In [2]:
```
# Importing required libraries

import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df= pd.read_csv('https://raw.githubusercontent.com/dsrscientist/dataset1/master/c
```

In [4]:
```
df
```

Out[4]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationsh
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husbar
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-fam
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husbar
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wi
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wi
...	...	...	...	...	...	...	...	
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wi
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husbar
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarrie
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-ch
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wi

32560 rows × 15 columns

In [5]:
```
df.shape
```

Out[5]:
```
(32560, 15)
```

In [6]:
```
print("Data have ",df.shape[0],'Rows and ',df.shape[1] ,'Columns')
```

```
Data have  32560 Rows and  15 Columns
```

In [7]:
```
df.head()
```

Out[7]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	I
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	V
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	V
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	E
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	E
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	V

In [8]:
```
df.tail()
```

Out[8]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationsh
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wi
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husbar
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarrie
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-ch
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wi

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32560 entries, 0 to 32559
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Age             32560 non-null  int64 
 1   Workclass       32560 non-null  object
 2   Fnlwgt          32560 non-null  int64 
 3   Education       32560 non-null  object
 4   Education_num   32560 non-null  int64 
 5   Marital_status  32560 non-null  object
 6   Occupation      32560 non-null  object
 7   Relationship    32560 non-null  object
 8   Race            32560 non-null  object
 9   Sex             32560 non-null  object
 10  Capital_gain    32560 non-null  int64 
 11  Capital_loss    32560 non-null  int64 
 12  Hours_per_week  32560 non-null  int64 
 13  Native_country  32560 non-null  object
 14  Income          32560 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [10]:
```
# we have no null values,
# 6  features are of Integer data types
# 9  features are of Catagorical data typ
```

In [11]:
```
df.describe().T
```

Out[11]:

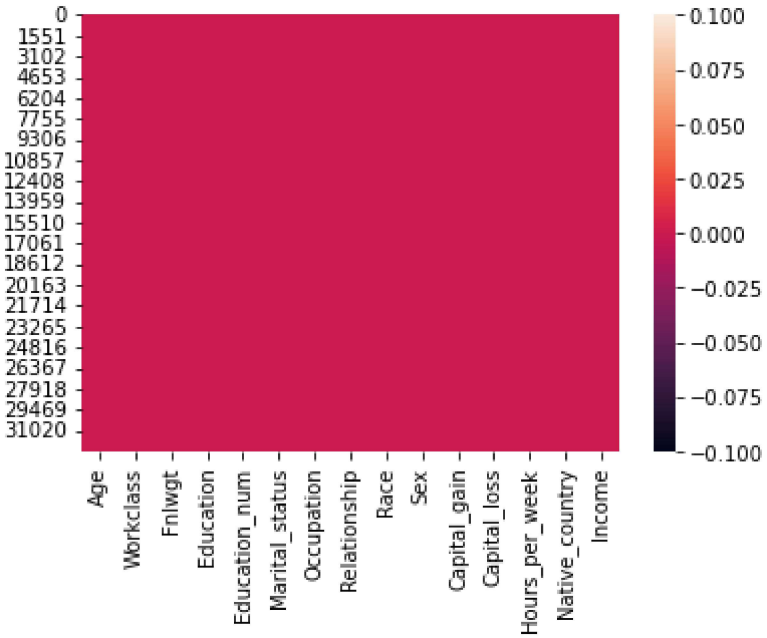
	count	mean	std	min	25%	50%	75%	
Age	32560.0	38.581634	13.640642	17.0	28.0	37.0	48.0	
Fnlwgt	32560.0	189781.814373	105549.764924	12285.0	117831.5	178363.0	237054.5	14
Education_num	32560.0	10.080590	2.572709	1.0	9.0	10.0	12.0	
Capital_gain	32560.0	1077.615172	7385.402999	0.0	0.0	0.0	0.0	
Capital_loss	32560.0	87.306511	402.966116	0.0	0.0	0.0	0.0	
Hours_per_week	32560.0	40.437469	12.347618	1.0	40.0	40.0	45.0	

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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 [13]:
```
sns.heatmap(df.isnull())
```

Out[13]: <AxesSubplot:>



In [14]:
```
# Null Vlaues in %

(df.isnull().sum()/df.shape[0])*100
```

Out[14]:
```
Age               0.0
Workclass         0.0
Fnlwgt            0.0
Education         0.0
Education_num     0.0
Marital_status    0.0
Occupation        0.0
Relationship      0.0
Race              0.0
Sex               0.0
Capital_gain      0.0
Capital_loss      0.0
Hours_per_week    0.0
Native_country    0.0
Income            0.0
dtype: float64
```

In [15]:
```
# Check unique values:
df.columns
```

Out[15]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

In [16]:
```
for i in df.columns:
    print( i ,'have unique values = ',df[i].unique(),'\n')
```

```
Age have unique values =  [50 38 53 28 37 49 52 31 42 30 23 32 40 34 25 43 54 3
5 59 56 19 39 20 45
 22 48 21 24 57 44 41 29 18 47 46 36 79 27 67 33 76 17 55 61 70 64 71 68
 66 51 58 26 60 90 75 65 77 62 63 80 72 74 69 73 81 78 88 82 83 84 85 86
 87]

Workclass have unique values =  [' Self-emp-not-inc' ' Private' ' State-gov' '
Federal-gov' ' Local-gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked']

Fnlwgt have unique values =  [ 83311 215646 234721 ...  34066  84661 257302]

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

Education_num have unique values =  [13  9  7 14  5 10 12 11  4 16 15  3  6  2
1  8]

Marital_status have unique values =  [' Married-civ-spouse' ' Divorced' ' Marri
ed-spouse-absent'
 ' Never-married' ' Separated' ' Married-AF-spouse' ' Widowed']

Occupation have unique values =  [' Exec-managerial' ' Handlers-cleaners' ' Pro
f-specialty'
 ' Other-service' ' Adm-clerical' ' Sales' ' Craft-repair'
 ' Transport-moving' ' Farming-fishing' ' Machine-op-inspct'
 ' Tech-support' ' ?' ' Protective-serv' ' Armed-Forces'
 ' Priv-house-serv']

Relationship have unique values =  [' Husband' ' Not-in-family' ' Wife' ' Own-c
hild' ' Unmarried'
 ' Other-relative']

Race have unique values =  [' White' ' Black' ' Asian-Pac-Islander' ' Amer-Indi
an-Eskimo' ' Other']

Sex have unique values =  [' Male' ' Female']

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

Capital_loss have unique values =  [   0 2042 1408 1902 1573 1887 1719 1762 156
4 2179 1816 1980 1977 1876
```

```
 1340 2206 1741 1485 2339 2415 1380 1721 2051 2377 1669 2352 1672  653
 2392 1504 2001 1590 1651 1628 1848 1740 2002 1579 2258 1602  419 2547
 2174 2205 1726 2444 1138 2238  625  213 1539  880 1668 1092 1594 3004
 2231 1844  810 2824 2559 2057 1974  974 2149 1825 1735 1258 2129 2603
 2282  323 4356 2246 1617 1648 2489 3770 1755 3683 2267 2080 2457  155
 3900 2201 1944 2467 2163 2754 2472 1411]

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

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

Income have unique values =  [' <=50K' ' >50K']
```

In [17]:
```
# Workclass  , Occupation , Native_country  have some null values in form of ' ?
df.isin([' ?']).sum()
```

Out[17]:
```
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 [18]:
```
# % withcolumns

(df.isin([' ?']).sum()/df.shape[0])*100
```

Out[18]:
```
Age               0.000000
Workclass         5.638821
Fnlwgt            0.000000
Education         0.000000
Education_num     0.000000
Marital_status    0.000000
Occupation        5.660319
Relationship      0.000000
Race              0.000000
Sex               0.000000
Capital_gain      0.000000
Capital_loss      0.000000
Hours_per_week    0.000000
Native_country    1.790541
Income            0.000000
dtype: float64
```

In [19]:
```
df.head()
```

Out[19]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	V
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	V
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	E
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	E
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	V

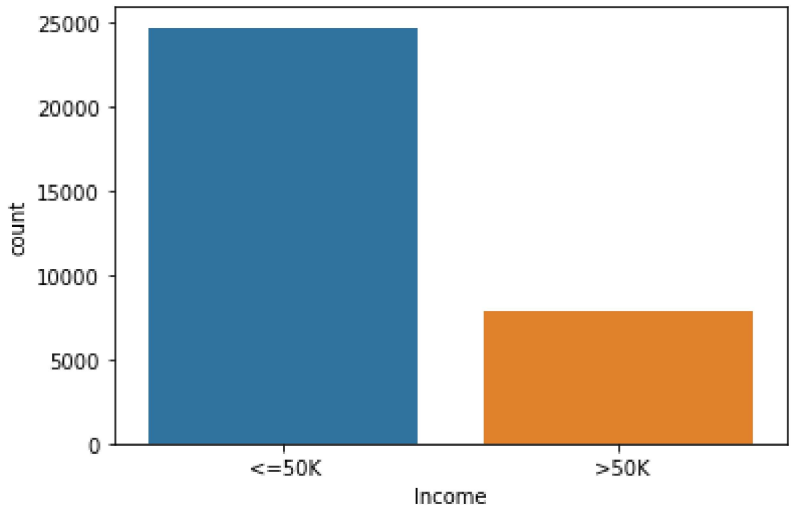
EDA

In [20]:
```
# Target Variable  : Income
df['Income'].value_counts()
```

Out[20]:
```
 <=50K    24719
 >50K      7841
Name: Income, dtype: int64
```

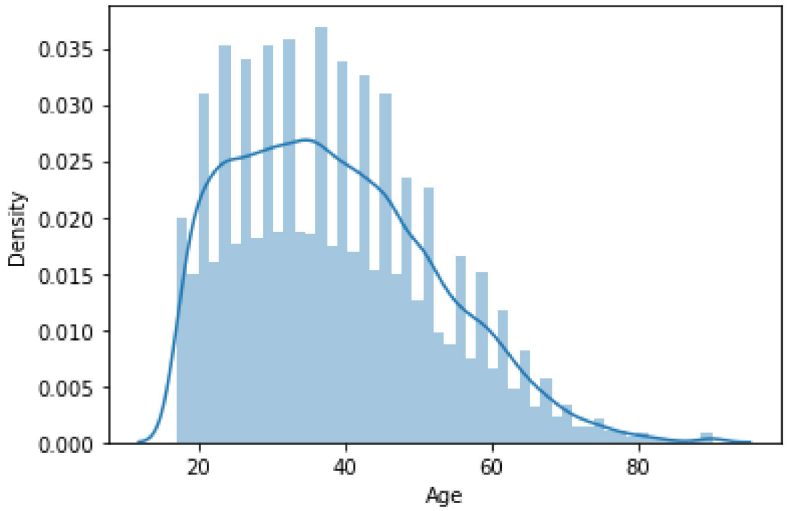
In [21]:
```
# This is imbalanced data, we have to balanced this as well
sns.countplot(df['Income'])
```

Out[21]: <AxesSubplot:xlabel='Income', ylabel='count'>



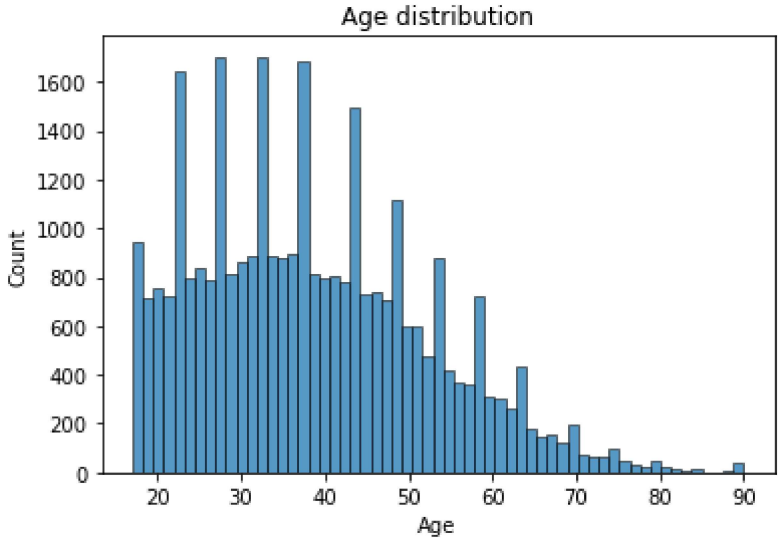
In [22]:
```
# Age
sns.distplot(df['Age'])
```

Out[22]: <AxesSubplot:xlabel='Age', ylabel='Density'>



In [23]:
```
sns.histplot(df['Age'])
plt.title('Age distribution')
```

Out[23]:
```
Text(0.5, 1.0, 'Age distribution')
```

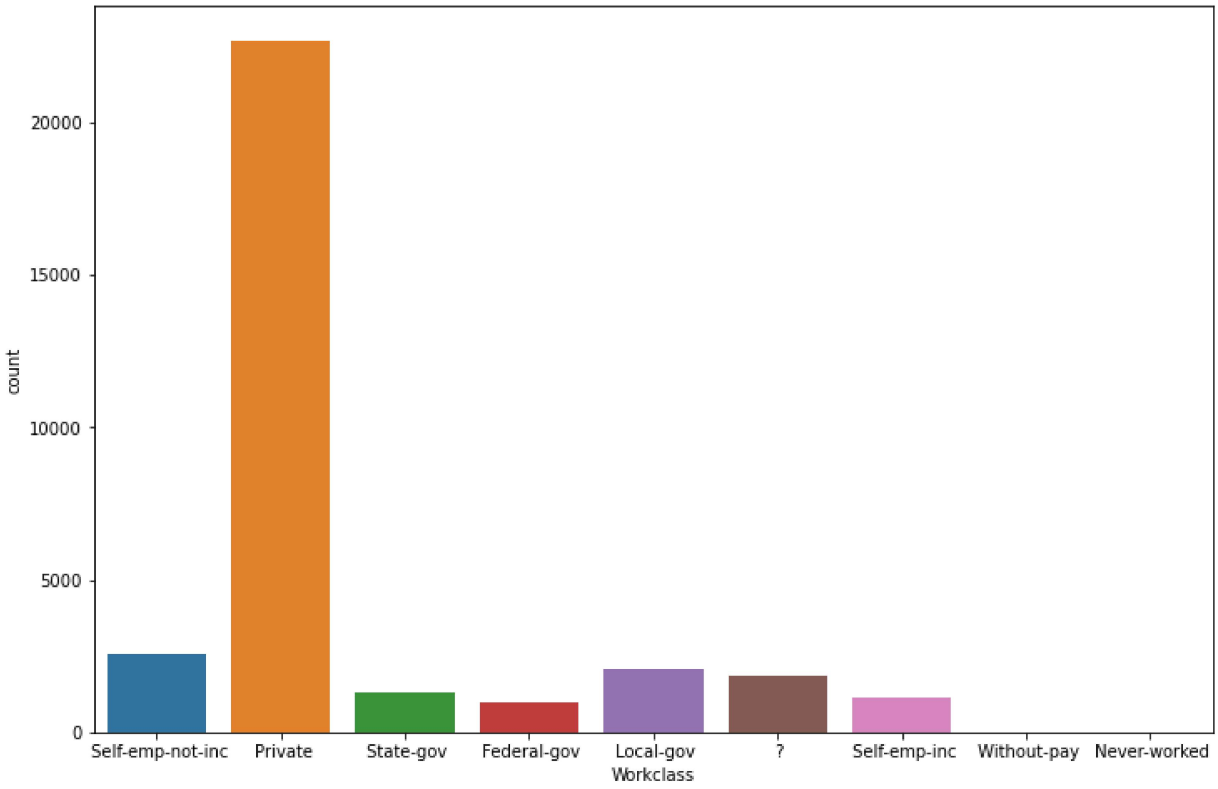


In [24]:
```
# Age data is right skewed
df['Workclass'].unique()
```

Out[24]:
```
array([' Self-emp-not-inc', ' Private', ' State-gov', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

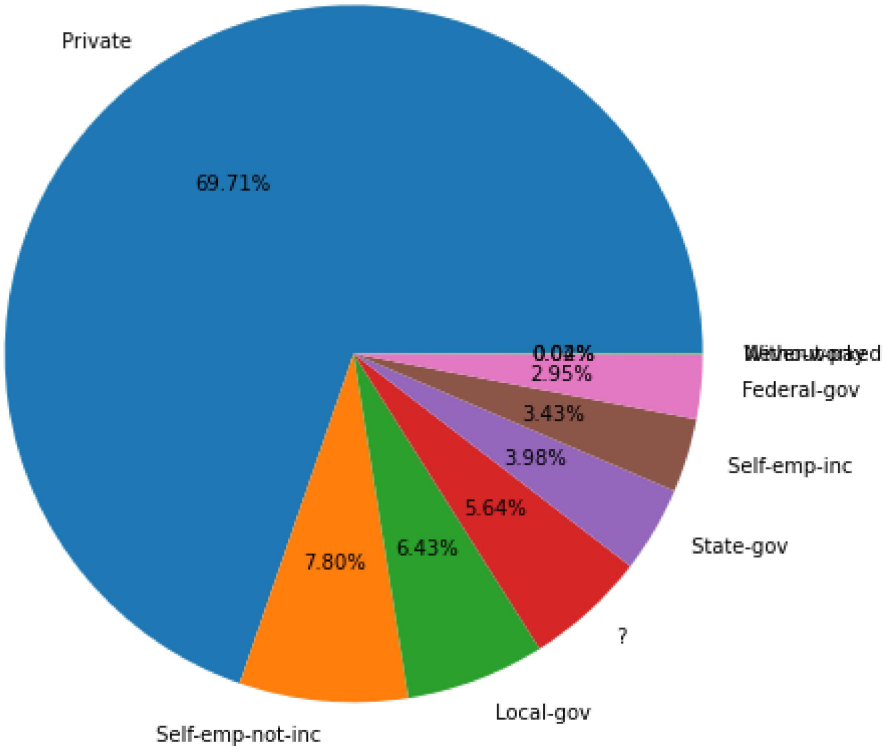
In [25]:
```
plt.figure(figsize=(12,8))
sns.countplot(df['Workclass'])
```

Out[25]: <AxesSubplot:xlabel='Workclass', ylabel='count'>



In [26]:
```
plt.figure(figsize=(12,8))
plt.pie(df['Workclass'].value_counts().values,labels=df['Workclass'].value_counts
```

Out[26]:
```
([<matplotlib.patches.Wedge at 0x292f196c7c0>,
  <matplotlib.patches.Wedge at 0x292f196cf40>,
  <matplotlib.patches.Wedge at 0x292f174b6a0>,
  <matplotlib.patches.Wedge at 0x292f174bdc0>,
  <matplotlib.patches.Wedge at 0x292f1759520>,
  <matplotlib.patches.Wedge at 0x292f1759c40>,
  <matplotlib.patches.Wedge at 0x292f17683a0>,
  <matplotlib.patches.Wedge at 0x292f1768ac0>,
  <matplotlib.patches.Wedge at 0x292f1773220>],
 [Text(-0.6382930911712503, 0.8958693709258343, ' Private'),
  Text(-0.09614160158809627, -1.0957904874765414, ' Self-emp-not-inc'),
  Text(0.3870941794944444, -1.029639789538809, ' Local-gov'),
  Text(0.7406590513513429, -0.8132798839583633, ' ?'),
  Text(0.9491955757802714, -0.5559026523764384, ' State-gov'),
  Text(1.0518455289897355, -0.3219021328669688, ' Self-emp-inc'),
  Text(1.0948632169908754, -0.10618161837338717, ' Federal-gov'),
  Text(1.099995985562683, -0.0029718253619576878, ' Without-pay'),
  Text(1.0999997490710822, -0.0007429963368566515, ' Never-worked')],
 [Text(-0.3481598679115911, 0.48865602050500045, '69.71%'),
  Text(-0.052440873593507055, -0.5977039022599315, '7.80%'),
  Text(0.21114227972424238, -0.5616217033848049, '6.43%'),
  Text(0.40399584619164153, -0.44360720943183446, '5.64%'),
  Text(0.5177430413346934, -0.3032196285689664, '3.98%'),
  Text(0.5737339249034921, -0.17558298156380112, '3.43%'),
  Text(0.5971981183586592, -0.05791724638548391, '2.95%'),
  Text(0.5999978103069179, -0.0016209956519769204, '0.04%'),
  Text(0.5999998631296811, -0.00040527072919453716, '0.02%')])
```

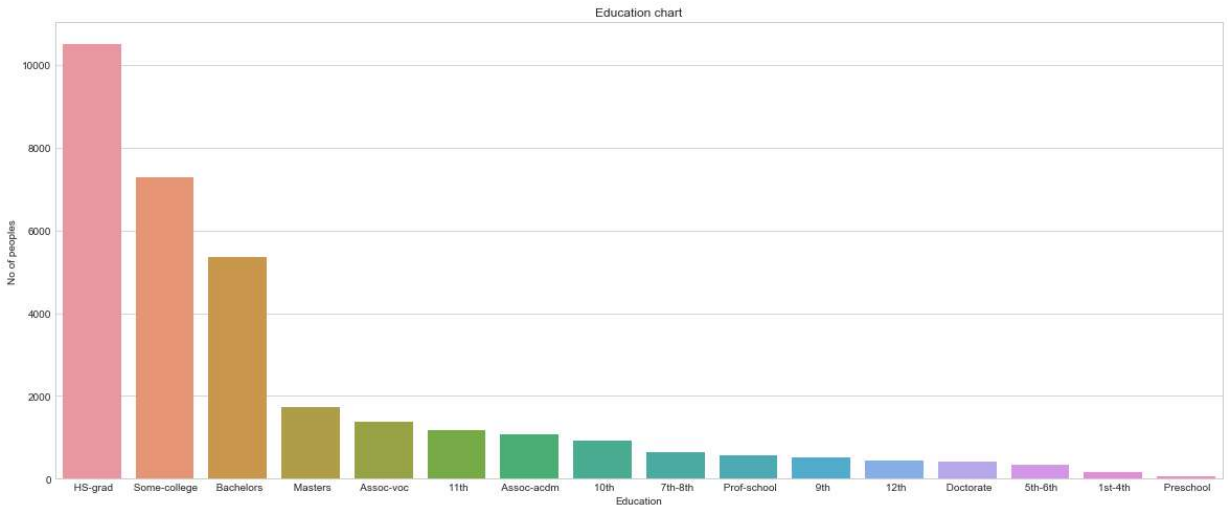


In [27]:
```
# Education
df['Education'].unique()
```

Out[27]:
```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

In [28]:
```
plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(20,8))
sns.countplot(df['Education'],order=df['Education'].value_counts().index)
plt.title('Education chart')
plt.xlabel('Education')
plt.ylabel('No of peoples')
```

Out[28]:
```
Text(0, 0.5, 'No of peoples')
```

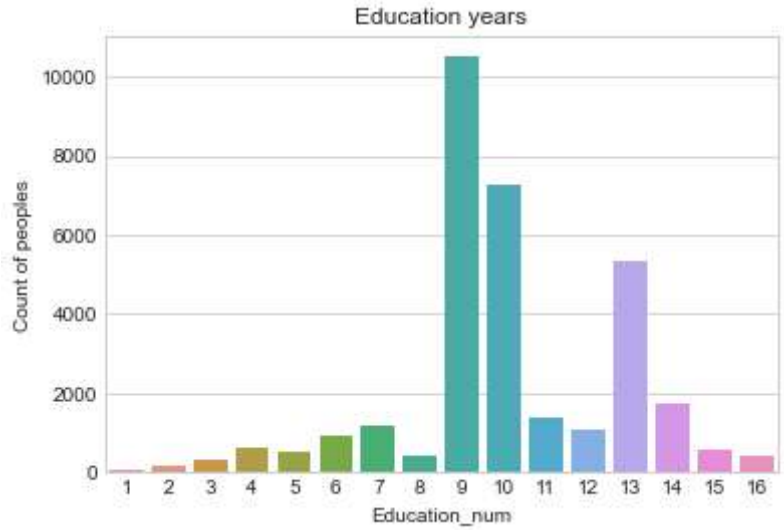


In [29]:
```
df['Education_num'].unique()
```

Out[29]:
```
array([13,  9,  7, 14,  5, 10, 12, 11,  4, 16, 15,  3,  6,  2,  1,  8],
      dtype=int64)
```

In [30]:
```
sns.countplot(df['Education_num'])
plt.title('Education years')
plt.ylabel('Count of peoples')
```

Out[30]:
```
Text(0, 0.5, 'Count of peoples')
```



In [31]:
```
# Maximum peoples have taken 9years education and then 10 years
df.columns
```

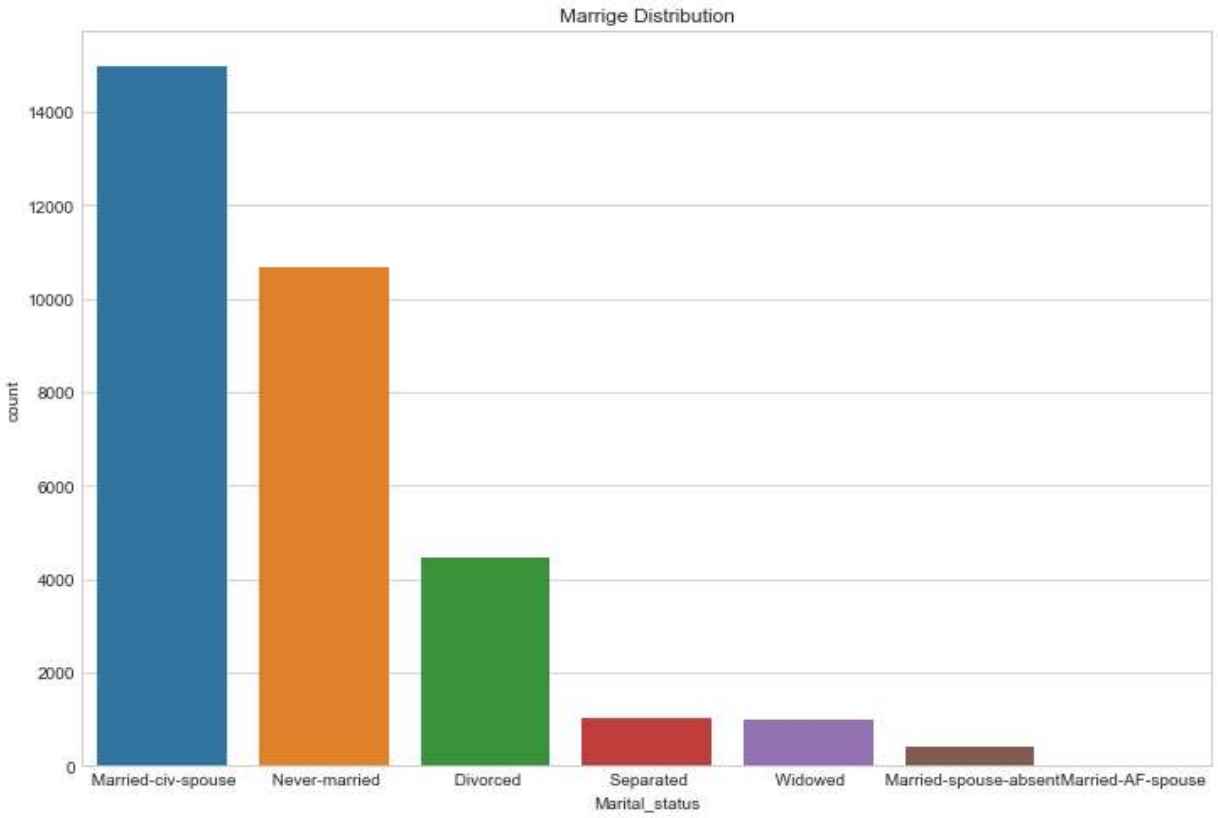
Out[31]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

In [32]:
```
# Marital_status
df['Marital_status'].unique()
```

Out[32]:
```
array([' Married-civ-spouse', ' Divorced', ' Married-spouse-absent',
       ' Never-married', ' Separated', ' Married-AF-spouse', ' Widowed'],
      dtype=object)
```

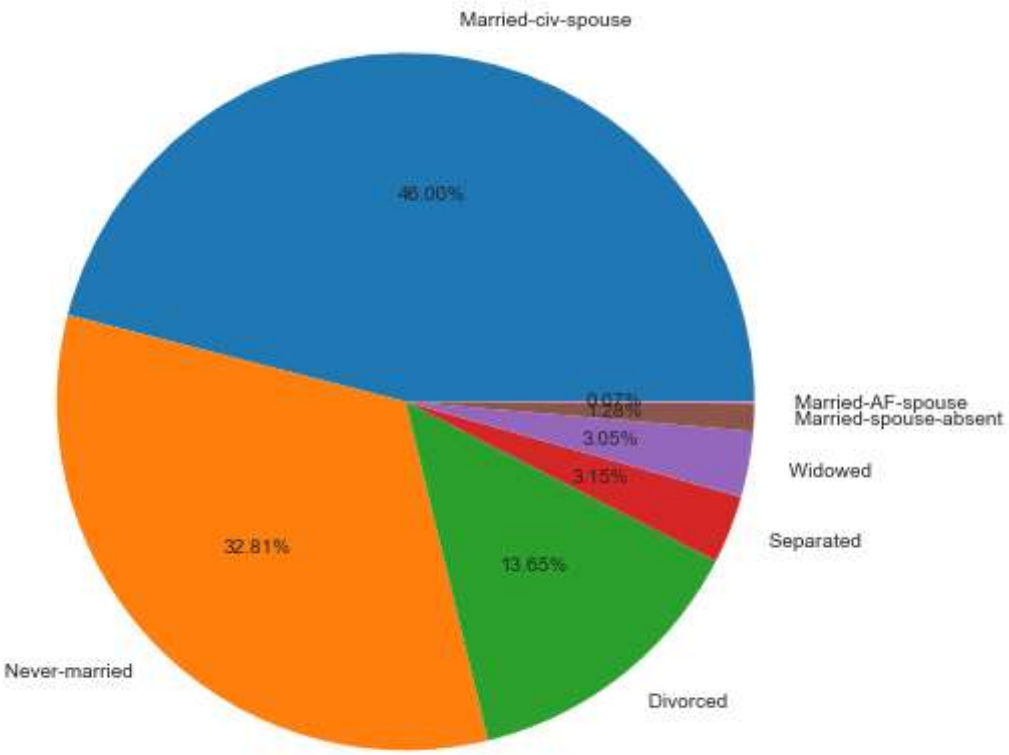
In [33]:
```
plt.figure(figsize=(12,8))
sns.countplot(df['Marital_status'],order=df['Marital_status'].value_counts().inde
plt.title('Marrige Distribution')
```

Out[33]:
```
Text(0.5, 1.0, 'Marrige Distribution')
```



In [34]:
```
plt.figure(figsize=(12,8))
plt.pie(df['Marital_status'].value_counts().values,labels=df['Marital_status'].va
```

Out[34]:
```
([<matplotlib.patches.Wedge at 0x292f19ce070>,
  <matplotlib.patches.Wedge at 0x292f19ce7f0>,
  <matplotlib.patches.Wedge at 0x292f19cef10>,
  <matplotlib.patches.Wedge at 0x292f19d8670>,
  <matplotlib.patches.Wedge at 0x292f19d8d90>,
  <matplotlib.patches.Wedge at 0x292f19e24f0>,
  <matplotlib.patches.Wedge at 0x292f19e2c10>],
 [Text(0.13803501708558066, 1.0913048767682585, ' Married-civ-spouse'),
  Text(-0.7827548151178912, -0.7728485617569305, ' Never-married'),
  Text(0.6810034412960271, -0.8638485474566527, ' Divorced'),
  Text(1.0233080239964072, -0.40353523764916543, ' Separated'),
  Text(1.082048081689935, -0.1979190463576769, ' Widowed'),
  Text(1.0988978136033434, -0.04923002394669188, ' Married-spouse-absent'),
  Text(1.0999972916242746, -0.002440987353635856, ' Married-AF-spouse')],
 [Text(0.07529182750122582, 0.5952572055099591, '46.00%'),
  Text(-0.4269571718824861, -0.4215537609583257, '32.81%'),
  Text(0.37145642252510563, -0.4711901167945378, '13.65%'),
  Text(0.5581680130889494, -0.22011012962681747, '3.15%'),
  Text(0.5902080445581462, -0.10795584346782375, '3.05%'),
  Text(0.5993988074200054, -0.026852740334559207, '1.28%'),
  Text(0.5999985227041498, -0.0013314476474377396, '0.07%')])
```

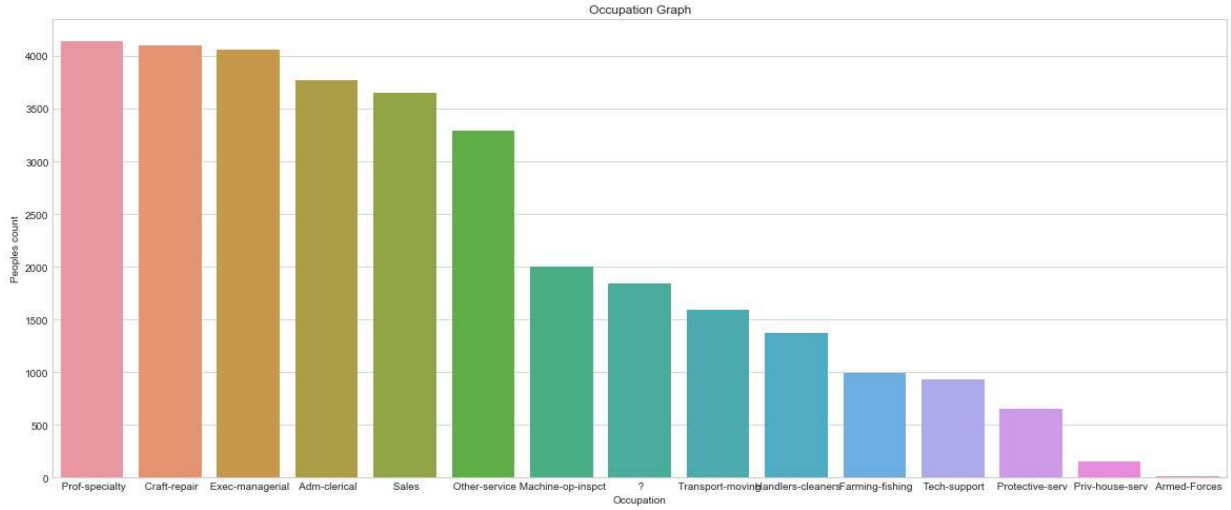


In [35]:
```
# Occupation|
df['Occupation'].unique()
```

Out[35]:
```
array([' Exec-managerial', ' Handlers-cleaners', ' Prof-specialty',
       ' Other-service', ' Adm-clerical', ' Sales', ' Craft-repair',
       ' Transport-moving', ' Farming-fishing', ' Machine-op-inspct',
       ' Tech-support', ' ?', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

In [36]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Occupation'], order=df['Occupation'].value_counts().index)
plt.title('Occupation Graph')
plt.ylabel('Peoples count')
```

Out[36]:
```
Text(0, 0.5, 'Peoples count')
```

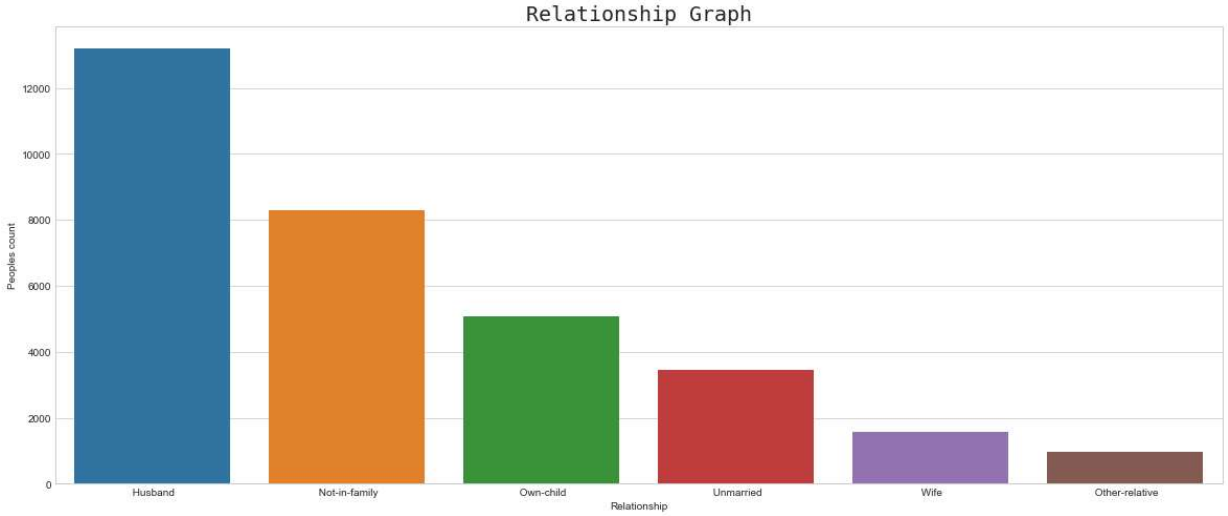


In [37]:
```
# Relationship
df['Relationship'].unique()
```

Out[37]:
```
array([' Husband', ' Not-in-family', ' Wife', ' Own-child', ' Unmarried',
       ' Other-relative'], dtype=object)
```

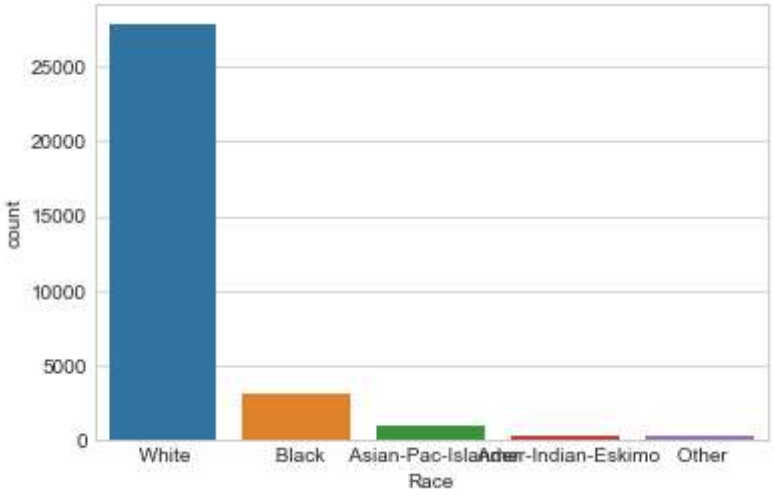
In [38]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Relationship'], order=df['Relationship'].value_counts().index)
plt.title('Relationship Graph',fontdict={'fontname':'monospace','fontsize':20})
plt.ylabel('Peoples count')
```

Out[38]: Text(0, 0.5, 'Peoples count')



In [39]:
```
# Race
sns.countplot(df['Race'])
```

Out[39]: <AxesSubplot:xlabel='Race', ylabel='count'>

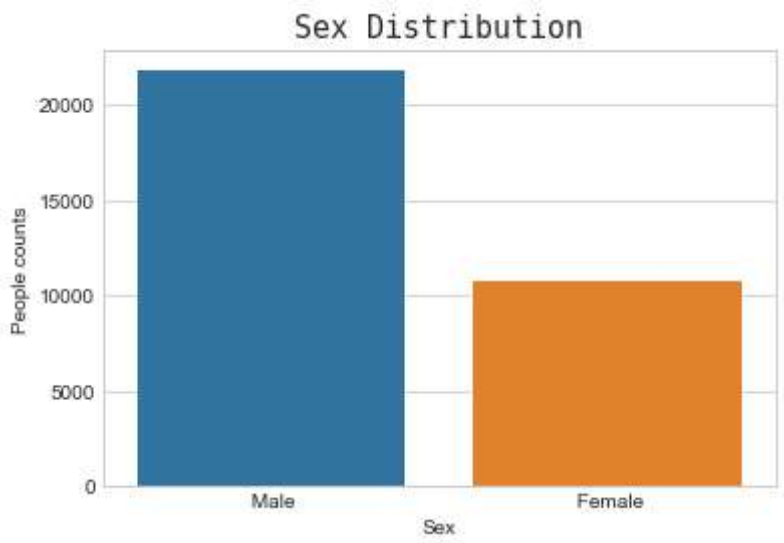


In [40]:
```
df['Race'].value_counts()
```

Out[40]:
```
White                27815
Black                 3124
Asian-Pac-Islander    1039
Amer-Indian-Eskimo     311
Other                  271
Name: Race, dtype: int64
```

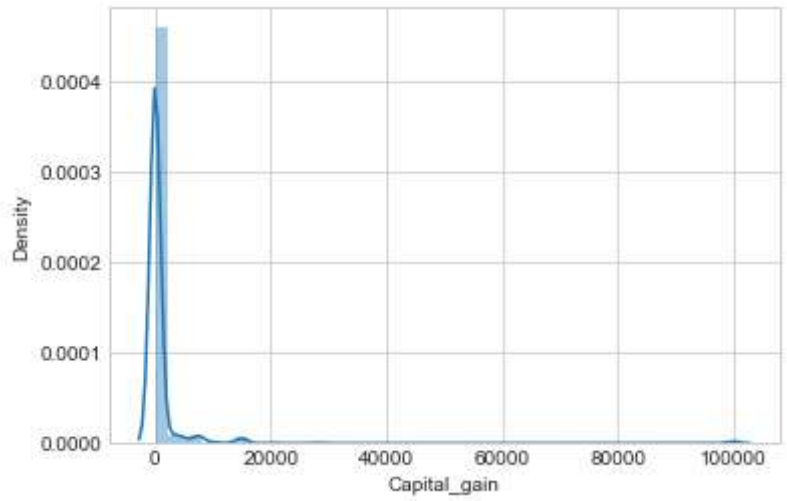
In [41]:
```
sns.countplot(df['Sex'])
plt.title('Sex Distribution',fontdict={'fontname':'monospace','fontsize':15})
plt.ylabel('People counts')
```

Out[41]:
```
Text(0, 0.5, 'People counts')
```



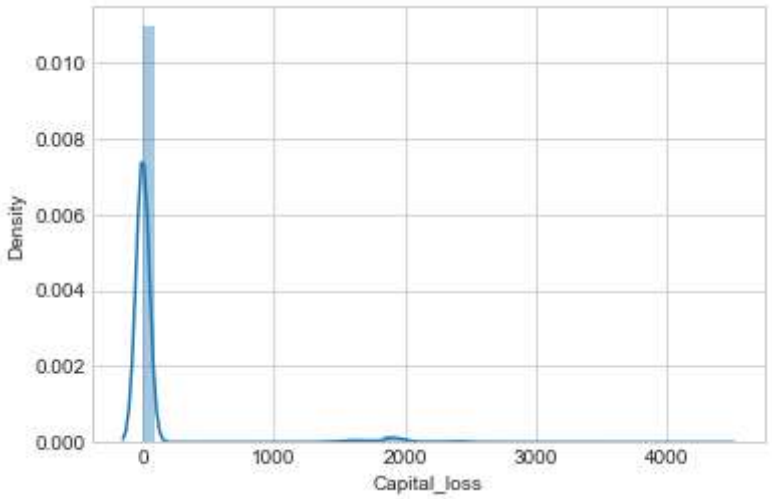
In [42]:
```
# Capital_gain
sns.distplot(df['Capital_gain'])
```

Out[42]:
```
<AxesSubplot:xlabel='Capital_gain', ylabel='Density'>
```



In [43]:
```
# right skewed data
#Capital_loss
sns.distplot(df['Capital_loss'])
```

Out[43]:
```
<AxesSubplot:xlabel='Capital_loss', ylabel='Density'>
```

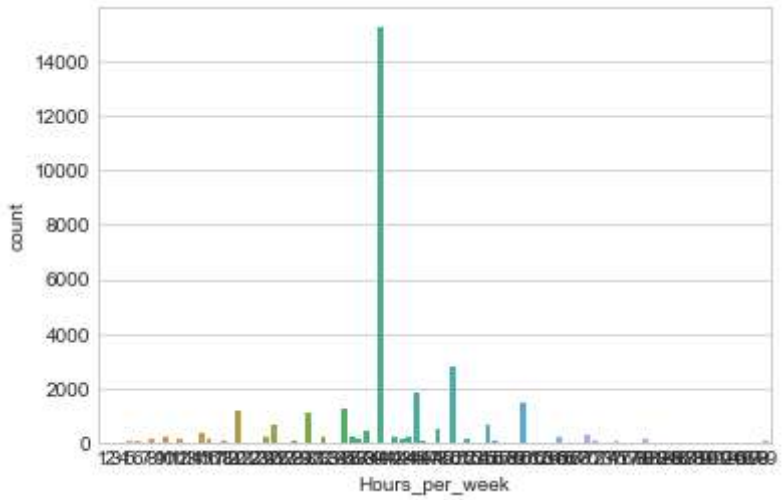


In [44]:
```
# Right skewed data
# Hours_per_week
df['Hours_per_week'].unique()
```

Out[44]:
```
array([13, 40, 16, 45, 50, 80, 30, 35, 60, 20, 52, 44, 15, 25, 38, 43, 55,
       48, 58, 32, 70,  2, 22, 56, 41, 28, 36, 24, 46, 42, 12, 65,  1, 10,
       34, 75, 98, 33, 54,  8,  6, 64, 19, 18, 72,  5,  9, 47, 37, 21, 26,
       14,  4, 59,  7, 99, 53, 39, 62, 57, 78, 90, 66, 11, 49, 84,  3, 17,
       68, 27, 85, 31, 51, 77, 63, 23, 87, 88, 73, 89, 97, 94, 29, 96, 67,
       82, 86, 91, 81, 76, 92, 61, 74, 95], dtype=int64)
```

In [45]:
```
sns.countplot(df['Hours_per_week'])
```

Out[45]:
```
<AxesSubplot:xlabel='Hours_per_week', ylabel='count'>
```

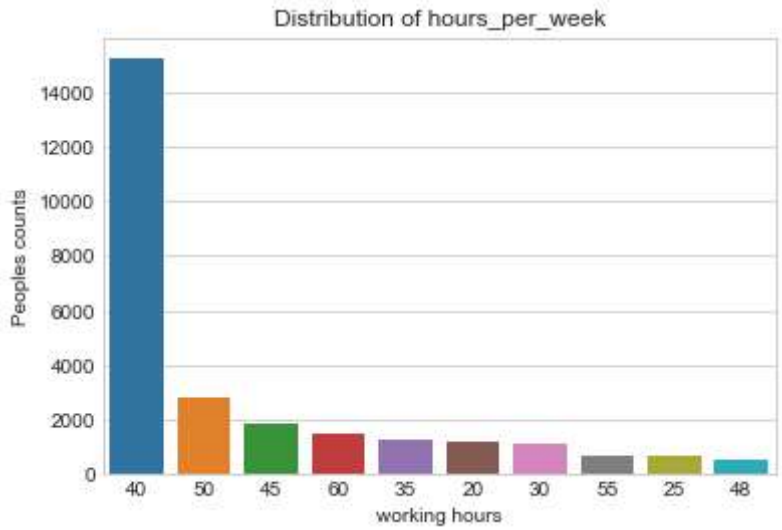


In [46]:
```
hours=df['Hours_per_week'].value_counts().head(10)
hours
```

Out[46]:
```
40    15216
50     2819
45     1824
60     1475
35     1297
20     1224
30     1149
55      694
25      674
48      517
Name: Hours_per_week, dtype: int64
```

In [47]:
```
sns.barplot(hours.index,hours.values,order=hours.index)
plt.title('Distribution of hours_per_week')
plt.ylabel('Peoples counts')
plt.xlabel('working hours')
```

Out[47]:
```
Text(0.5, 0, 'working hours')
```



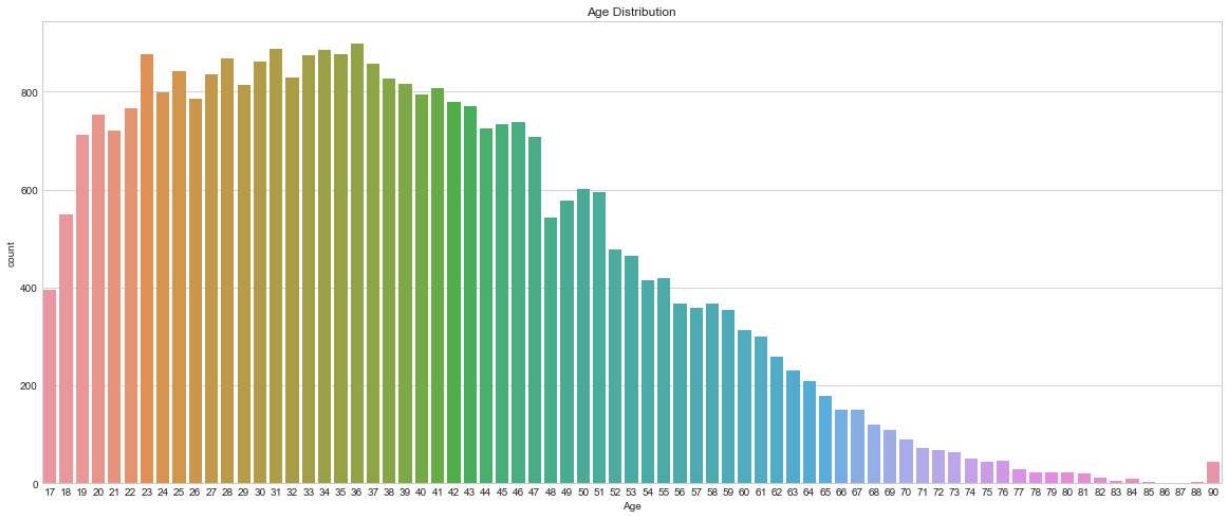
Bivarient analysis

In [48]:
```
df.columns
```

Out[48]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

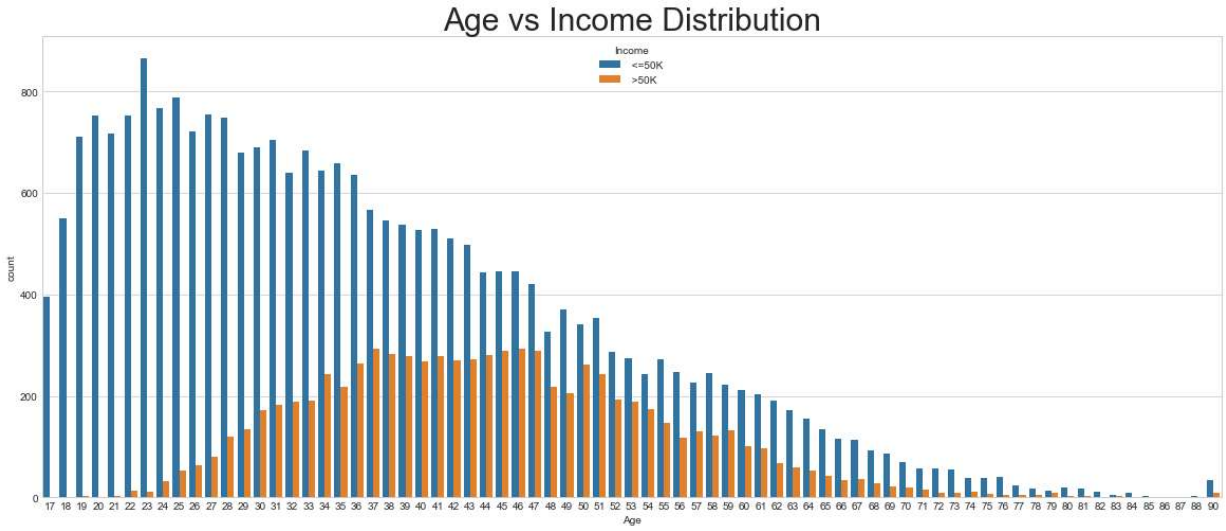
In [49]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Age'])
plt.title('Age Distribution')
```

Out[49]: Text(0.5, 1.0, 'Age Distribution')



In [50]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Age'],hue=df['Income'])
plt.title('Age vs Income Distribution',fontdict={'fontsize':30})
```

Out[50]: Text(0.5, 1.0, 'Age vs Income Distribution')

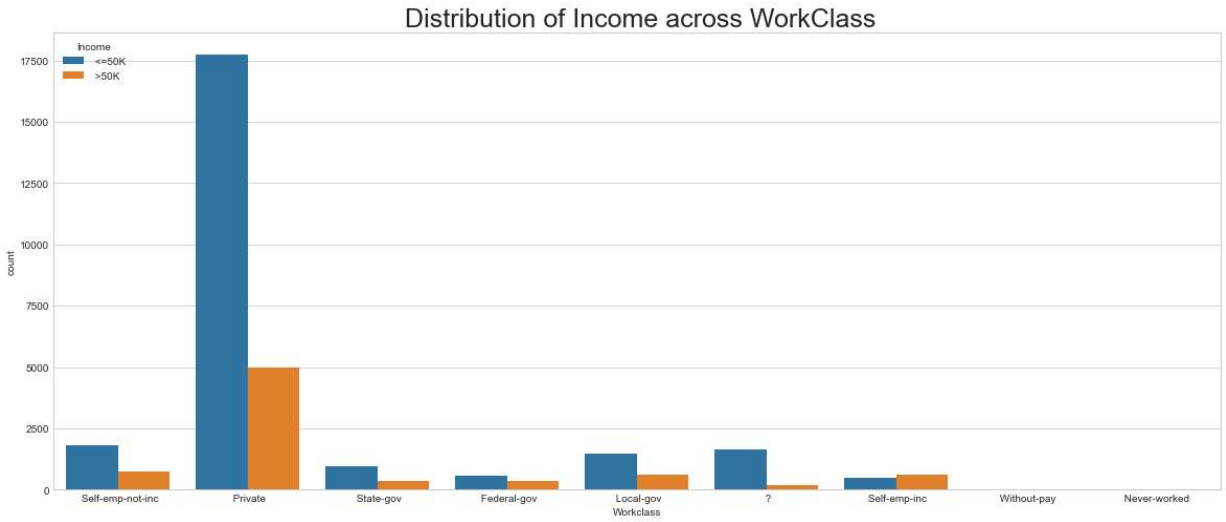


In [51]:
```
# More Peoples between 36 years to 47 years have earned more than 50K
# Peoples started to earn at age 17, from 17 to 41 majority earn below 50k
df.columns
```

Out[51]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

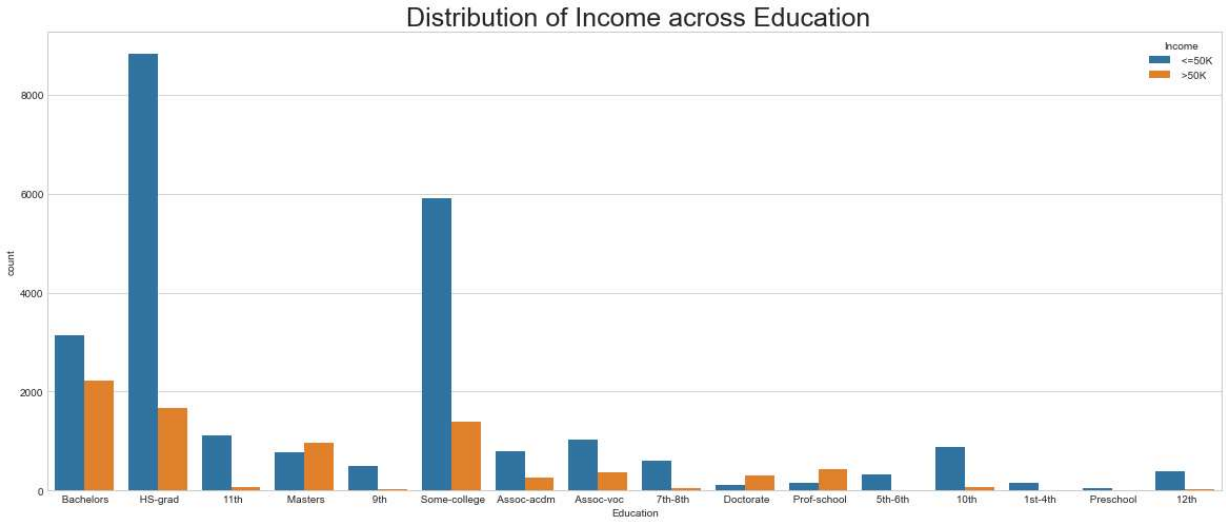
In [52]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Workclass'],hue='Income', data=df)
plt.title(' Distribution of Income across WorkClass',fontdict={'fontsize':25})
```

Out[52]:
```
Text(0.5, 1.0, ' Distribution of Income across WorkClass')
```



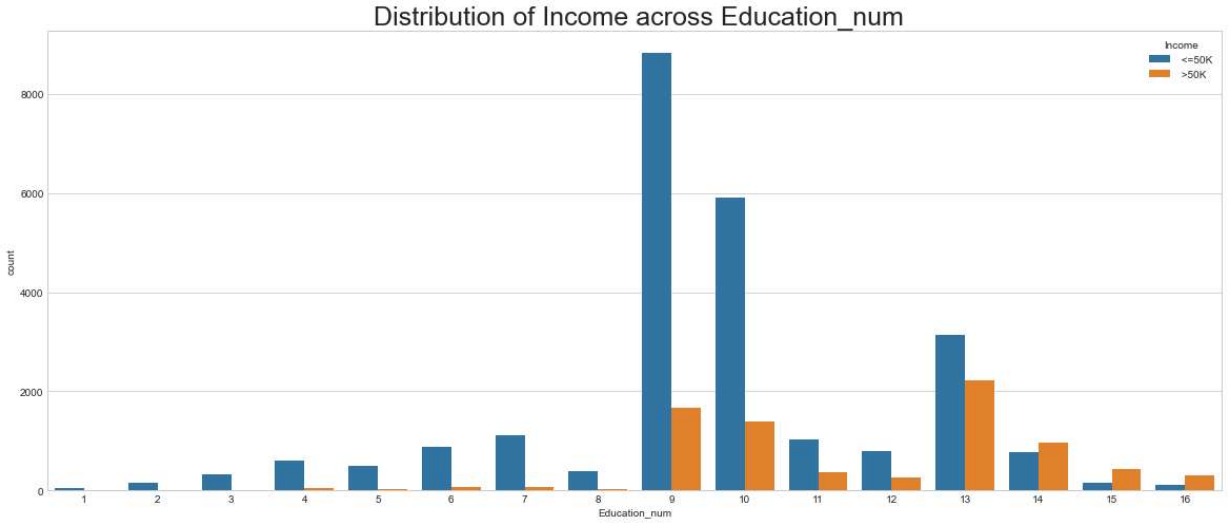
In [53]:
```
# In wvery woork class, most people argeeting income <50k, However selfwmploid pe
plt.figure(figsize=(20,8))
sns.countplot(df['Education'],hue='Income', data=df)
plt.title(' Distribution of Income across Education',fontdict={'fontsize':25})
```

Out[53]:
```
Text(0.5, 1.0, ' Distribution of Income across Education')
```



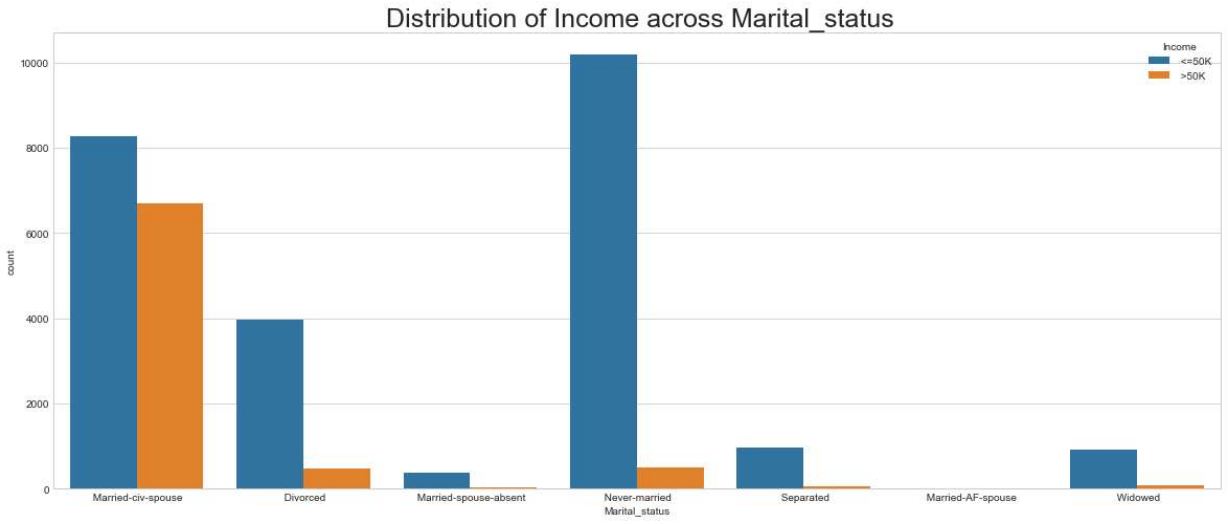
In [54]:
```
# More persons in Doctorate and Prof-School qualified persons got more than >50K
plt.figure(figsize=(20,8))
sns.countplot(df['Education_num'],hue='Income', data=df)
plt.title(' Distribution of Income across Education_num',fontdict={'fontsize':25}
```

Out[54]: Text(0.5, 1.0, ' Distribution of Income across Education_num')



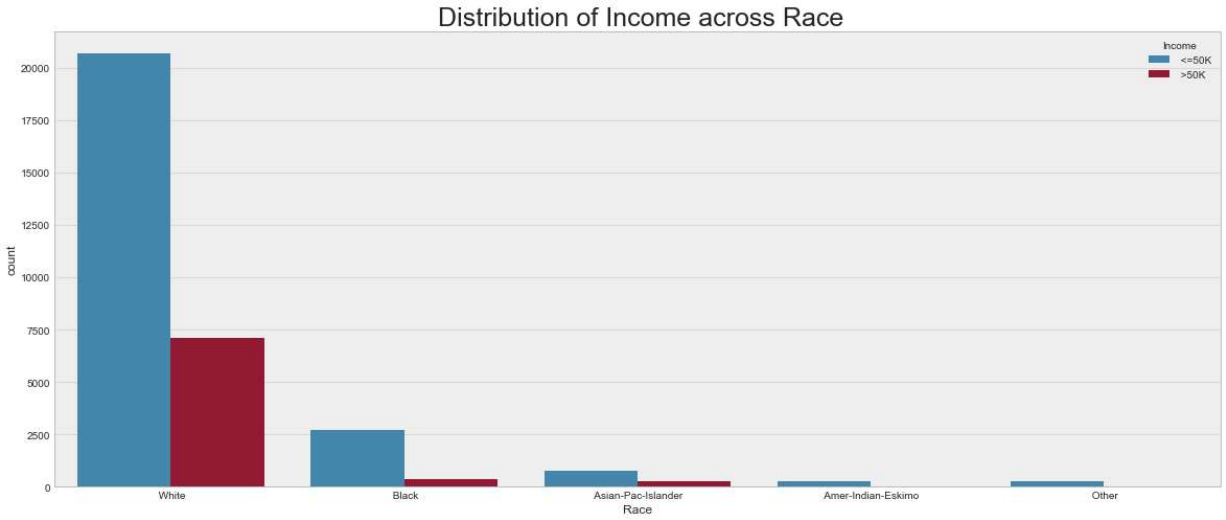
In [55]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Marital_status'],hue='Income', data=df)
plt.title(' Distribution of Income across Marital_status',fontdict={'fontsize':25
```

Out[55]: Text(0.5, 1.0, ' Distribution of Income across Marital_status')



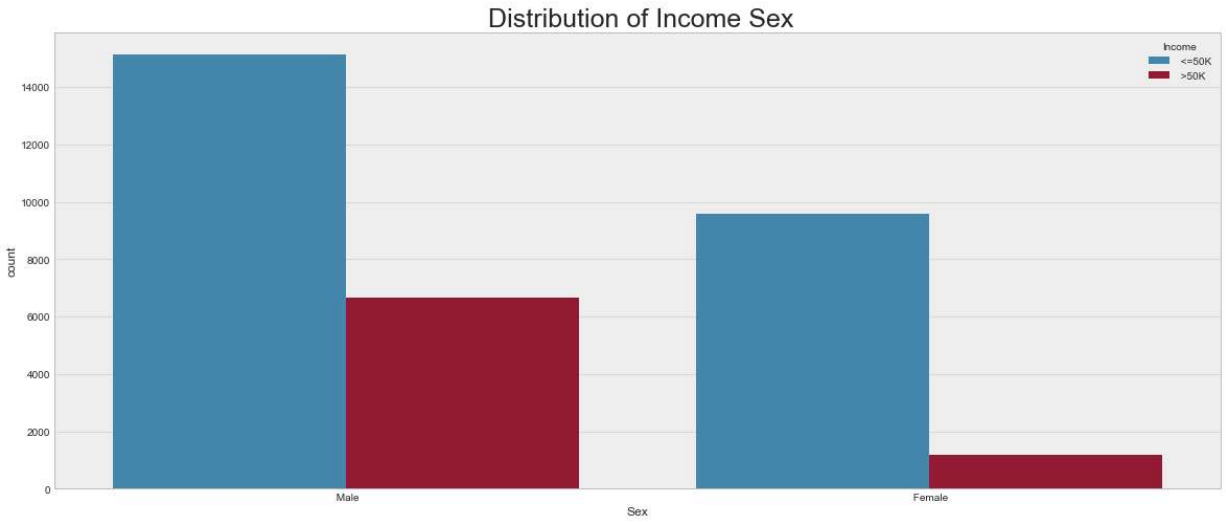
In [56]:
```
plt.style.use('bmh')
plt.figure(figsize=(20,8))
sns.countplot(df['Race'],hue='Income', data=df)
plt.title(' Distribution of Income across Race',fontdict={'fontsize':25})
```

Out[56]: Text(0.5, 1.0, ' Distribution of Income across Race')



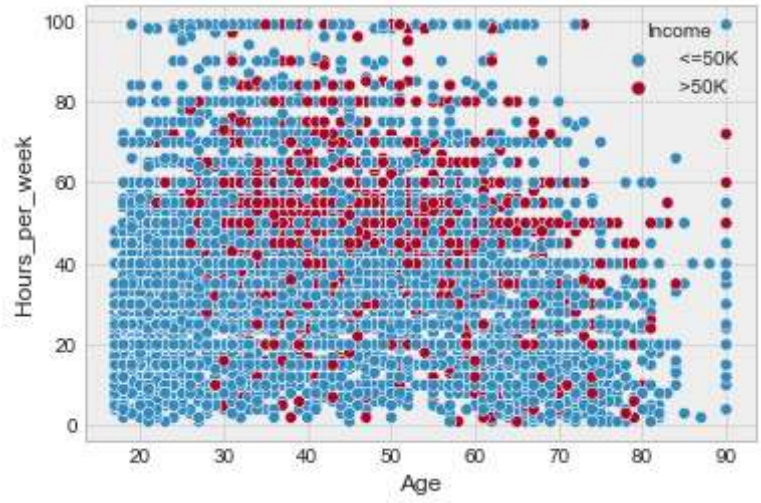
In [57]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Sex'],hue='Income', data=df)
plt.title(' Distribution of Income Sex',fontdict={'fontsize':25})
```

Out[57]: Text(0.5, 1.0, ' Distribution of Income Sex')



In [58]:
```
sns.scatterplot('Age','Hours_per_week',hue='Income', data=df)
```

Out[58]:
```
<AxesSubplot:xlabel='Age', ylabel='Hours_per_week'>
```



In [59]:
```
# After 70yr age, Mostly peoples works toll 60hrs per week
df.columns
```

Out[59]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

In [60]:
```
# Lets analyse correelation with target variable
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
df['Income']=le.fit_transform(df['Income'])
df.head()
```

Out[60]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	I
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	V
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	V
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	I
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	I
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	V

In [61]:
```
# Multivarient Analysis
df.corr()
```

Out[61]:

	Age	Fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_we
Age	1.000000	-0.076646	0.036527	0.077674	0.057775	0.0687
Fnlwgt	-0.076646	1.000000	-0.043159	0.000437	-0.010259	-0.0187
Education_num	0.036527	-0.043159	1.000000	0.122627	0.079932	0.1481
Capital_gain	0.077674	0.000437	0.122627	1.000000	-0.031614	0.0784
Capital_loss	0.057775	-0.010259	0.079932	-0.031614	1.000000	0.0542
Hours_per_week	0.068756	-0.018770	0.148127	0.078409	0.054256	1.0000
Income	0.234039	-0.009481	0.335182	0.223333	0.150523	0.2296

In [62]:
```
# Correlation with target variablessns.heatmap(df.corr(),annot=True,linecolor='wh
# Correlation with target variabel
# Age ahs correlation with income  .23
# Education_num  has correlation .34
# Capital gain has correlation .22
# Hours_per_week has correlation .23


# Final_Weight have low correlation with target feature = -.0095, its very very l
sns.pairplot(df)
```

Out[62]: <seaborn.axisgrid.PairGrid at 0x292f0bf1be0>

In [63]:
```
# here missing values are by ' ?' in categorical fetures,
df.columns
```

Out[63]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

In [64]:
```
df[['Workclass','Occupation','Native_country']]
```

Out[64]:

	Workclass	Occupation	Native_country
0	Self-emp-not-inc	Exec-managerial	United-States
1	Private	Handlers-cleaners	United-States
2	Private	Handlers-cleaners	United-States
3	Private	Prof-specialty	Cuba
4	Private	Exec-managerial	United-States
...	...	...	...
32555	Private	Tech-support	United-States
32556	Private	Machine-op-inspct	United-States
32557	Private	Adm-clerical	United-States
32558	Private	Adm-clerical	United-States
32559	Self-emp-inc	Exec-managerial	United-States

32560 rows × 3 columns

In [65]:
```
df['Workclass'].unique()
```

Out[65]:
```
array([' Self-emp-not-inc', ' Private', ' State-gov', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

In [66]:
```
# Mode replacement
df['Workclass'].mode()
```

Out[66]:
```
0     Private
dtype: object
```

In [67]:
```
df['Workclass'].mode()[0]
```

Out[67]:
```
' Private'
```

In [68]:
```
# replace ? with Nan
df=df.replace(' ?',np.nan)
for i in ['Workclass','Occupation','Native_country']:
    df[i].fillna(df[i].mode()[0], inplace=True)
df.isin([' ?']).sum()
```

Out[68]:
```
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 [69]:
```
df.columns
```

Out[69]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

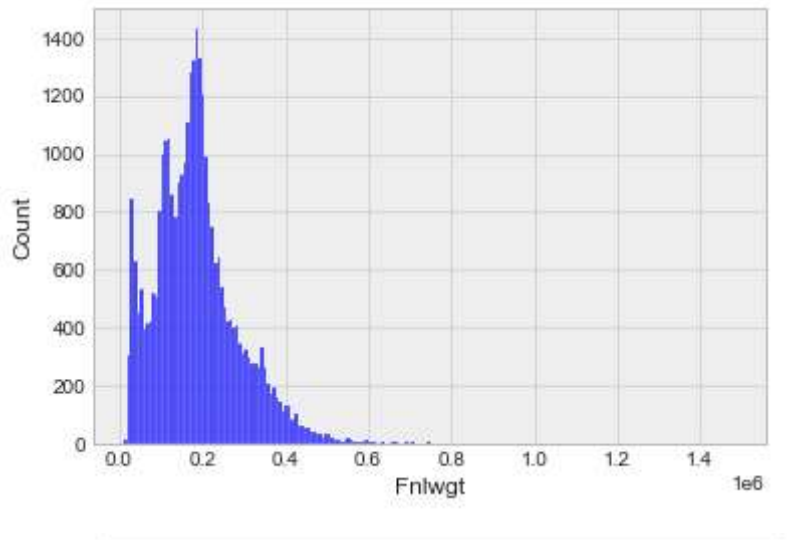
In [70]:
```
num=df.select_dtypes(np.number)
num
```

Out[70]:

	Age	Fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_week	Income
0	50	83311	13	0	0	13	0
1	38	215646	9	0	0	40	0
2	53	234721	7	0	0	40	0
3	28	338409	13	0	0	40	0
4	37	284582	14	0	0	40	0
...	...	...	...	...	...	...	...
32555	27	257302	12	0	0	38	0
32556	40	154374	9	0	0	40	1
32557	58	151910	9	0	0	40	0
32558	22	201490	9	0	0	20	0
32559	52	287927	9	15024	0	40	1

32560 rows × 7 columns

In [71]:
```
for i in num:
    sns.histplot(df[i])
    plt.show()
```



Label Encoding for categorical features

In [72]:
```
from sklearn.preprocessing import LabelEncoder
df.columns
```

Out[72]:
```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

In [73]:
```
for i in df.columns:
    if df[i].dtypes =='object':
        le=LabelEncoder()
        df[i]= le.fit_transform(df[i])
df
```

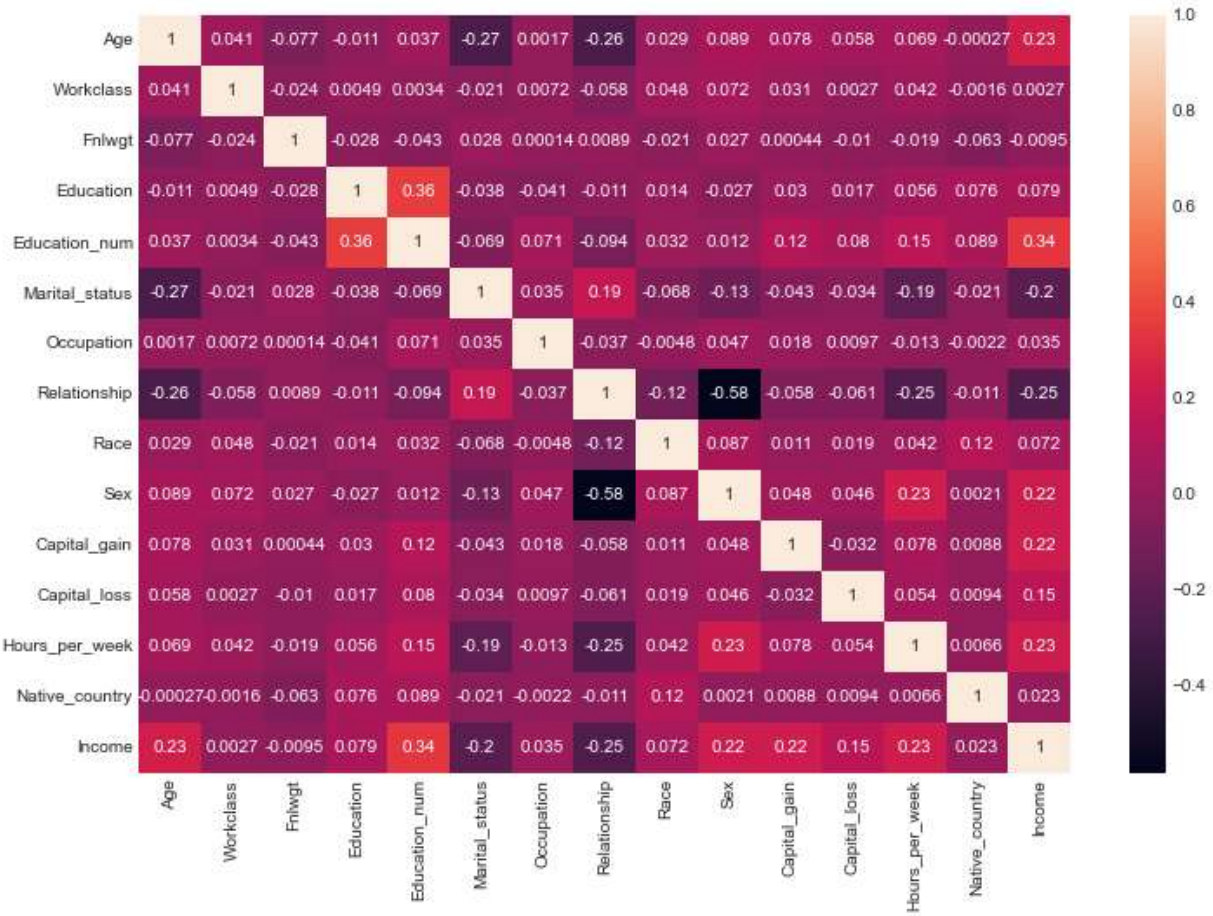
Out[73]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationsh
0	50	5	83311	9	13	2	3	
1	38	3	215646	11	9	0	5	
2	53	3	234721	1	7	2	5	
3	28	3	338409	9	13	2	9	
4	37	3	284582	12	14	2	3	
...	...	...	...	...	...	...	...	...
32555	27	3	257302	7	12	2	12	
32556	40	3	154374	11	9	2	6	
32557	58	3	151910	11	9	6	0	
32558	22	3	201490	11	9	4	0	
32559	52	4	287927	11	9	2	3	

32560 rows × 15 columns

In [74]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True)
```

Out[74]: <AxesSubplot:>



In [75]:
```
# Correlation after encoding with Target variable
# Few features are  very low correlated with target variable,
```

Split independent and dependent features

In [76]:
```
X=df.drop('Income', axis=1)
Y=df['Income']
X.shape , Y.shape
```

Out[76]:
```
((32560, 14), (32560,))
```

Extra Tree Classifier for Feature Selection

In [77]:
```
from sklearn.ensemble import ExtraTreesClassifier
select= ExtraTreesClassifier()
select.fit(X,Y)
```

Out[77]:
```
ExtraTreesClassifier

ExtraTreesClassifier()
```

In [78]:
```
add=select.feature_importances_
for index,feature_imp in enumerate(select.feature_importances_):
    print(index , feature_imp)
```

```
0 0.15796314003769335
1 0.040371737438582335
2 0.16744520870115578
3 0.03639106657985906
4 0.08905098907177159
5 0.067247716891538
6 0.07417452110847161
7 0.09235366117090354
8 0.014692200465694662
9 0.03121976818114038
10 0.09005208555018336
11 0.028661378622763488
12 0.09520388285343112
13 0.01517264332681178
```

In [79]:
```
imp_feature=pd.DataFrame()
imp_feature['Feature']= X.columns
imp_feature['Values']=select.feature_importances_
```

In [80]:
```
imp_feature.sort_values('Values',ascending=False)
```

Out[80]:

	Feature	Values
2	Fnlwgt	0.167445
0	Age	0.157963
12	Hours_per_week	0.095204
7	Relationship	0.092354
10	Capital_gain	0.090052
4	Education_num	0.089051
6	Occupation	0.074175
5	Marital_status	0.067248
1	Workclass	0.040372
3	Education	0.036391
9	Sex	0.031220
11	Capital_loss	0.028661
13	Native_country	0.015173
8	Race	0.014692

In [81]:
```
16.7+15.7+9.5+8.8+8.8+8.7+7.6+7.3+4.1+3.7+2.8+2.7
```

Out[81]: 96.39999999999999

In [82]:
```
# 12 features explanes the variation 96.3%, we can drop last 2 features
X=X.drop(['Native_country','Race'], axis=1)
X.head()
```

Out[82]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	S
0	50	5	83311	9	13	2	3	0	
1	38	3	215646	11	9	0	5	1	
2	53	3	234721	1	7	2	5	0	
3	28	3	338409	9	13	2	9	5	
4	37	3	284582	12	14	2	3	5	

Scaling()

In [83]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

In [84]:
```
X=sc.fit_transform(X)
```

In [ ]:
```

```

In [85]:
```
pd.DataFrame(X)
```

Out[85]:

	0	1	2	3	4	5	6	7	
0	0.837097	1.721336	-1.008742	-0.335443	1.134779	-0.406183	-0.790156	-0.900177	0.703
1	-0.042640	-0.085223	0.245046	0.181319	-0.420027	-1.734026	-0.286703	-0.277810	0.703
2	1.057031	-0.085223	0.425770	-2.402489	-1.197429	-0.406183	-0.286703	-0.900177	0.703
3	-0.775755	-0.085223	1.408146	-0.335443	1.134779	-0.406183	0.720204	2.211658	-1.422
4	-0.115952	-0.085223	0.898170	0.439700	1.523480	-0.406183	-0.790156	2.211658	-1.422
...	...	...	...	...	...	...	...	...	
32555	-0.849066	-0.085223	0.639710	-0.852204	0.746077	-0.406183	1.475385	2.211658	-1.422
32556	0.103982	-0.085223	-0.335466	0.181319	-0.420027	-0.406183	-0.034976	-0.900177	0.703
32557	1.423589	-0.085223	-0.358811	0.181319	-0.420027	2.249503	-1.545336	1.589291	-1.422
32558	-1.215624	-0.085223	0.110927	0.181319	-0.420027	0.921660	-1.545336	0.966924	0.703
32559	0.983720	0.818056	0.929862	0.181319	-0.420027	-0.406183	-0.790156	2.211658	-1.422

32560 rows × 12 columns

Balance Dataset

In [86]:
```
from imblearn.over_sampling import SMOTE
smt=SMOTE()
Y.value_counts()*100
```

Out[86]:
```
0    2471900
1     784100
Name: Income, dtype: int64
```

In [87]:
```
Y.value_counts(normalize=True)*100
```

Out[87]:
```
0    75.918305
1    24.081695
Name: Income, dtype: float64
```

In [88]:
```
balanced_X,balanced_Y=smt.fit_resample(X,Y)
balanced_X.shape  , balanced_Y.shape
```

Out[88]:
```
((49438, 12), (49438,))
```

In [89]:
```
balanced_Y.value_counts(normalize=True)*100
```

Out[89]:
```
0    50.0
1    50.0
Name: Income, dtype: float64
```

Train-Test Split

In [90]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [91]:
```
x_train,x_test,y_train,y_test= train_test_split(balanced_X,balanced_Y,random_stat
```

In [92]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import RidgeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [93]:
```
LR_model= LogisticRegression()
RD_model= RidgeClassifier()
DT_model= DecisionTreeClassifier()
SV_model= SVC()
KNR_model= KNeighborsClassifier()
RFR_model= RandomForestClassifier()
XGB_model= XGBClassifier()
SGH_model= SGDClassifier()
Bag_model=BaggingClassifier()
ADA_model=AdaBoostClassifier()
GB_model= GradientBoostingClassifier()

model=[LR_model,RD_model,DT_model,SV_model,KNR_model,RFR_model,XGB_model,SGH_mode
```

In [94]:
```
for m in model:
    m.fit(x_train,y_train)
    m.score(x_train,y_train)
    pred= m.predict(x_test)
    print('Accuracy_Score of ',m, 'is', accuracy_score(y_test,pred)*100)
    print("F1 SCore                ", f1_score(y_test,pred)*100)
    print('Confusion Matrix of ',m,' is \n', confusion_matrix(y_test,pred) )
    print(classification_report(y_test,pred))
    print('*'*50)
```

```
Accuracy_Score of  LogisticRegression() is 77.14401294498381
F1 SCore                76.93877551020408
Confusion Matrix of  LogisticRegression()  is
 [[5787 1665]
 [1725 5655]]
              precision    recall  f1-score   support

           0       0.77      0.78      0.77      7452
           1       0.77      0.77      0.77      7380

    accuracy                           0.77     14832
   macro avg       0.77      0.77      0.77     14832
weighted avg       0.77      0.77      0.77     14832

**************************************************
Accuracy_Score of  RidgeClassifier() is 75.903451995685
F1 SCore                76.3436589886153
Confusion Matrix of  RidgeClassifier()  is
 [[5491 1961]
```

Cross Validation

In [95]:
```
from sklearn.model_selection import cross_val_score
```

In [96]:
```
for i in model:
    print('Accuracy_Score of ',i, 'is', accuracy_score(y_test,i.predict(x_test))*
    print("cross Validation accuracy score of ",i ," is ",cross_val_score(i,balar
    print('*'*50)
```

```
Accuracy_Score of  LogisticRegression() is 77.14401294498381
cross Validation accuracy score of  LogisticRegression()  is  76.97723617492554
**************************************************
Accuracy_Score of  RidgeClassifier() is 75.903451995685
cross Validation accuracy score of  RidgeClassifier()  is  75.85057897772336
**************************************************
Accuracy_Score of  DecisionTreeClassifier() is 85.03910463861921
cross Validation accuracy score of  DecisionTreeClassifier()  is  85.2462971136
9543
**************************************************
Accuracy_Score of  SVC() is 82.7467637540453
cross Validation accuracy score of  SVC()  is  83.3164984797467
**************************************************
Accuracy_Score of  KNeighborsClassifier() is 85.09304207119742
cross Validation accuracy score of  KNeighborsClassifier()  is  86.099804817086
8
**************************************************
Accuracy_Score of  RandomForestClassifier() is 89.81256742179072
cross Validation accuracy score of  RandomForestClassifier()  is  89.7954149281
7053
**************************************************
Accuracy_Score of  XGBClassifier(base_score=0.5, booster='gbtree', callbacks=No
ne,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...) is 90.48678532901833
cross Validation accuracy score of  XGBClassifier(base_score=0.5, booster='gbtr
ee', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)  is  89.3019461337057
**************************************************
Accuracy_Score of  SGDClassifier() is 76.94174757281553
cross Validation accuracy score of  SGDClassifier()  is  77.10264663840557
**************************************************
Accuracy_Score of  BaggingClassifier() is 88.39670981661273
cross Validation accuracy score of  BaggingClassifier()  is  88.41189646075082
**************************************************
Accuracy_Score of  AdaBoostClassifier() is 84.86380798274003
```

```
cross Validation accuracy score of  AdaBoostClassifier()  is  84.55848253549902
**********************************************
Accuracy_Score of  GradientBoostingClassifier() is 86.35382955771306
cross Validation accuracy score of  GradientBoostingClassifier()  is  86.051287
72278856
**********************************************
```

Hypertuning

In [97]:
```
n_estimator= [200, 400, 600, 800, 1000]  # no of tree in Random forest, default 1
max_features= ['auto','sqrt','log2']  # mini no of features to create Decission T
max_depth=[10, 64, 118, 173, 227, 282, 336]  #  Max depth of decision tree
min_samples_split= [1,2,3]   # mini no of sample rerquired to split node
min_samples_leaf= [1,3,4,6,7,9]  #mini no of sample required at each leaf node

param_grid= {'n_estimators': n_estimator,
             'max_features':max_features,
             'max_depth':max_depth,
             'min_samples_split':min_samples_split,
             'min_samples_leaf':min_samples_leaf,
             }
param_grid
```

Out[97]:
```
{'n_estimators': [200, 400, 600, 800, 1000],
 'max_features': ['auto', 'sqrt', 'log2'],
 'max_depth': [10, 64, 118, 173, 227, 282, 336],
 'min_samples_split': [1, 2, 3],
 'min_samples_leaf': [1, 3, 4, 6, 7, 9]}
```

In [98]:
```
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV
```

In [99]:
```
random= RandomizedSearchCV(RFR_model,param_distributions=param_grid,cv=5,n_jobs=-
random.fit(x_train,y_train)
```

Out[99]:

▸ RandomizedSearchCV

▸ estimator: RandomForestClassifier

▸ RandomForestClassifier

In [100]:
```
random.best_estimator_
```

Out[100]:

▾ RandomForestClassifier

RandomForestClassifier(max_depth=227, min_samples_split=3, n_estimators=1000)

In [101]:
```
random.best_params_
```

Out[101]:
```
{'n_estimators': 1000,
 'min_samples_split': 3,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 227}
```

In [102]:
```
print('Random Forest Classifier:')
print('Accuracy score:', round(accuracy_score(y_test,pred) * 100, 2))
print('F1 score:', round(f1_score(y_test,pred) * 100, 2))
```

```
Random Forest Classifier:
Accuracy score: 86.35
F1 score: 86.7
```

In [103]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,pred)
```

Out[103]:
```
array([[6210, 1242],
       [ 782, 6598]], dtype=int64)
```

In [104]:
```
sns.heatmap(confusion_matrix(y_test,pred),annot=True, fmt='d')
```

Out[104]:
```
<AxesSubplot:>
```

In [105]:
```
print(classification_report(y_test,pred))
```

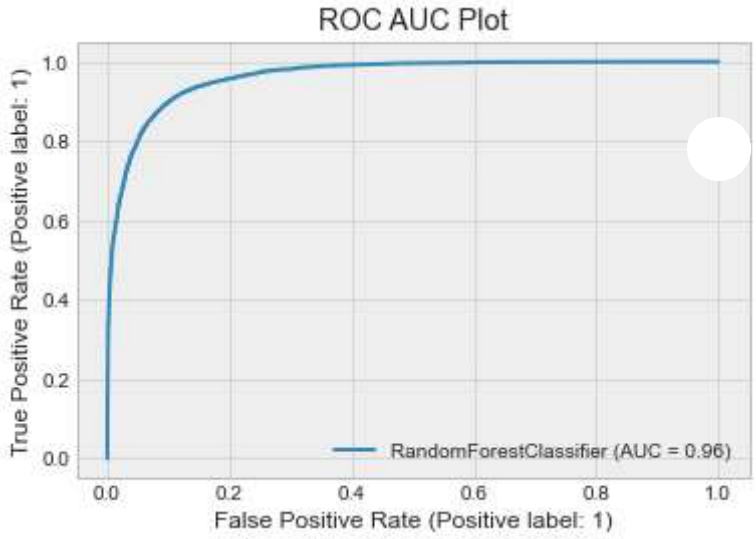
```
              precision    recall  f1-score   support

           0       0.89      0.83      0.86      7452
           1       0.84      0.89      0.87      7380

    accuracy                           0.86     14832
   macro avg       0.86      0.86      0.86     14832
weighted avg       0.86      0.86      0.86     14832
```

In [106]:
```
from sklearn.metrics import roc_auc_score,roc_curve,plot_roc_curve
```

In [ ]:

In [107]:
```
plot_roc_curve(random.best_estimator_,x_test,y_test)
plt.title('ROC AUC Plot')
```

Out[107]:
```
Text(0.5, 1.0, 'ROC AUC Plot')
```



Model accuracy is 89.31 while AUC is 96%

In [108]:
```
# Save the Model
import joblib
joblib.dump(random.best_estimator_,'Income_Prediction.pkl')
```

Out[108]:
```
['Income_Prediction.pkl']
```

In [ ]: