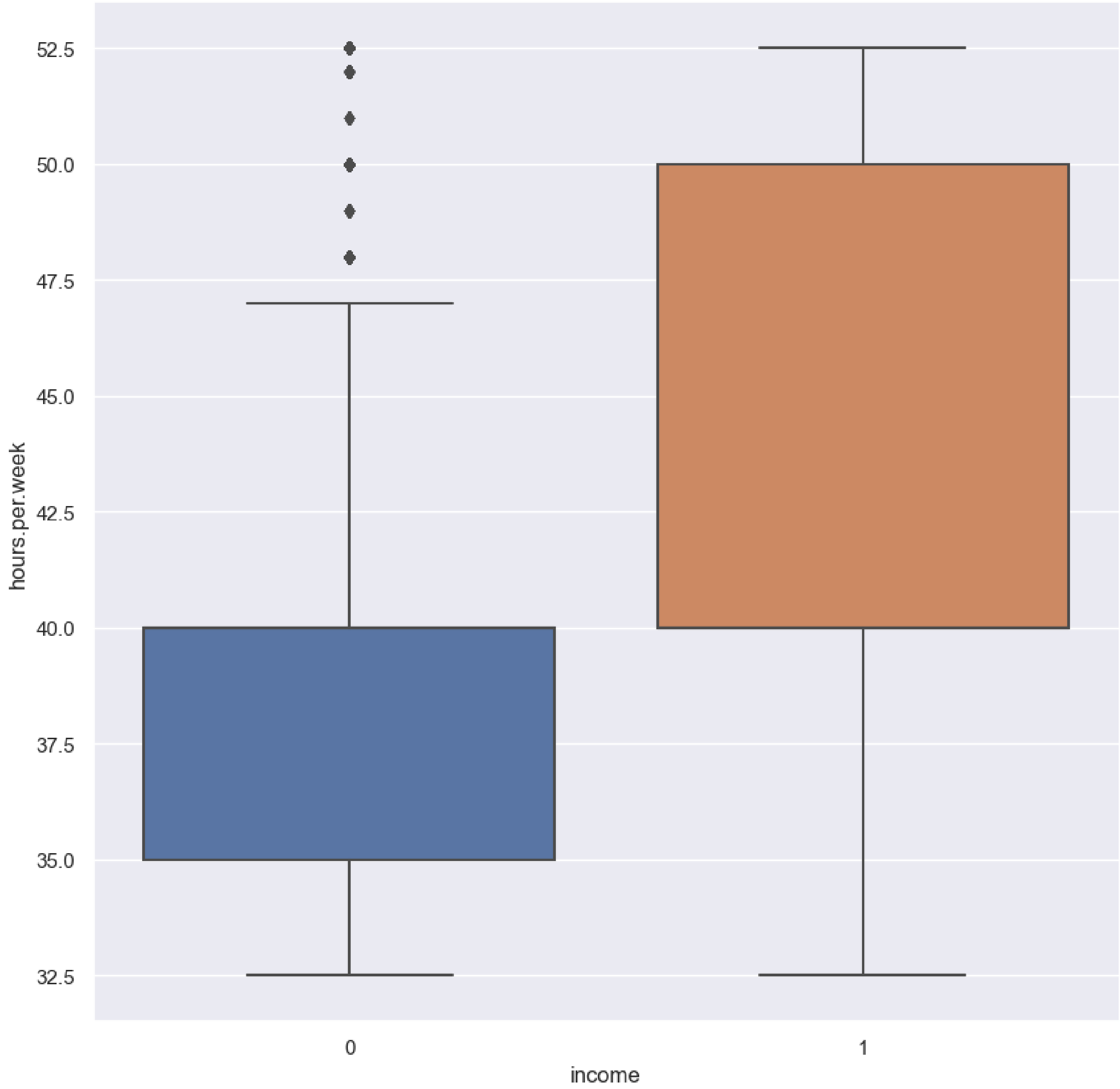
In [79]:

```
from scipy import stats
import random
from scipy.stats import ttest_ind, ttest_rel
#Hypothesis test (to test the relationship between 'income' & 'age' )
df = df[(np.abs(stats.zscore(df["age"])) < 3)]
income_1 = df[df['income']==1]['age']
income_0 = df[df['income']==0]['age']

income_0 = income_0.values.tolist()
income_0 = random.sample(income_0, 100)
income_1 = income_1.values.tolist()
income_1 = random.sample(income_1, 100)

ttest,pval = ttest_ind(income_1,income_0,equal_var = False)
print("ttest",ttest)
print('p value',pval)


if pval <0.05:
    print("we reject null hypothesis")
else:
    print("we accept null hypothesis")


#Using statistical analysis, we conclude that there is a significant difference in the mean ages of income gro
       #and income group <=50k.It means that age has some contribution to the distinguish income groups.
```

```
ttest 4.611001896522119
p value 7.344483569695718e-06
we reject null hypothesis
```

In [80]:

```
#Boxplot relation between 'income' and 'hours.per.week'
fig = plt.figure(figsize=(10,10))
sns.boxplot(x="income", y="hours.per.week", data=df)
plt.show()

#The median "hours.per.week" for income group who earns >50k is greater than the income group who earns <=50k
#The boxplot for Income group who earns <=50k has small range ~[28,48].
#The boxplot for Income group who earns >50k has large range ~[25,65].
#Income group who earns >50k have flexible working hours
#More Outliers present in the Income group who earns <=50k.
```



In [81]:

```
#Hypothesis test (to test the relationship between 'income' & 'hours.per.week' )
df = df[(np.abs(stats.zscore(df["hours.per.week"])) < 3)]
income_1 = df[df['income']==1]["hours.per.week"]
income_0 = df[df['income']==0]["hours.per.week"]

income_0 = income_0.values.tolist()
income_0 = random.sample(income_0, 100)
income_1 = income_1.values.tolist()
income_1 = random.sample(income_1, 100)

ttest,pval = ttest_ind(income_1,income_0,equal_var = False)
print("ttest",ttest)
print('p value',format(pval, '.70f'))

if pval <0.05:
    print("we reject null hypothesis")
else:
    print("we accept null hypothesis")

#We can conclude that there is difference in Mean of income group >50k and income group <=50k.
#It means that hours-per-week has some contribution to the distinguish income groups.
```

```
ttest 2.9916437116170367
p value 0.0031279895728443369261329021213668966083787381649017333984375000000000
we reject null hypothesis
```

In [82]:

```
from scipy import stats
```

In [83]:

```
df.head(8)
```

Out[83]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.ga
2	66	never-worked	186061	Some-college	10	Widowed	never-worked	Unmarried	Black	Female	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	
7	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	
8	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	
9	41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	

In [84]:

```
df.tail(8)
```

Out[84]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	ca
32553	43	Private	84661	Assoc-voc	11	Married-civ-spouse	Sales	Husband	White	Male	
32554	32	Private	116138	Masters	14	Never-married	Tech-support	Not-in-family	Asian-Pac-Islander	Male	
32555	53	Private	321865	Masters	14	Married-civ-spouse	Exec-managerial	Husband	White	Male	
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Male	
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	

In [85]:

```
Sales = df[(df['occupation'] == 'Sales')]
Sales.shape
```

Out[85]:

```
(3639, 15)
```

In [86]:

```
Adm=df[(df['occupation'] == 'Adm-clerical')]
Adm.shape
```

Out[86]:

```
(3759, 15)
```

In [87]:

```
Sales['income']=Sales['income'].sample(28)
Adm['income']=Adm['income'].sample(28)
```

In [88]:

```
print(np.mean(Sales['age']))
print(np.mean(Adm['age']))
```

```
37.21077219016213
36.82255919127427
```

In [89]:

```
tvalue,pvalue=stats.ttest_ind(Sales['age'], Adm['age'])
```

In [90]:

```
pvalue
```

Out[90]:

```
0.21773563264672965
```

In [91]:

```
tvalue
```

Out[91]:

```
1.2326765836747073
```

In [92]:

```
H0="Mean value of both distributions is same"
H1="Mean value is different"
```

In [93]:

```
if pvalue>=0.05:
    print(H0)
else:
    print(H1)
```

```
Mean value of both distributions is same
```

In [94]:

```
df['age'].head(20)
```

Out[94]:

```
2     66
3     54
4     41
5     34
6     38
7     74
8     68
9     41
10    45
11    38
12    52
13    32
14    51
15    46
16    45
17    57
18    22
19    34
20    37
21    29
Name: age, dtype: int64
```

In [95]:

```
# ztest
```

In [96]:

```
capital_gain=df[df['capital.gain']==0]['income']
capital_loss=df[df['capital.loss']>0]['income']
```

In [97]:

```
from statsmodels.stats.weightstats import ztest
```

In [98]:

```
z_score,p_val = ztest(capital_gain,capital_loss)
if p_val>0.05:
    print('Ho:hypothsis is true(there is no effect in income)')
else:
    print('H1:hypothsis in not true(there is effect on income)')
```

```
H1:hypothsis in not true(there is effect on income)
```

In [99]:

```
print(p_val)
```

```
8.991549662209643e-173
```

In [100]:

```
#Conclusions
#We did the entire EDA process for this dataset from looking at the head of the dataset to get the insights o
#every feature whether it is univariate analysis or the bivariate analysis and along with getting the insights
#numerically we also have used two one of the most interactive visualization libraries i.e. Count plot, Bar pl
#heatmap,line plot, distplot, etc..





#75.92% of them are belong to income group 1 (who earns more than 50k) and 24.08% fall under the income group
#          less than 50k).

#Females have more flexible working hours per week in the income groups who earns <=50k.

#Males have more flexible working hours per week in the income groups who earns >50k.

#Generally people can be seen working for 30 hours to 40 hours per week and they are not living with their fa

#For "female" earning more than 50k is rare with only 3.57% of all observations But for male, 19.99% of all pe
            #more than 50k .

#self-emp-inc workclass is only where more people earn >50k(belong to income group 1).

#People having degree doctorate,prof-school,masters are making salary more than 50K

#The people who are working mostly are unmarried probably belong to United states and working in private sect
          #occupation is Adm-clerical.

#people of age group 25-45 are mostly working.

#most of the people who are working privately are high school graduate.

#maximum people race is white and males are more than female in whole population.

#Males are doing there jobs more than females and mostly males who are working are husbands.
```

In [101]:

```
from sklearn.preprocessing import LabelEncoder
```

In [102]:

```
for col in df.columns:
    if df[col].dtypes == 'object':
        encoder = LabelEncoder()
        df[col] = encoder.fit_transform(df[col])
```

Model Building

In [103]:

```
X=df.drop(['income'],axis=1)
Y=df['income']
```

Feature scaling

In [104]:

```
#As we have many features contains categorical variable so we are using pandas get_dummies function to convert
```

In [105]:

```
df= pd.get_dummies(df,drop_first=True)
pd.set_option('display.max_columns',100)#to display all columns
```

In [106]:

```
df.head(10)
#Now our data set has been transform into numeric.
```

Out[106]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain
2	66	8	186061	15	10	6	14	4	2	0	0
3	54	3	140359	5	4	0	6	4	4	0	0
4	41	3	264663	15	10	5	9	3	4	0	0
5	34	3	216864	11	9	0	7	4	4	0	0
6	38	3	150601	0	6	5	0	4	4	1	0
7	74	6	88638	10	16	4	9	2	4	0	0
8	68	0	422013	11	9	0	9	1	4	0	0
9	41	3	70037	15	10	4	2	4	4	1	0
10	45	3	172274	10	16	0	9	4	2	0	0
11	38	5	164526	14	15	4	9	1	4	1	0

In [107]:

```
df.shape
```

Out[107]:

(32440, 15)

In [108]:

```
# Now our almost data values is 0 and 1 except few features like "'Age','Fnlwgt','Education_num','Hours_per_we
# we can use standard scaler we and convert those features in same scale.
```

In [109]:

```
from sklearn.preprocessing import StandardScaler
```

In [110]:

```
scaler = StandardScaler()
train_col_sacle = df[['age','fnlwgt','education.num','hours.per.week']]
train_scaler_col = scaler.fit_transform(train_col_sacle)
train_scaler_col = pd.DataFrame(train_scaler_col,columns=train_col_sacle.columns)
df['age']= train_scaler_col['age']
df['fnlwgt']= train_scaler_col['fnlwgt']
df['education.num']= train_scaler_col['education.num']
df['hours.per.week']= train_scaler_col['hours.per.week']
```

In [111]:

```
#Data is now divided in independent and dependent.
```

Creating a train test split

In [112]:

```
from sklearn.model_selection import train_test_split
```

In [113]:

```
X_train, X_test, Y_train, Y_test  = train_test_split(X,Y, test_size=0.30, random_state=11)
```

In [114]:

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", Y_train.shape)
print("Y_test shape:", Y_test.shape)
```

```
X_train shape: (22708, 14)
X_test shape: (9732, 14)
Y_train shape: (22708,)
Y_test shape: (9732,)
```

In [115]:

```
#Our data set divided into train and test. Now we will continue with model building.
```

Data Modelling

Logistic Regression

In [116]:

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state=42)
```

In [117]:

```
log_reg.fit(X_train, Y_train)
```

Out[117]:

```
LogisticRegression(random_state=42)
```

In [118]:

```
Y_pred_log_reg = log_reg.predict(X_test)
```

KNN Classifier

In [119]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
```

In [120]:

```
knn.fit(X_train, Y_train)
```

Out[120]:

```
KNeighborsClassifier()
```

In [121]:

```
Y_pred_knn = knn.predict(X_test)
```

Decision Tress

In [122]:

```
from sklearn.tree import DecisionTreeClassifier
dec_tree = DecisionTreeClassifier(random_state=42)
```

In [123]:

```
dec_tree.fit(X_train, Y_train)
```

Out[123]:

```
DecisionTreeClassifier(random_state=42)
```

In [124]:

```
Y_pred_dec_tree = dec_tree.predict(X_test)
```

Random Forest Classifier

In [125]:

```
from sklearn.ensemble import RandomForestClassifier
ran_for = RandomForestClassifier(random_state=123)
```

In [126]:

```
ran_for.fit(X_train, Y_train)
```

Out[126]:

```
RandomForestClassifier(random_state=123)
```

In [127]:

```
Y_pred_ran_for = ran_for.predict(X_test)
```

Support Vector Classifier

In [128]:

```
from sklearn.svm import SVC
svc = SVC(random_state=42)
```

In [129]:

```
svc.fit(X_train, Y_train)
```

Out[129]:

```
SVC(random_state=42)
```

In [130]:

```
Y_pred_svc = svc.predict(X_test)
```

Model Evaluation

In [131]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
```

In [132]:

```
print('Logistic Regression:')
print('Accuracy score:', round(accuracy_score(Y_test, Y_pred_log_reg) * 100, 2))
print('F1 score:', round(f1_score(Y_test, Y_pred_log_reg) * 100, 2))
```

```
Logistic Regression:
Accuracy score: 79.47
F1 score: 39.2
```

In [133]:

```
print('KNN Classifier:')
print('Accuracy score:', round(accuracy_score(Y_test, Y_pred_knn) * 100, 2))
print('F1 score:', round(f1_score(Y_test, Y_pred_knn) * 100, 2))
```

```
KNN Classifier:
Accuracy score: 77.36
F1 score: 41.55
```

In [134]:

```
print('Decision Tree Classifier:')
print('Accuracy score:', round(accuracy_score(Y_test, Y_pred_dec_tree) * 100, 2))
print('F1 score:', round(f1_score(Y_test, Y_pred_dec_tree) * 100, 2))
```

```
Decision Tree Classifier:
Accuracy score: 80.59
F1 score: 61.24
```

In [135]:

```
print('Random Forest Classifier:')
print('Accuracy score:', round(accuracy_score(Y_test, Y_pred_ran_for) * 100, 2))
print('F1 score:', round(f1_score(Y_test, Y_pred_ran_for) * 100, 2))
```

```
Random Forest Classifier:
Accuracy score: 85.9
F1 score: 68.63
```

In [136]:

```
print('Support Vector Classifier:')
print('Accuracy score:', round(accuracy_score(Y_test, Y_pred_svc) * 100, 2))
print('F1 score:', round(f1_score(Y_test, Y_pred_svc) * 100, 2))
```

```
Support Vector Classifier:
Accuracy score: 78.88
F1 score: 26.48
```

In [137]:

```
#From the above Model building outcomes it can be analyse that Random Forest Classifier & Decision Tree Classi
#the best models with best F1 score and Accuracy score.
```

Hyperparameter Tuning

In [138]:

```
from sklearn.model_selection import RandomizedSearchCV
```

In [139]:

```
n_estimators = [int(x) for x in np.linspace(start=40, stop=150, num=15)]
max_depth = [int(x) for x in np.linspace(40, 150, num=15)]
```

In [140]:

```
param_dist = {
    'n_estimators': n_estimators,
    'max_depth': max_depth,
}
```

In [141]:

```
rf_tuned = RandomForestClassifier(random_state=42)
```

In [142]:

```
rf_cv = RandomizedSearchCV(
    estimator=rf_tuned, param_distributions=param_dist, cv=5, random_state=42)
```

In [143]:

```
rf_cv.fit(X_train, Y_train)
```

Out[143]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
                   param_distributions={'max_depth': [40, 47, 55, 63, 71, 79,
                                                      87, 95, 102, 110, 118,
                                                      126, 134, 142, 150],
                                        'n_estimators': [40, 47, 55, 63, 71, 79,
                                                         87, 95, 102, 110, 118,
                                                         126, 134, 142, 150]},
                   random_state=42)
```

In [144]:

```
rf_cv.best_score_
```

Out[144]:

```
0.8561741735434409
```

In [145]:

```
rf_cv.best_params_
```

Out[145]:

```
{'n_estimators': 126, 'max_depth': 79}
```

In [146]:

```
rf_best = RandomForestClassifier(
    max_depth=102, n_estimators=142, random_state=123)
```

In [147]:

```
rf_best.fit(X_train, Y_train)
```

Out[147]:

```
RandomForestClassifier(max_depth=102, n_estimators=142, random_state=123)
```

In [148]:

```
Y_pred_rf_best = rf_best.predict(X_test)
```

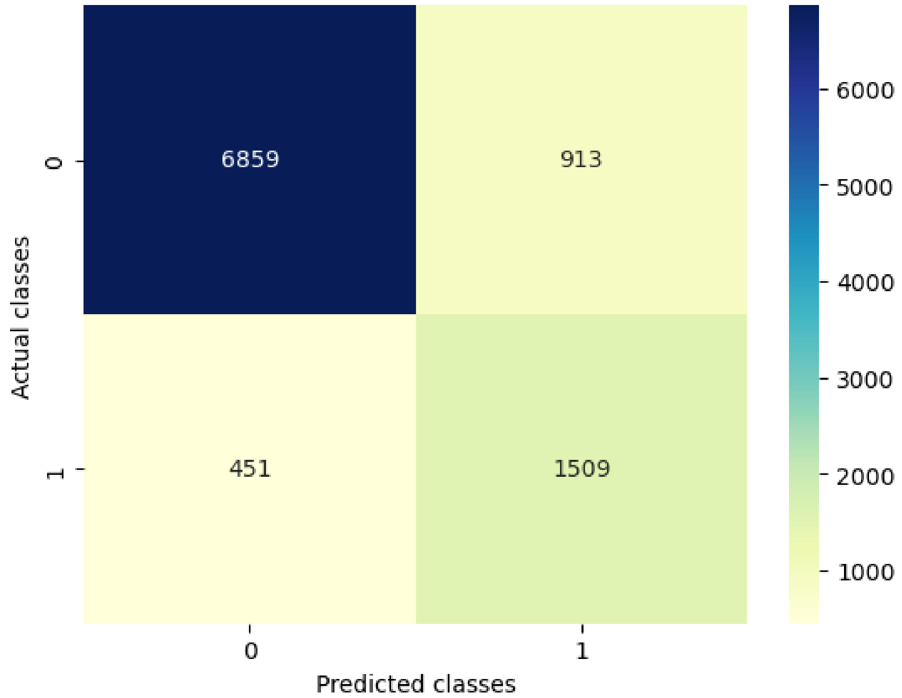
In [149]:

```
print('Random Forest Classifier:')
print('Accuracy score:', round(accuracy_score(Y_test, Y_pred_rf_best) * 100, 2))
print('F1 score:', round(f1_score(Y_test, Y_pred_rf_best) * 100, 2))
```

```
Random Forest Classifier:
Accuracy score: 85.98
F1 score: 68.87
```

In [150]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix( Y_pred_rf_best, Y_test)
plt.style.use('default')
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu')
plt.xlabel('Predicted classes')
plt.ylabel('Actual classes')
plt.savefig('heatmap.png')
plt.show()
```



In [151]:

```
#Interpretation

#Y-axis represents the actual classes
#X-axis represents the predicted classes
#6859 times when the model correctly predicted 0 when the actual class was 0
#451 times the model predicted 0 when the actual class was 1
#913 times the model predicted 1 when the actual class was 0
#1509 times the model correctly predicted 1 when the actual class was 1
```

In [152]:

```
from sklearn.metrics import classification_report
print(classification_report(Y_test, Y_pred_rf_best))
```

```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      7310
           1       0.77      0.62      0.69      2422

    accuracy                           0.86      9732
   macro avg       0.83      0.78      0.80      9732
weighted avg       0.85      0.86      0.85      9732
```

In [153]:

```
#In this project, we build various models like
    # logistic regression
    # knn classifier
    # support vector classifier
    # decision tree classifier
    # random forest classifier

#A hyperparameter tuned random forest classifier gives the highest accuracy score of 85.98 and f1 score of 68
```

Other method of hyper parameter tuning

In [*]:

```
from sklearn.model_selection import GridSearchCV
n_estimators_List = [40, 47, 55, 63, 71, 79, 87, 95, 102, 110, 118, 126, 134, 142, 150]
max_features_List =[40, 47, 55, 63, 71, 79, 87, 95, 102, 110, 118, 126, 134, 142, 150]
min_samples_leaf_List = [5, 10, 25, 50, 30, 35, 40, 75, 85, 105, 110, 125, 130, 145, 150]

my_param_grid = {'n_estimators': n_estimators_List,
                 'max_features': max_features_List,
                 'min_samples_leaf' : min_samples_leaf_List}

Grid_Search_Model = GridSearchCV(estimator = RandomForestClassifier(random_state=123),
                     param_grid=my_param_grid, scoring='accuracy', cv=3).fit(X_train, Y_train) # param_grid i

Model_Validation_Df4 = pd.DataFrame.from_dict(Grid_Search_Model.cv_results_)
# Grid_Search_Model.cv_results_

# Based on the selected hyperparamters, you should build a final model on the COMPLETE training data (trainX,
RF_Final = RandomForestClassifier(random_state = 123, n_estimators = 75,
                              max_features = 9, min_samples_leaf = 5).fit(X_train, Y_train)
Test_Pred = RF_Final.predict(X_test)

# Confusion Matrix
Confusion_Mat = pd.crosstab(Y_test, Test_Pred) # R, C format (Actual = testY, Predicted = Test_Pred)
Confusion_Mat

# Validation on Testset
print(classification_report(Y_test, Test_Pred)) # Actual, Predicted
```

In [ ]:

```

```