Yutong Wang_Frequent Itemset Mining Project

October 15, 2020

1 Data Cleaning and EDA

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
import resource
```

In [2]:
```
#Load data from uci
df = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult
                 sep = ',', header = None,
                 names = ['Age', 'Workclass', 'Fnlwgt', 'Education', 'Education_num'
                          'Marital_status', 'Occupation', 'Relationship', 'Race', 'Se
                          'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_c
                          'Income'])
df.head()
```

Out[2]:

	Age	Workclass	Fnlwgt	Education	Education_num	\
0	39	State-gov	77516	Bachelors	13	
1	50	Self-emp-not-inc	83311	Bachelors	13	
2	38	Private	215646	HS-grad	9	
3	53	Private	234721	11th	7	
4	28	Private	338409	Bachelors	13	

	Marital_status	Occupation	Relationship	Race	Sex	\
0	Never-married	Adm-clerical	Not-in-family	White	Male	
1	Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	Divorced	Handlers-cleaners	Not-in-family	White	Male	
3	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
4	Married-civ-spouse	Prof-specialty	Wife	Black	Female	

	Capital_gain	Capital_loss	Hours_per_week	Native_country	Income
0	2174	0	40	United-States	<=50K
1	0	0	13	United-States	<=50K
2	0	0	40	United-States	<=50K
3	0	0	40	United-States	<=50K
4	0	0	40	Cuba	<=50K

```
In [3]: #preview the size of data
        print(df.shape)
```

```
(32561, 15)
```

```
In [4]: #Get the info of the df: column type, number of valid data
        df[df == ' ?'] = np.nan
        print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
Age               32561 non-null int64
Workclass         30725 non-null object
Fnlwgt            32561 non-null int64
Education         32561 non-null object
Education_num     32561 non-null int64
Marital_status    32561 non-null object
Occupation        30718 non-null object
Relationship      32561 non-null object
Race              32561 non-null object
Sex               32561 non-null object
Capital_gain      32561 non-null int64
Capital_loss      32561 non-null int64
Hours_per_week    32561 non-null int64
Native_country    31978 non-null object
Income            32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
None
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/pandas/core/ops.p
  result = method(y)
```

```
In [5]: #Preview the number of missing value
        print(df.isnull().sum())
```

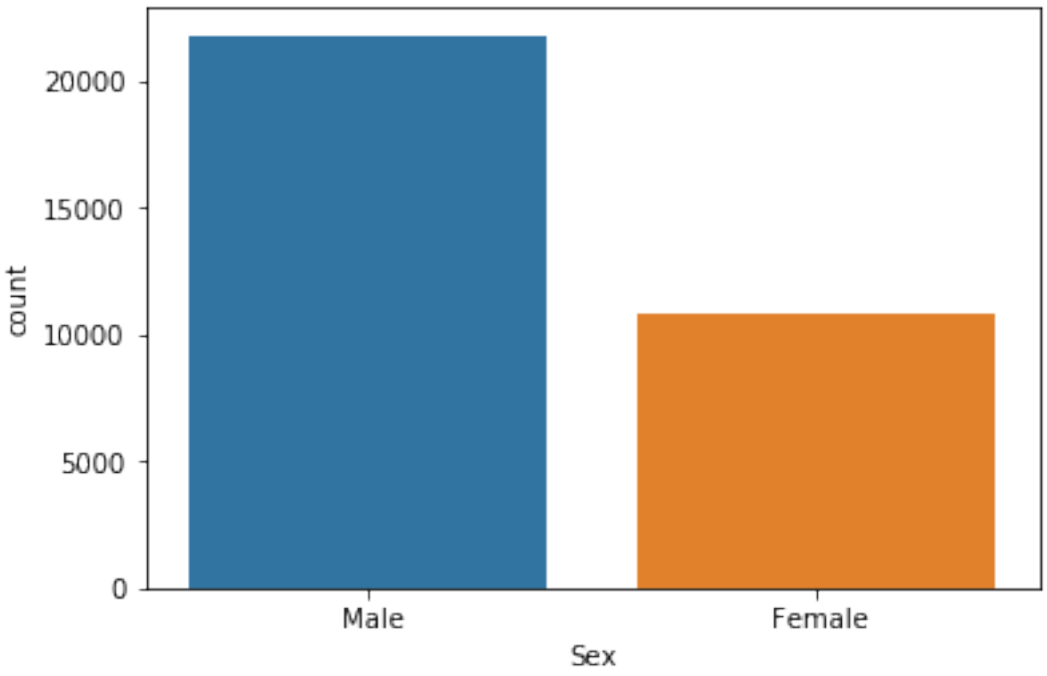
```
Age                  0
Workclass         1836
Fnlwgt               0
Education            0
Education_num        0
Marital_status       0
Occupation        1843
Relationship         0
Race                 0
```

```
Sex                0
Capital_gain       0
Capital_loss       0
Hours_per_week     0
Native_country   583
Income             0
dtype: int64
```

2 Data visualization

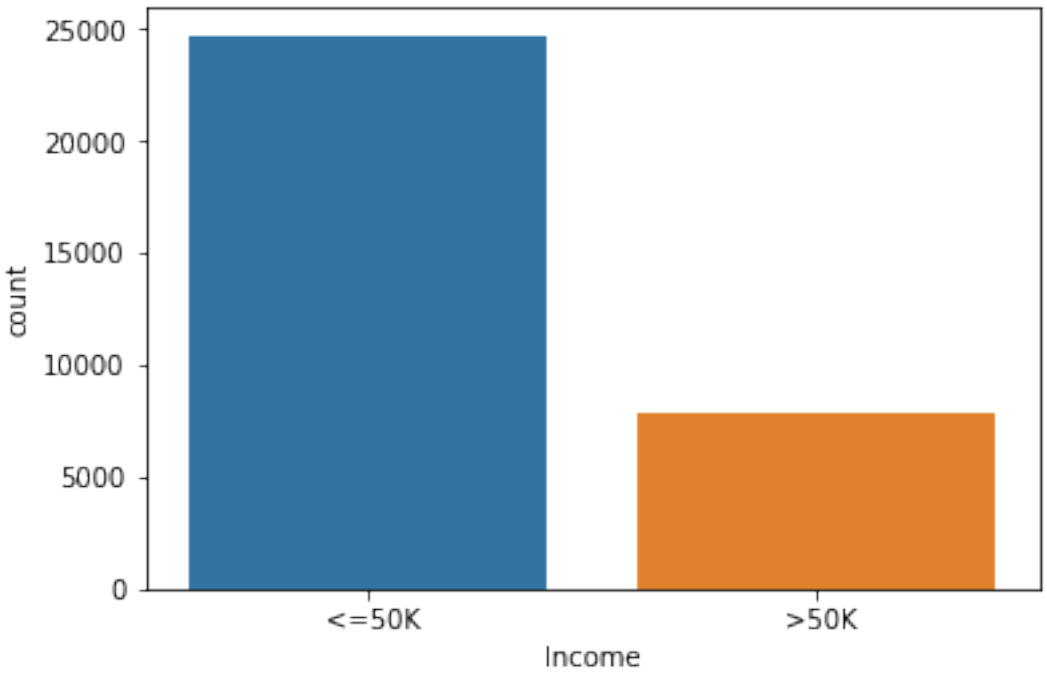
```
In [6]: sns.countplot(x='Sex', data=df)
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x12b567518>
```



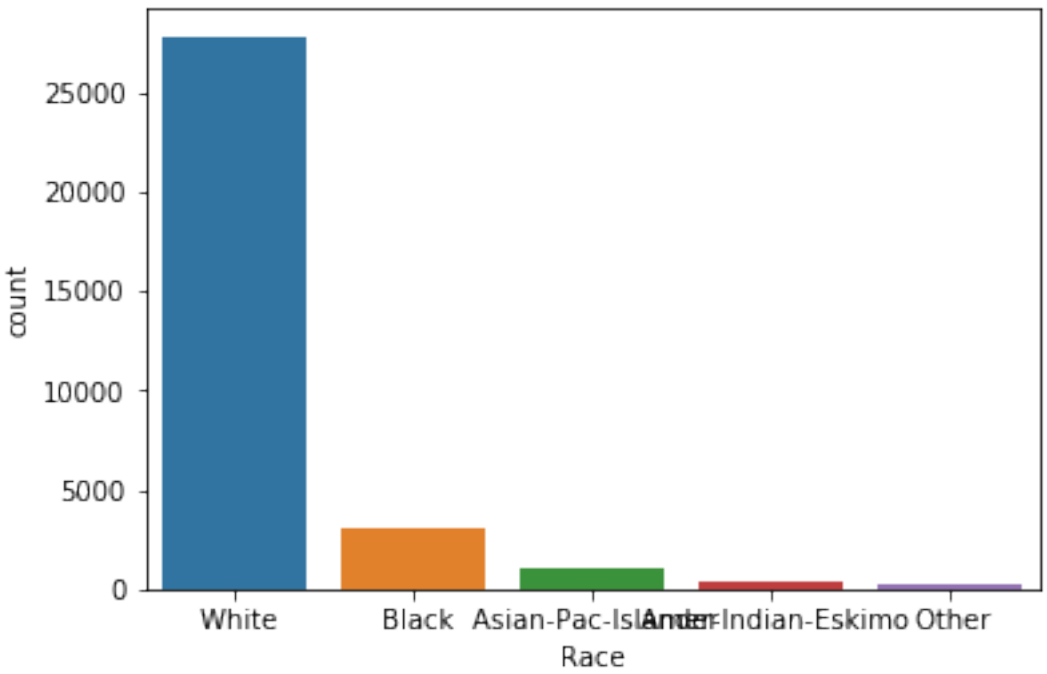

```
In [538]: sns.countplot(x='Income', data=df)
```

```
Out[538]: <matplotlib.axes._subplots.AxesSubplot at 0x12fd50710>
```

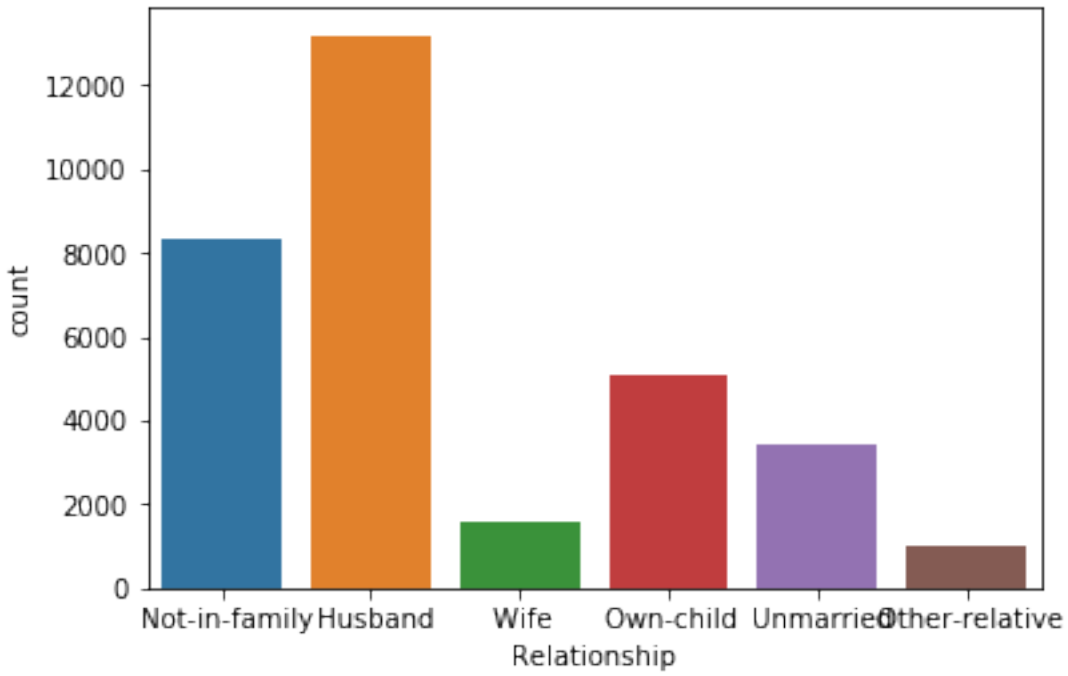



```
In [7]: sns.countplot(x='Race', data=df)
```

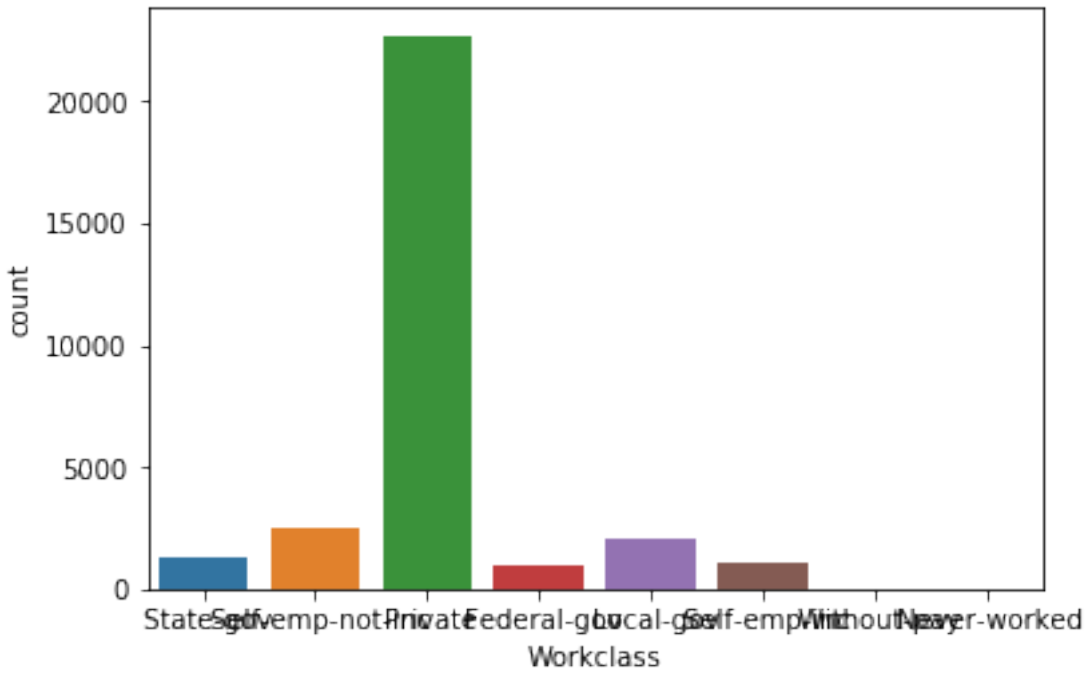
```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x12b453b70>
```



```
In [536]: sns.countplot(x='Relationship', data=df)

Out[536]: <matplotlib.axes._subplots.AxesSubplot at 0x12fd915f8>
```

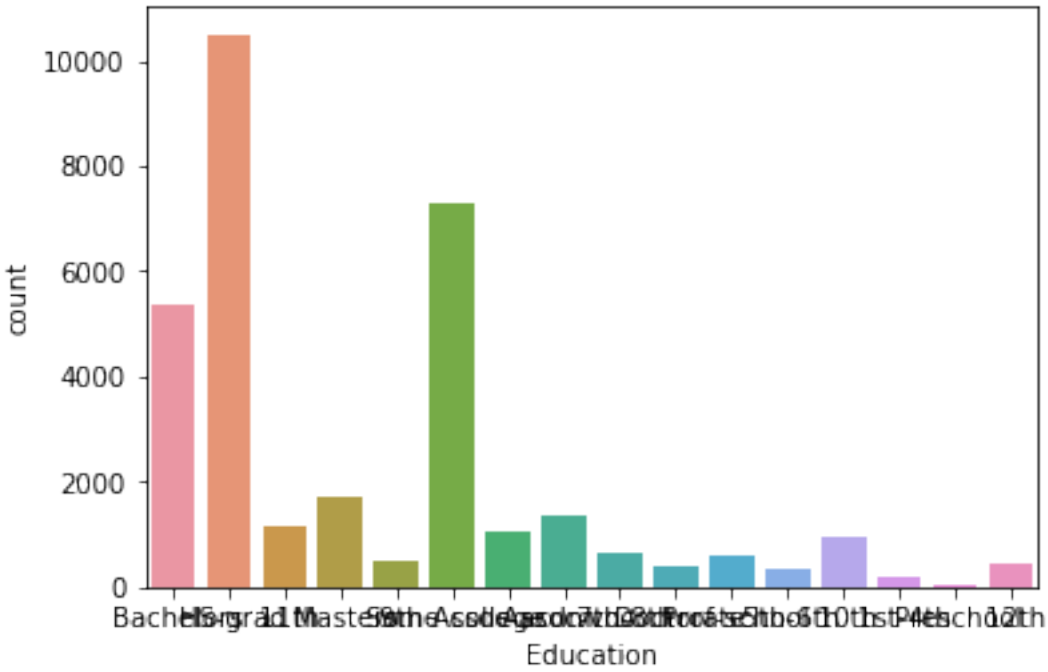



```
In [532]: sns.countplot(x='Workclass', data=df)

Out[532]: <matplotlib.axes._subplots.AxesSubplot at 0x1296fb4a8>
```



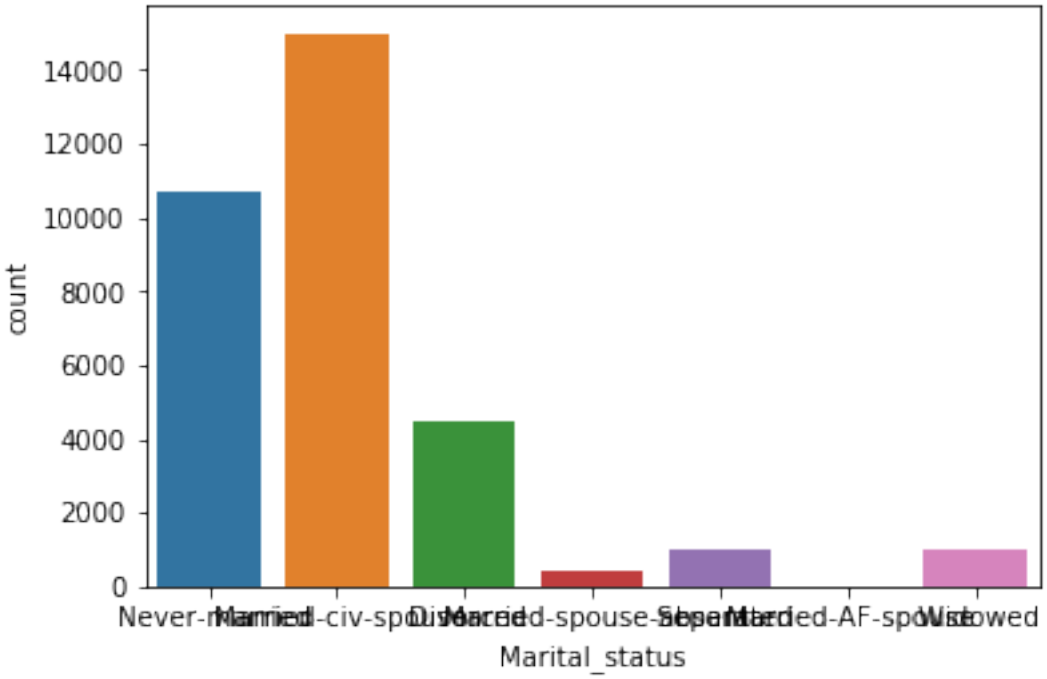
In [533]: sns.countplot(x='Education', data=df)

Out[533]: <matplotlib.axes._subplots.AxesSubplot at 0x12fcb5f60>



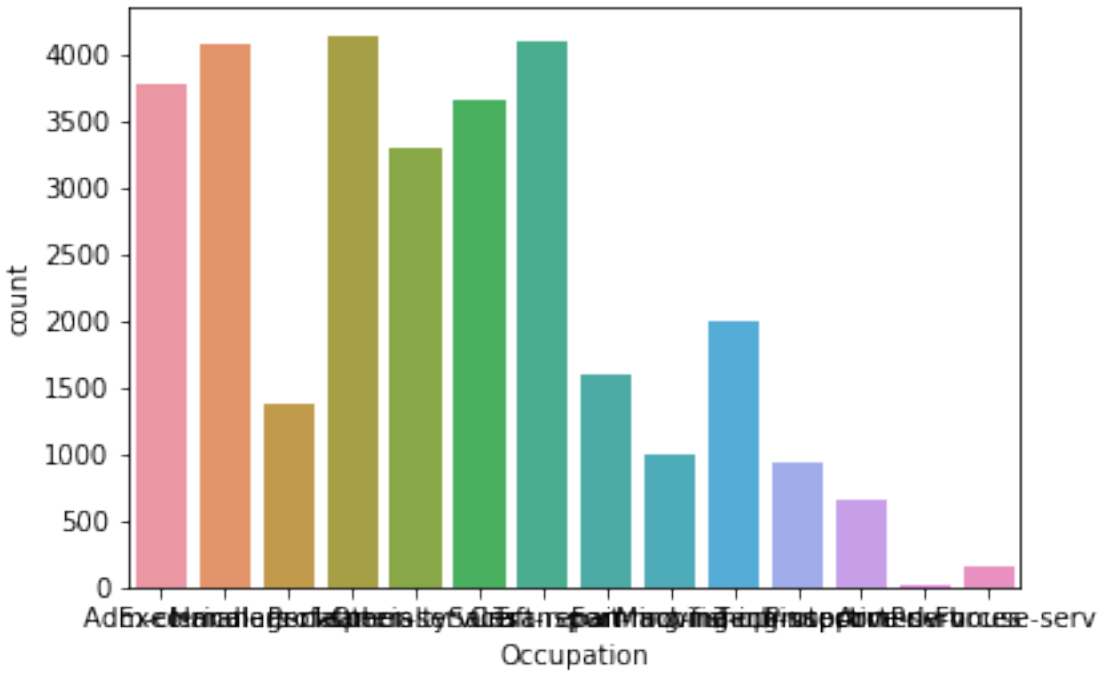
In [534]: sns.countplot(x='Marital_status', data=df)

Out[534]: <matplotlib.axes._subplots.AxesSubplot at 0x1298e35f8>



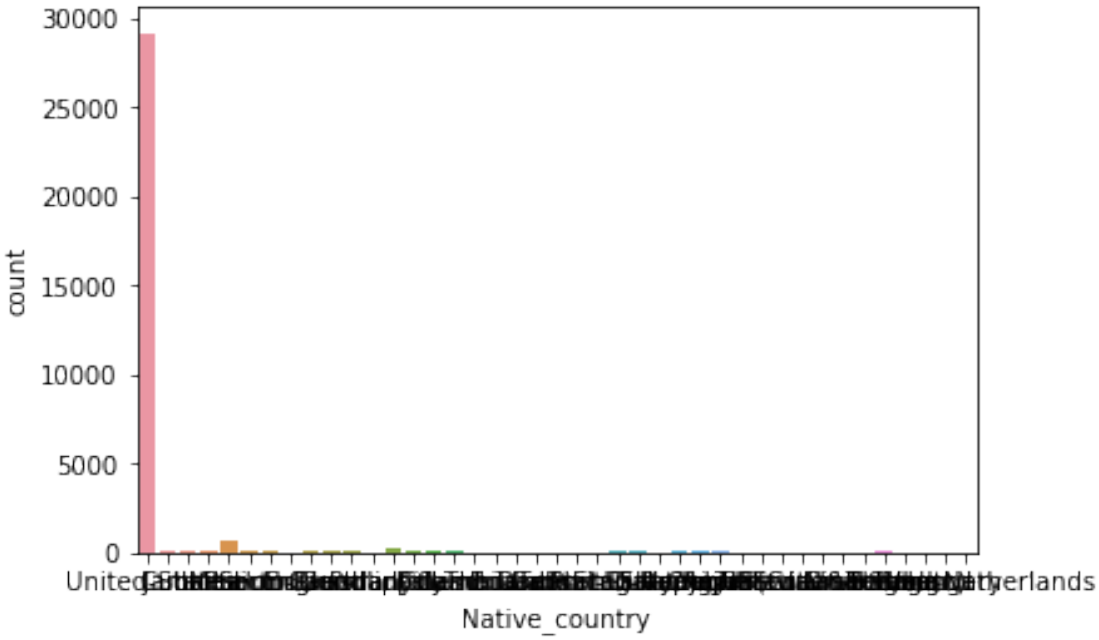
In [535]: sns.countplot(x='Occupation', data=df)

Out[535]: <matplotlib.axes._subplots.AxesSubplot at 0x129f1ef60>



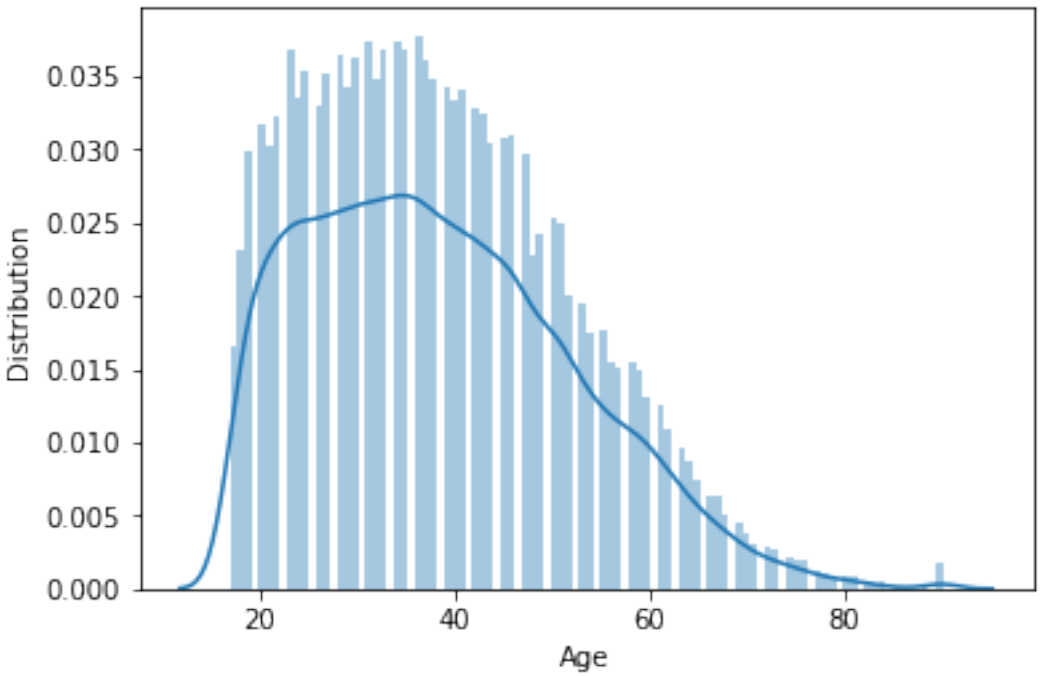
In [537]: sns.countplot(x='Native_country', data=df)

Out[537]: <matplotlib.axes._subplots.AxesSubplot at 0x12fd387b8>



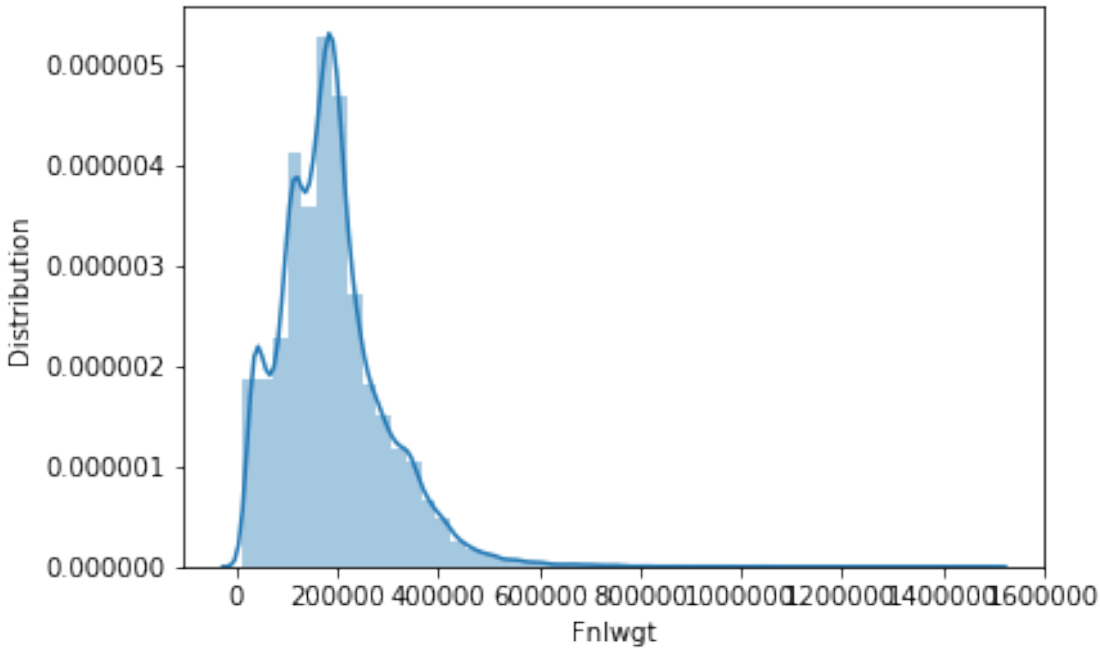
```
In [8]: sns.distplot(df['Age'], bins = 100)
        plt.ylabel("Distribution")
        plt.xlabel("Age")
        print ("The minimum age is", df['Age'].min())
        print ("The maximum age is", df['Age'].max())
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)


The minimum age is 17
The maximum age is 90
```



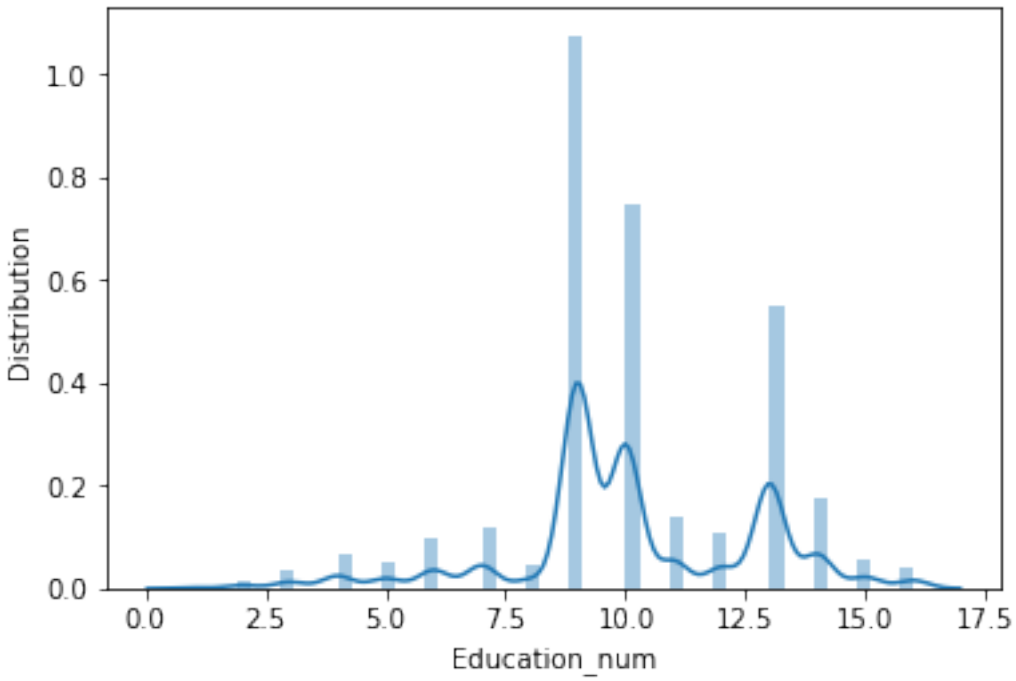

```
In [527]: sns.distplot(df['Fnlwgt'])
          plt.ylabel("Distribution")
          plt.xlabel("Fnlwgt")
          print ("The minimum age is", df['Fnlwgt'].min())
          print ("The maximum age is", df['Fnlwgt'].max())
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)


The minimum age is 12285
The maximum age is 1484705
```



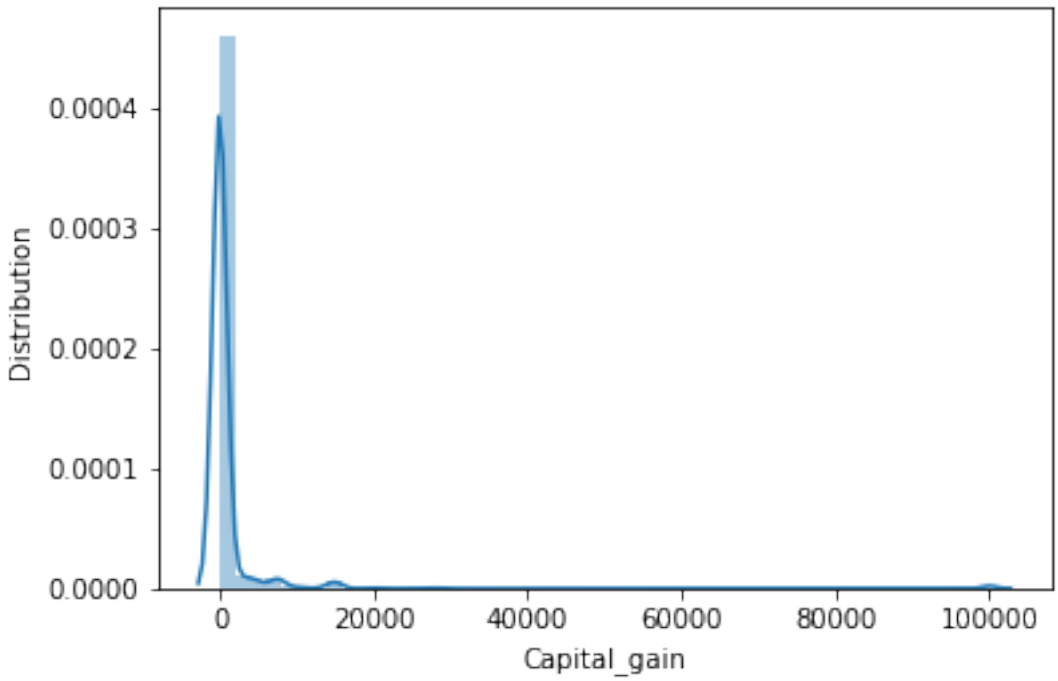

```
In [528]: sns.distplot(df['Education_num'])
          plt.ylabel("Distribution")
          plt.xlabel("Education_num")
          print ("The minimum age is", df['Education_num'].min())
          print ("The maximum age is", df['Education_num'].max())
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)


The minimum age is 1
The maximum age is 16
```

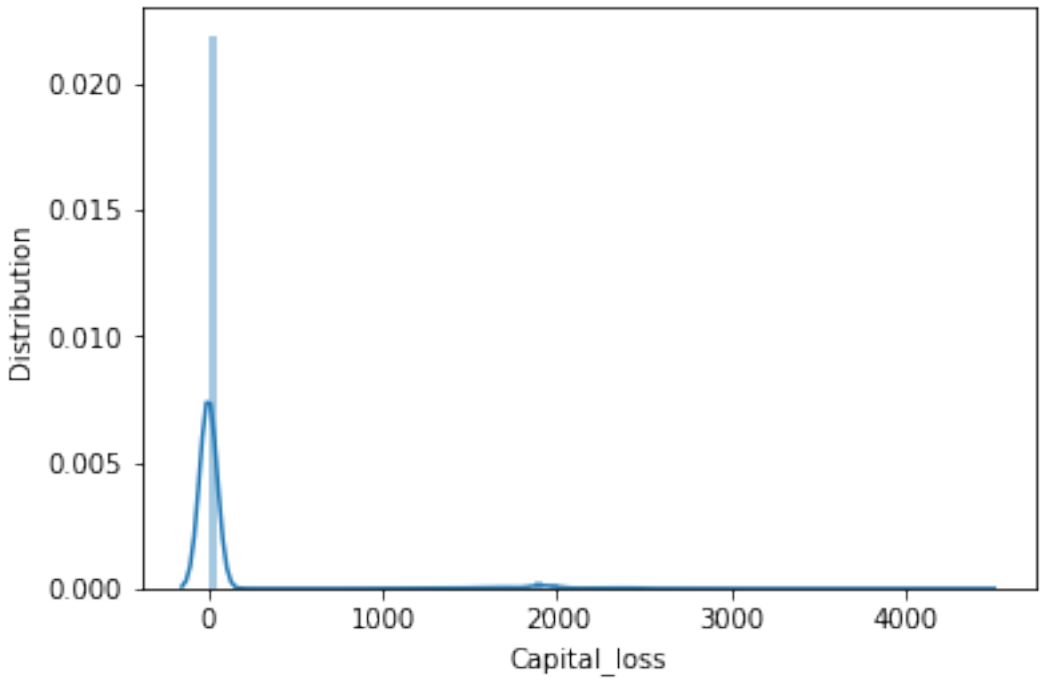



```
In [529]: sns.distplot(df['Capital_gain'])
          plt.ylabel("Distribution")
          plt.xlabel("Capital_gain")
          print ("The minimum age is", df['Capital_gain'].min())
          print ("The maximum age is", df['Capital_gain'].max())
```

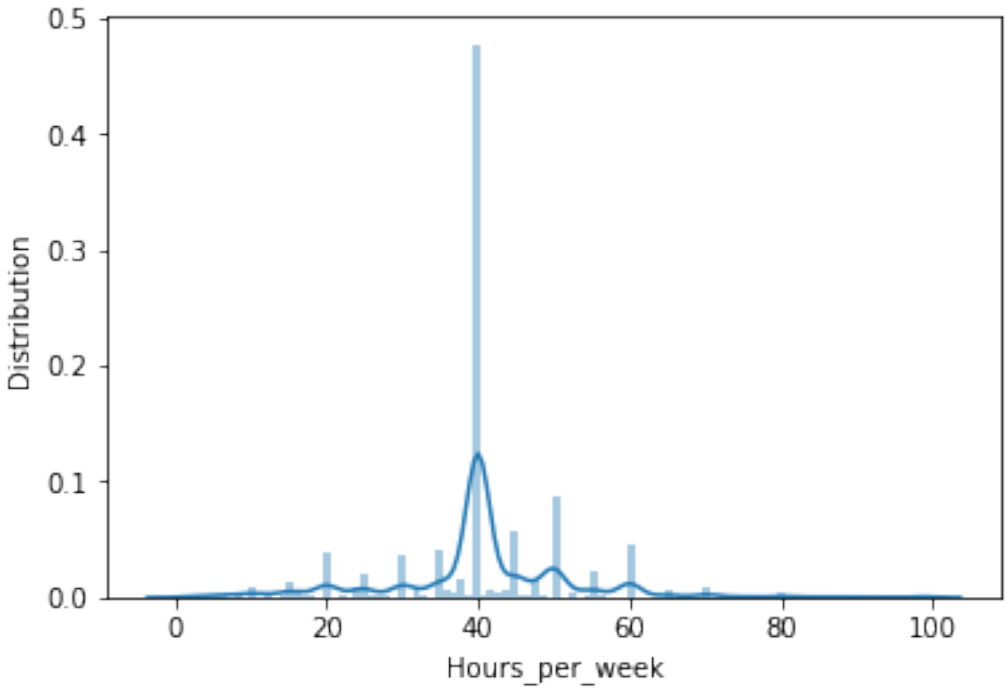
```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)


The minimum age is 0
The maximum age is 99999
```




```
In [530]: sns.distplot(df['Capital_loss'], bins = 100)
          plt.ylabel("Distribution")
          plt.xlabel("Capital_loss")
          print ("The minimum age is", df['Capital_loss'].min())
          print ("The maximum age is", df['Capital_loss'].max())

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)


The minimum age is 0
The maximum age is 4356
```




```
In [531]: sns.distplot(df['Hours_per_week'], bins = 100)
          plt.ylabel("Distribution")
          plt.xlabel("Hours_per_week")
          print ("The minimum age is", df['Hours_per_week'].min())
          print ("The maximum age is", df['Hours_per_week'].max())

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)


The minimum age is 1
The maximum age is 99
```



3 Apriori Method

```
In [10]: df1 = df[['Workclass', 'Education', 'Marital_status', 'Occupation', 'Relationship', '
                   'Income']]
         df1.head()
```

Out[10]:

	Workclass	Education	Marital_status	Occupation	\
0	State-gov	Bachelors	Never-married	Adm-clerical	
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	
2	Private	HS-grad	Divorced	Handlers-cleaners	
3	Private	11th	Married-civ-spouse	Handlers-cleaners	
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	

	Relationship	Race	Sex	Native_country	Income
0	Not-in-family	White	Male	United-States	<=50K
1	Husband	White	Male	United-States	<=50K
2	Not-in-family	White	Male	United-States	<=50K
3	Husband	Black	Male	United-States	<=50K
4	Wife	Black	Female	Cuba	<=50K

```
In [541]: def InitialC1L1(df, minsup):
              ListC = {}
              for i in range(len(df.columns)):
                  for j in range(len(df.index)):
```

```
                    if (df.iloc[j, i] in ListC):
                        ListC[df.iloc[j, i]] += 1
                    else:
                        ListC[df.iloc[j, i]] = 1

            C1 = pd.DataFrame.from_dict(ListC, orient = 'index', columns = ['supcount'])
            C1['support'] = C1['supcount']/len(df.index)
            C1 = C1.sort_values(by = ['support'])
            ListL = C1[C1['support'] >= minsup]
            ListL = ListL.sort_values(by = ['support'])
            return C1, ListL
```

In [542]:
```
def CreateC2L2(df, L, minsup):
    L = L.index.tolist()
    df_list = df.values.tolist()
    ListC = []
    C2 = {}
    L2 = {}
    for i in range(len(L)-1):
        for j in range(i+1, len(L)):
            ListC.append([L[i], L[j]])
    for itemset in df_list:
        for itemset1 in ListC:
            if set(itemset1).issubset(set(itemset)):
                if tuple(itemset1) in C2:
                    C2[tuple(itemset1)] += 1
                else:
                    C2[tuple(itemset1)] = 1
    C2 = pd.DataFrame.from_dict(C2, orient = 'index', columns = ['supcount'])
    C2['support'] = C2['supcount']/len(df.index)
    L2 = C2[C2['support'] >= minsup]
    L2 = L2.sort_values(by = ['support'])
    return C2, L2
```

In [543]:
```
def Pruning(Citem, Lsort):
    for h in Citem:
        testset = list(Citem)
        testset.remove(h)
        if sorted(testset) not in Lsort:
            return False
    return True
```

In [544]:
```
def CreateCkLk(df, L, k, minsup):
    L = L.index.tolist()
    Lsort = []
    for i in L:
        Lsort.append(sorted(i))
```

```
    df_list = df.values.tolist()
    ListC = []
    Ck = {}
    if k > 2:
        for i in range(len(Lsort)-1):
            for j in range(i+1, len(Lsort)):
                l1 = list(Lsort[i])
                l2 = list(Lsort[j])
                if l1[0:k-2] == l2[0:k-2]:
                    Citem = list(set(l1) | set(l2))
                    if Pruning(Citem, Lsort):
                        ListC.append(Citem)

    for itemset in df_list:
        for itemset1 in ListC:
            if set(itemset1).issubset(set(itemset)):
                if tuple(itemset1) in Ck:
                    Ck[tuple(itemset1)] += 1
                else:
                    Ck[tuple(itemset1)] = 1
    Ck = pd.DataFrame.from_dict(Ck, orient = 'index', columns = ['supcount'])
    Ck['support'] = Ck['supcount']/len(df.index)
    Lk = Ck[Ck['support'] >= minsup]
    Lk = Lk.sort_values(by = ['support'])
    return Ck, Lk
```

```
In [540]: C1, L1 = InitialC1L1(df = df1, minsup = 0.3)
          C1
```

Out[540]:

	supcount	support
Holand-Netherlands	1	0.000031
Never-worked	7	0.000215
Armed-Forces	9	0.000276
Scotland	12	0.000369
Honduras	13	0.000399
Hungary	13	0.000399
Outlying-US(Guam-USVI-etc)	14	0.000430
Without-pay	14	0.000430
Yugoslavia	16	0.000491
Laos	18	0.000553
Thailand	18	0.000553
Trinadad&Tobago	19	0.000584
Cambodia	19	0.000584
Hong	20	0.000614
Married-AF-spouse	23	0.000706
Ireland	24	0.000737
Ecuador	28	0.000860
France	29	0.000891

```
 Greece                          29  0.000891
 Peru                            31  0.000952
 Nicaragua                       34  0.001044
 Portugal                        37  0.001136
 Iran                            43  0.001321
 Haiti                           44  0.001351
 Taiwan                          51  0.001566
 Preschool                       51  0.001566
 Columbia                        59  0.001812
 Poland                          60  0.001843
 Japan                           62  0.001904
 Guatemala                       64  0.001966
...                             ...       ...
 Transport-moving              1597  0.049046
 Masters                       1723  0.052916
 Machine-op-inspct             2002  0.061485
 Local-gov                     2093  0.064279
 Self-emp-not-inc              2541  0.078038
 Black                         3124  0.095943
 Other-service                 3295  0.101195
 Unmarried                     3446  0.105832
 Sales                         3650  0.112097
 Adm-clerical                  3770  0.115783
 Exec-managerial               4066  0.124873
 Craft-repair                  4099  0.125887
 Prof-specialty                4140  0.127146
NaN                            4262  0.130893
 Divorced                      4443  0.136452
 Own-child                     5068  0.155646
 Bachelors                     5355  0.164461
 Some-college                  7291  0.223918
 >50K                          7841  0.240810
 Not-in-family                 8305  0.255060
 HS-grad                      10501  0.322502
 Never-married                10683  0.328092
 Female                       10771  0.330795
 Husband                      13193  0.405178
 Married-civ-spouse           14976  0.459937
 Male                         21790  0.669205
 Private                      22696  0.697030
 <=50K                        24720  0.759190
 White                        27816  0.854274
 United-States                29170  0.895857

[102 rows x 2 columns]
```

```
In [545]: L1
```

```
Out[545]:                     supcount   support
```

```
HS-grad                10501  0.322502
Never-married          10683  0.328092
Female                 10771  0.330795
Husband                13193  0.405178
Married-civ-spouse     14976  0.459937
Male                   21790  0.669205
Private                22696  0.697030
<=50K                  24720  0.759190
White                  27816  0.854274
United-States          29170  0.895857
```

In [546]:
```
C2, L2 = CreateC2L2(df = df1, L = L1, minsup = 0.3)
C2
```

Out[546]:

	supcount	support
(Never-married, Male)	5916	0.181690
(Never-married, <=50K)	10192	0.313012
(Never-married, White)	8757	0.268941
(Never-married, United-States)	9579	0.294186
(Male, <=50K)	15128	0.464605
(Male, White)	19174	0.588864
(Male, United-States)	19488	0.598507
(<=50K, White)	20699	0.635699
(<=50K, United-States)	21999	0.675624
(White, United-States)	25621	0.786862
(Husband, Married-civ-spouse)	13184	0.404902
(Husband, Male)	13192	0.405147
(Husband, <=50K)	7275	0.223427
(Husband, White)	11940	0.366696
(Husband, United-States)	11861	0.364270
(Married-civ-spouse, Male)	13319	0.409048
(Married-civ-spouse, <=50K)	8284	0.254415
(Married-civ-spouse, White)	13410	0.411842
(Married-civ-spouse, United-States)	13368	0.410553
(HS-grad, Male)	7111	0.218390
(HS-grad, Private)	7780	0.238936
(HS-grad, <=50K)	8826	0.271060
(HS-grad, White)	8904	0.273456
(HS-grad, United-States)	9702	0.297964
(Male, Private)	14944	0.458954
(Private, <=50K)	17733	0.544609
(Private, White)	19404	0.595928
(Private, United-States)	20135	0.618378
(Husband, Private)	8572	0.263260
(Married-civ-spouse, Private)	9732	0.298885
(Female, Married-civ-spouse)	1657	0.050889
(Female, Private)	7752	0.238076
(Female, <=50K)	9592	0.294586

```
( Female,  White)                          8642  0.265410
( Female,  United-States)                  9682  0.297350
( HS-grad,  Husband)                       4279  0.131415
( HS-grad,  Married-civ-spouse)            4845  0.148798
( Never-married,  Female)                  4767  0.146402
( Never-married,  Private)                 8186  0.251405
( HS-grad,  Never-married)                 3089  0.094868
( HS-grad,  Female)                        3390  0.104112
( Female,  Husband)                           1  0.000031
```

In [547]: L2

Out[547]:

	supcount	support
(Never-married, <=50K)	10192	0.313012
(Husband, United-States)	11861	0.364270
(Husband, White)	11940	0.366696
(Husband, Married-civ-spouse)	13184	0.404902
(Husband, Male)	13192	0.405147
(Married-civ-spouse, Male)	13319	0.409048
(Married-civ-spouse, United-States)	13368	0.410553
(Married-civ-spouse, White)	13410	0.411842
(Male, Private)	14944	0.458954
(Male, <=50K)	15128	0.464605
(Private, <=50K)	17733	0.544609
(Male, White)	19174	0.588864
(Private, White)	19404	0.595928
(Male, United-States)	19488	0.598507
(Private, United-States)	20135	0.618378
(<=50K, White)	20699	0.635699
(<=50K, United-States)	21999	0.675624
(White, United-States)	25621	0.786862

In [548]: C3, L3 = CreateCkLk(df = df1, L = L2, k = 3, minsup = 0.3)

In [549]: C3

Out[549]:

	supcount	support
(<=50K, White, Male)	13085	0.401861
(<=50K, United-States, Male)	13389	0.411197
(United-States, White, Male)	17653	0.542152
(<=50K, United-States, White)	18917	0.580971
(White, United-States, Husband)	11053	0.339455
(Married-civ-spouse, United-States, Husband)	11852	0.363994
(United-States, Male, Husband)	11860	0.364239
(Married-civ-spouse, White, Husband)	11931	0.366420
(White, Male, Husband)	11939	0.366666
(Married-civ-spouse, Male, Husband)	13183	0.404871
(Married-civ-spouse, White, Male)	12036	0.369645
(Married-civ-spouse, United-States, Male)	11947	0.366911

```
( Married-civ-spouse,  White,  United-States)       12369  0.379872
( Private,  White,  Male)                           13123  0.403028
( Private,  United-States,  Male)                   13209  0.405669
( <=50K,  Private,  Male)                           10707  0.328829
( <=50K,  White,  Private)                          14872  0.456743
( <=50K,  United-States,  Private)                  15594  0.478916
( Private,  United-States,  White)                  17728  0.544455
```

```
In [550]: L3
```

```
Out[550]:                                                  supcount   support
( <=50K,  Private,  Male)                                     10707  0.328829
( White,  United-States,  Husband)                            11053  0.339455
( Married-civ-spouse,  United-States,  Husband)               11852  0.363994
( United-States,  Male,  Husband)                             11860  0.364239
( Married-civ-spouse,  White,  Husband)                       11931  0.366420
( White,  Male,  Husband)                                     11939  0.366666
( Married-civ-spouse,  United-States,  Male)                  11947  0.366911
( Married-civ-spouse,  White,  Male)                          12036  0.369645
( Married-civ-spouse,  White,  United-States)                 12369  0.379872
( <=50K,  White,  Male)                                       13085  0.401861
( Private,  White,  Male)                                     13123  0.403028
( Married-civ-spouse,  Male,  Husband)                        13183  0.404871
( Private,  United-States,  Male)                             13209  0.405669
( <=50K,  United-States,  Male)                               13389  0.411197
( <=50K,  White,  Private)                                    14872  0.456743
( <=50K,  United-States,  Private)                            15594  0.478916
( United-States,  White,  Male)                               17653  0.542152
( Private,  United-States,  White)                            17728  0.544455
( <=50K,  United-States,  White)                              18917  0.580971
```

```
In [22]: C4, L4 = CreateCkLk(df = df1, L = L3, k = 4, minsup = 0.3)
```

```
In [551]: C4
```

```
Out[551]:                                                  supcount   support
( United-States,  <=50K,  White,  Male)                       11913  0.365867
( Married-civ-spouse,  United-States,  Husband,...            11044  0.339179
( United-States,  Husband,  White,  Male)                     11052  0.339424
( Married-civ-spouse,  United-States,  Husband,...            11851  0.363963
( Married-civ-spouse,  Husband,  White,  Male)                11930  0.366389
( Married-civ-spouse,  United-States,  White,  ...            11125  0.341666
( Private,  <=50K,  White,  Male)                              9230  0.283468
( Private,  United-States,  <=50K,  Male)                      9330  0.286539
( United-States,  Private,  White,  Male)                     11956  0.367188
( Private,  United-States,  <=50K,  White)                    13452  0.413132
```

```
In [552]: L4
```

```
Out[552]:                                                    supcount   support
          ( Married-civ-spouse,  United-States,  Husband,...     11044  0.339179
          ( United-States,  Husband,  White,  Male)              11052  0.339424
          ( Married-civ-spouse,  United-States,  White,  ...     11125  0.341666
          ( Married-civ-spouse,  United-States,  Husband,...     11851  0.363963
          ( United-States,  <=50K,  White,  Male)                11913  0.365867
          ( Married-civ-spouse,  Husband,  White,  Male)         11930  0.366389
          ( United-States,  Private,  White,  Male)              11956  0.367188
          ( Private,  United-States,  <=50K,  White)             13452  0.413132

In [38]: C5, L5 = CreateCkLk(df = df1, L = L4, k = 5, minsup = 0.3)

In [553]: L5

Out[553]:                                                    supcount   support
          ( Married-civ-spouse,  Husband,  White,  Male, ...     11043  0.339148

In [559]: time_start = time.perf_counter()

          C1, L1 = InitialC1L1(df = df1, minsup = 0.3)
          C2, L2 = CreateC2L2(df = df1, L = L1, minsup = 0.3)
          C3, L3 = CreateCkLk(df = df1, L = L2, k = 3, minsup = 0.3)
          C4, L4 = CreateCkLk(df = df1, L = L3, k = 4, minsup = 0.3)
          C5, L5 = CreateCkLk(df = df1, L = L4, k = 5, minsup = 0.3)
          FrequentItemSets = []
          FrequentItemSets.append(L1.index.tolist())
          FrequentItemSets.append(L2.index.tolist())
          FrequentItemSets.append(L3.index.tolist())
          FrequentItemSets.append(L4.index.tolist())
          FrequentItemSets.append(L5.index.tolist())
          print(FrequentItemSets)

          time_elapsed = (time.perf_counter() - time_start)
          memMb=resource.getrusage(resource.RUSAGE_SELF).ru_maxrss/1024.0/1024.0
          print ("%5.1f secs %5.1f MByte" % (time_elapsed,memMb))

[[' HS-grad', ' Never-married', ' Female', ' Husband', ' Married-civ-spouse', ' Male', ' Privat
 10.0 secs 189.6 MByte
```

4 Improvement of the Apriori Method

```
In [566]: def ImprovedCreateCkLk(df, L, k, minsup):
              L = L.index.tolist()
              Lsort = []
              for i in L:
                  Lsort.append(sorted(i))
              df_list = df.values.tolist()
```

```
    ListC = []
    Ck = {}
    if k > 2:
        for i in range(len(Lsort)-1):
            for j in range(i+1, len(Lsort)):
                l1 = list(Lsort[i])
                l2 = list(Lsort[j])
                if l1[0:k-2] == l2[0:k-2]:
                    Citem = list(set(l1) | set(l2))
                    if ImprovedPruning(Citem, Lsort, l1, l2):
                        ListC.append(Citem)

    for itemset in df_list:
        for itemset1 in ListC:
            if set(itemset1).issubset(set(itemset)):
                if tuple(itemset1) in Ck:
                    Ck[tuple(itemset1)] += 1
                else:
                    Ck[tuple(itemset1)] = 1
    Ck = pd.DataFrame.from_dict(Ck, orient = 'index', columns = ['supcount'])
    Ck['support'] = Ck['supcount']/len(df.index)
    Lk = Ck[Ck['support'] >= minsup]
    Lk = Lk.sort_values(by = ['support'])
    return Ck, Lk
```

In [569]:
```
def ImprovedPruning(Citem, Lsort, l1, l2):
    for h in Citem:
        testset = list(Citem)
        testset.remove(h)
        if testset != l1 and testset != l2:
            if sorted(testset) not in Lsort:
                return False
    return True
```

In [574]:
```
time_start = time.perf_counter()

C1, L1 = InitialC1L1(df = df1, minsup = 0.3)
C2, L2 = CreateC2L2(df = df1, L = L1, minsup = 0.3)
C3, L3 = ImprovedCreateCkLk(df = df1, L = L2, k = 3, minsup = 0.3)
C4, L4 = ImprovedCreateCkLk(df = df1, L = L3, k = 4, minsup = 0.3)
C5, L5 = ImprovedCreateCkLk(df = df1, L = L4, k = 5, minsup = 0.3)
FrequentItemSets = []
FrequentItemSets.append(L1.index.tolist())
FrequentItemSets.append(L2.index.tolist())
FrequentItemSets.append(L3.index.tolist())
FrequentItemSets.append(L4.index.tolist())
FrequentItemSets.append(L5.index.tolist())
print(FrequentItemSets)
```

```
        time_elapsed = (time.perf_counter() - time_start)
        memMb=resource.getrusage(resource.RUSAGE_SELF).ru_maxrss/1024.0/1024.0
        print ("%5.1f secs %5.1f MByte" % (time_elapsed,memMb))
```

```
[[' HS-grad', ' Never-married', ' Female', ' Husband', ' Married-civ-spouse', ' Male', ' Privat
 10.2 secs 189.6 MByte
```

5 FP-Growth Method

```
In [27]: def FList(df, minsup):
             PlaceHold = {}
             for i in range(len(df.columns)):
                 for j in range(len(df.index)):
                     if (df.iloc[j, i] in PlaceHold):
                         PlaceHold[df.iloc[j, i]] += 1
                     else:
                         PlaceHold[df.iloc[j, i]] = 1

             FList = pd.DataFrame.from_dict(PlaceHold, orient = 'index', columns = ['supcount']
             FList['support'] = FList['supcount']/len(df.index)
             FList = FList[FList['support'] >= minsup]
             FList = FList.sort_values(by = ['support'], ascending = False)
             return FList
```

```
In [578]: def Cleandf(df, FL):
              FL = Flist.index.tolist()
              df_list = df1.values.tolist()
              for i in df_list:
                  for j in i:
                      if j not in FL:
                          i.remove(j)
              for i in df_list:
                  for j in i:
                      if j not in FL:
                          i.remove(j)
              for i in df_list:
                  for j in i:
                      if j not in FL:
                          i.remove(j)
              for itemset in df_list:
                  itemset = itemset.sort(key = lambda i: FL.index(i))
              return df_list
```

```
In [580]: def FPTREE(df_list, FL):
              FL = FL.sort_values(by = ['support'], ascending = True)
              FL = FL.index.tolist()
```

```
                #df_list = df.values.tolist()
                Tree = {}
                for i in FL:
                    Tree[i] = {}
                for i in FL:
                    for itemset in df_list:
                        if i in itemset:
                            index = itemset.index(i)
                            if tuple(itemset[0:index]) in Tree[i]:
                                Tree[i][tuple(itemset[0:index])] += 1
                            else:
                                Tree[i][tuple(itemset[0:index])] = 1
                return Tree

In [681]: def  ConditionalFPtrees(df, FL, FPtree, minsup):
                FL = FL.sort_values(by = ['support'], ascending = True)
                FL = FL.index.tolist()
                CPB = []
                frequentitemsets = {}
                frequentitemsets3 = []
                for i in FPtree:
                    for j in FPtree[i]:
                        CPB.append([i, list(j), FPtree[i][j]])
                CPB = pd.DataFrame(CPB, columns = ['condition','subsets', 'supcount'])
                CPB['support'] = CPB['supcount']/len(df.index)
                for condition in FL:
                    #len2 frequent itemsets
                    placehold = CPB[CPB['condition'] == condition]
                    placehold = placehold.reset_index()
                    Count = {}
                    for i in range(len(placehold.index)):
                        for j in placehold['subsets'][i]:
                            if j in Count:
                                Count[j] += placehold['support'][i]
                            else:
                                Count[j] = placehold['support'][i]
                    placehold1 = pd.DataFrame.from_dict(Count, orient = 'index', columns = ['sup
                    placehold1 = placehold1[placehold1['support'] >= minsup]
                    placehold2 = placehold1.index.to_list()
                    for n in placehold2:
                        nsup = placehold1['support'][n]
                        nlist = [n, condition]
                        frequentitemsets[tuple(nlist)] = nsup
                    #len3 frequent itemsets
                    if len(placehold2) >= 2:
                        for i in range(len(placehold2)-1):
                            for j in range(i+1, len(placehold2)):
                                l1 = placehold2[i]
```

```
                                l2 = placehold2[j]
                                itemsethold = [l1, l2]
                                placehold3 = placehold.values.tolist()
                                for m in range(len(placehold.index)):
                                    if set(itemsethold).issubset(set(placehold3[m][2])):
                                        itemsethold1 = itemsethold.copy()
                                        itemsethold1.append(condition)
                                        if tuple(itemsethold1) in frequentitemsets:
                                            frequentitemsets[tuple(itemsethold1)] += placehold3[
                                        else:
                                            frequentitemsets[tuple(itemsethold1)] = placehold3[m]
                                            frequentitemsets3.append([condition, itemsethold])
                    frequentitemsets = dict((k, v) for k, v in frequentitemsets.items() if v >= minsu
                    return frequentitemsets, frequentitemsets3

In [671]: def lenkfrequentitemsets(itemsets, lenk, FL, FPtree, minsup):
                    FL = FL.sort_values(by = ['support'], ascending = True)
                    FL = FL.index.tolist()
                    itemdic = {}
                    itemlist = []

                    CPB = []
                    for i in FPtree:
                        for j in FPtree[i]:
                            CPB.append([i, list(j), FPtree[i][j]])
                    CPB = pd.DataFrame(CPB, columns = ['condition','subsets', 'supcount'])
                    CPB['support'] = CPB['supcount']/len(df.index)
                    for condition in FL:
                        placehold = CPB[CPB['condition'] == condition]
                        placehold = placehold.values.tolist()
                        placehold1 = []
                        for i in range(len(itemsets)):
                            if itemsets[i][0] == condition:
                                placehold1.append(itemsets[i][1])
                        if len(placehold1) > 0:
                            itemhold = []
                            for n in range(len(placehold1)-1):
                                for m in range(n+1, len(placehold1)):
                                    l1 = placehold1[n]
                                    l2 = placehold1[m]
                                    if l1[0:lenk-2] == l2[0:lenk-2]:
                                        itemsethold = list(set(l1) | set(l2))
                                        itemhold.append(itemsethold)
                            for g in itemhold:
                                for h in range(len(placehold)):
                                    if set(g).issubset(set(placehold[h][1])):
                                        itemsethold1 = list(g).copy()
                                        itemsethold1.append(condition)
```

```
                                if tuple(itemsethold1) in itemdic:
                                    itemdic[tuple(itemsethold1)] += placehold[h][3]
                                else:
                                    itemdic[tuple(itemsethold1)] = placehold[h][3]
                                    itemlist.append([condition, g])
            itemdic = dict((k, v) for k, v in itemdic.items() if v >= minsup)

            return itemdic, itemlist
```

In [687]:
```
def Merge(dict1, dict2):
    return(dict2.update(dict1))
```

In [688]:
```
Flist = FList(df1, 0.3)
Flist
```

Out[688]:

	supcount	support
United-States	29170	0.895857
White	27816	0.854274
<=50K	24720	0.759190
Private	22696	0.697030
Male	21790	0.669205
Married-civ-spouse	14976	0.459937
Husband	13193	0.405178
Female	10771	0.330795
Never-married	10683	0.328092
HS-grad	10501	0.322502

In [689]:
```
df_list = Cleandf(df1, Flist)
```

In [581]:
```
FPtree = FPTREE(df_list, Flist)
FPtree
```

Out[581]:
```
{' HS-grad': {(' United-States', ' White', ' <=50K', ' Private', ' Male'): 516,
  (' United-States',
   ' White',
   ' Male',
   ' Married-civ-spouse',
   ' Husband'): 378,
  (' United-States', ' White', ' <=50K', ' Male', ' Never-married'): 225,
  (' United-States',
   ' White',
   ' <=50K',
   ' Private',
   ' Male',
   ' Never-married'): 1158,
  (' United-States', ' <=50K', ' Private', ' Female'): 185,
  (' United-States', ' White', ' <=50K', ' Private', ' Female'): 898,
  (' United-States',
   ' White',
```

```
 ' <=50K',
 ' Private',
 ' Male',
 ' Married-civ-spouse',
 ' Husband'): 1683,
(' White', ' <=50K', ' Private', ' Female', ' Never-married'): 51,
(' United-States', ' White', ' <=50K', ' Male'): 182,
(' United-States',
 ' White',
 ' Private',
 ' Married-civ-spouse',
 ' Female'): 98,
(' White', ' <=50K', ' Private', ' Male', ' Never-married'): 93,
(' United-States',
 ' White',
 ' <=50K',
 ' Private',
 ' Married-civ-spouse',
 ' Female'): 184,
(' United-States', ' White', ' Private', ' Female'): 26,
(' <=50K', ' Private', ' Married-civ-spouse', ' Female'): 16,
(' United-States', ' <=50K', ' Male', ' Never-married'): 62,
(' United-States', ' White', ' <=50K', ' Female', ' Never-married'): 110,
(' White', ' Male', ' Married-civ-spouse', ' Husband'): 18,
(' United-States',
 ' White',
 ' <=50K',
 ' Private',
 ' Female',
 ' Never-married'): 747,
(' United-States',
 ' White',
 ' <=50K',
 ' Male',
 ' Married-civ-spouse',
 ' Husband'): 804,
(' United-States', ' <=50K', ' Private', ' Female', ' Never-married'): 199,
(' United-States', ' White', ' Male'): 25,
(' <=50K', ' Private', ' Female', ' Never-married'): 34,
(' United-States', ' <=50K', ' Private', ' Male', ' Never-married'): 209,
(' United-States',
 ' <=50K',
 ' Private',
 ' Male',
 ' Married-civ-spouse',
 ' Husband'): 168,
(' White', ' <=50K', ' Male', ' Never-married'): 14,
(' United-States', ' White', ' <=50K', ' Female'): 297,
```

```
(' United-States', ' White', ' Married-civ-spouse', ' Female'): 51,
(' United-States', ' <=50K', ' Male', ' Married-civ-spouse', ' Husband'): 67,
(' United-States',
 ' White',
 ' Private',
 ' Male',
 ' Married-civ-spouse',
 ' Husband'): 811,
(' United-States', ' White', ' <=50K', ' Married-civ-spouse', ' Female'): 75,
(' White',
 ' <=50K',
 ' Private',
 ' Male',
 ' Married-civ-spouse',
 ' Husband'): 103,
(' Male', ' Married-civ-spouse', ' Husband'): 3,
(' <=50K', ' Private', ' Male', ' Never-married'): 48,
(' United-States', ' <=50K', ' Female'): 73,
(' United-States', ' <=50K', ' Female', ' Never-married'): 80,
(' United-States',
 ' <=50K',
 ' Private',
 ' Married-civ-spouse',
 ' Female'): 41,
(' White', ' <=50K', ' Private', ' Female'): 60,
(' United-States', ' <=50K', ' Married-civ-spouse', ' Female'): 13,
(' United-States', ' <=50K', ' Male'): 35,
(' United-States',
 ' Private',
 ' Male',
 ' Married-civ-spouse',
 ' Husband'): 48,
(' White', ' <=50K', ' Male'): 8,
(' White', ' <=50K', ' Married-civ-spouse', ' Female'): 9,
(' United-States', ' White', ' Private', ' Male'): 35,
(' United-States', ' Male', ' Married-civ-spouse', ' Husband'): 34,
(' White', ' <=50K', ' Private', ' Married-civ-spouse', ' Female'): 20,
(' United-States', ' <=50K', ' Private', ' Male'): 71,
(' White', ' <=50K', ' Female'): 14,
(' <=50K', ' Female'): 12,
(' United-States', ' White', ' Private', ' Male', ' Never-married'): 18,
(' Private', ' Female'): 1,
(' Private', ' Male', ' Married-civ-spouse', ' Husband'): 11,
(' <=50K', ' Male', ' Married-civ-spouse', ' Husband'): 19,
(' United-States', ' Private', ' Married-civ-spouse', ' Female'): 4,
(' White', ' Female', ' Never-married'): 1,
(' White', ' Married-civ-spouse', ' Female'): 4,
(' Married-civ-spouse', ' Female'): 3,
```

```
(' White', ' Private', ' Male', ' Married-civ-spouse', ' Husband'): 30,
(' United-States', ' White', ' Male', ' Never-married'): 14,
(' White', ' <=50K', ' Male', ' Married-civ-spouse', ' Husband'): 40,
(' <=50K', ' Private', ' Female'): 36,
(' United-States', ' White', ' <=50K', ' Male', ' Married-civ-spouse'): 5,
(' White', ' <=50K', ' Private', ' Male'): 37,
(' <=50K', ' Private', ' Male'): 15,
(' United-States', ' White', ' <=50K', ' Private', ' Male', ' Husband'): 3,
(' Private', ' Male'): 1,
(' <=50K', ' Private', ' Male', ' Married-civ-spouse', ' Husband'): 56,
(' <=50K', ' Married-civ-spouse', ' Female'): 1,
(' United-States',
 ' White',
 ' <=50K',
 ' Private',
 ' Male',
 ' Married-civ-spouse'): 19,
(' <=50K', ' Female', ' Never-married'): 5,
(' United-States', ' White', ' Female'): 12,
(' White', ' Private', ' Married-civ-spouse', ' Female'): 7,
(' United-States', ' White', ' Private', ' Male', ' Married-civ-spouse'): 2,
(' United-States', ' Married-civ-spouse', ' Female'): 4,
(' Male',): 2,
(' United-States', ' Private', ' Female'): 4,
(' White', ' <=50K', ' Private', ' Male', ' Married-civ-spouse'): 1,
(' United-States',
 ' White',
 ' <=50K',
 ' Private',
 ' Married-civ-spouse',
 ' Husband',
 ' Female'): 1,
(' United-States', ' Male', ' Never-married'): 4,
(' United-States', ' <=50K', ' Private', ' Male', ' Married-civ-spouse'): 7,
(' United-States', ' White', ' <=50K', ' Male', ' Husband'): 1,
(' United-States', ' Private', ' Male'): 7,
(' United-States', ' White', ' Private', ' Female', ' Never-married'): 4,
(' United-States', ' White', ' Private', ' Male', ' Husband'): 1,
(' <=50K', ' Male', ' Never-married'): 5,
(' White', ' Private', ' Male', ' Married-civ-spouse'): 1,
(' Private', ' Male', ' Never-married'): 2,
(' White', ' <=50K', ' Female', ' Never-married'): 3,
(' Private', ' Married-civ-spouse', ' Female'): 2,
(' Female',): 1,
(' White', ' Private', ' Female'): 3,
(' <=50K', ' Male'): 4,
(' White', ' Private', ' Male', ' Never-married'): 1,
(' White', ' Private', ' Male'): 1,
```

```
 (' United-States', ' White', ' Female', ' Never-married'): 1,
 (' United-States', ' <=50K', ' Male', ' Married-civ-spouse'): 1,
 (' United-States', ' Private', ' Male', ' Never-married'): 1,
 (' White', ' <=50K', ' Male', ' Married-civ-spouse'): 1,
 (' <=50K', ' Private', ' Male', ' Married-civ-spouse'): 2,
 (' United-States', ' Male'): 1},
' Never-married': {(' United-States', ' White', ' <=50K', ' Male'): 993,
 (' United-States', ' White', ' Private', ' Female'): 95,
 (' United-States', ' White', ' <=50K', ' Private', ' Female'): 2642,
 (' United-States', ' <=50K', ' Private', ' Male'): 515,
 (' United-States', ' White', ' <=50K', ' Private', ' Male'): 3278,
 (' White', ' <=50K', ' Private', ' Male'): 361,
 (' White', ' <=50K', ' Private', ' Female'): 213,
 (' United-States', ' White', ' Female'): 44,
 (' United-States', ' <=50K', ' Male'): 172,
 (' United-States', ' White', ' <=50K', ' Female'): 747,
 (' <=50K', ' Private', ' Male'): 176,
 (' United-States', ' <=50K', ' Private', ' Female'): 537,
 (' <=50K', ' Private', ' Female'): 136,
 (' White', ' <=50K', ' Male'): 49,
 (' United-States', ' Private', ' Male'): 14,
 (' United-States', ' White', ' Private', ' Male'): 179,
 (' United-States', ' <=50K', ' Female'): 247,
 (' <=50K', ' Female'): 34,
 (' White', ' Private', ' Male'): 14,
 (' United-States', ' Private', ' Female'): 10,
 (' United-States', ' White', ' Male'): 86,
 (' <=50K', ' Male'): 47,
 (' White', ' Female'): 4,
 (' White', ' <=50K', ' Female'): 45,
 (' White', ' Male'): 4,
 (' United-States', ' Female'): 6,
 (' United-States', ' Male'): 14,
 (' Private', ' Male'): 10,
 (' Male',): 4,
 (' Private', ' Female'): 3,
 (' White', ' Private', ' Female'): 3,
 (' Female',): 1},
' Female': {(' <=50K', ' Private', ' Married-civ-spouse'): 57,
 (' United-States',
  ' White',
  ' <=50K',
  ' Private',
  ' Married-civ-spouse'): 414,
 (' <=50K', ' Private'): 234,
 (' United-States', ' White', ' Private'): 232,
 (' United-States', ' White', ' <=50K', ' Private'): 4756,
 (' United-States', ' White'): 117,
```

```
 (' United-States', ' <=50K', ' Private'): 1011,
 (' United-States', ' <=50K', ' Private', ' Married-civ-spouse'): 82,
 (' White', ' <=50K', ' Private'): 412,
 (' White', ' Private', ' Married-civ-spouse'): 29,
 (' United-States', ' White', ' Private', ' Married-civ-spouse'): 369,
 (' United-States', ' White', ' <=50K'): 1619,
 (' White', ' <=50K'): 114,
 (' United-States', ' <=50K'): 479,
 (' United-States', ' White', ' Married-civ-spouse'): 246,
 (' United-States', ' White', ' <=50K', ' Married-civ-spouse'): 214,
 (' <=50K', ' Married-civ-spouse'): 17,
 (' United-States',): 20,
 (' White', ' <=50K', ' Private', ' Married-civ-spouse'): 59,
 (' United-States', ' <=50K', ' Married-civ-spouse'): 34,
 (' United-States', ' Private', ' Married-civ-spouse'): 34,
 (' <=50K',): 64,
 (' United-States', ' Private'): 27,
 (' White', ' <=50K', ' Married-civ-spouse'): 25,
 (): 4,
 (' Private',): 7,
 (' White',): 9,
 (' White', ' Married-civ-spouse'): 17,
 (' Married-civ-spouse',): 13,
 (' Private', ' Married-civ-spouse'): 19,
 (' United-States', ' Married-civ-spouse'): 27,
 (' United-States',
  ' White',
  ' <=50K',
  ' Private',
  ' Married-civ-spouse',
  ' Husband'): 1,
 (' White', ' Private'): 9},
' Husband': {(' United-States',
  ' White',
  ' <=50K',
  ' Male',
  ' Married-civ-spouse'): 2094,
 (' United-States',
  ' <=50K',
  ' Private',
  ' Male',
  ' Married-civ-spouse'): 351,
 (' United-States', ' White', ' Male', ' Married-civ-spouse'): 1850,
 (' United-States',
  ' White',
  ' Private',
  ' Male',
  ' Married-civ-spouse'): 3268,
```

```
 (' United-States', ' Private', ' Male', ' Married-civ-spouse'): 170,
 (' Male', ' Married-civ-spouse'): 63,
 (' Private', ' Male', ' Married-civ-spouse'): 124,
 (' <=50K', ' Private', ' Male', ' Married-civ-spouse'): 187,
 (' United-States',
  ' White',
  ' <=50K',
  ' Private',
  ' Male',
  ' Married-civ-spouse'): 3831,
 (' United-States', ' <=50K', ' Male', ' Married-civ-spouse'): 154,
 (' White', ' Private', ' Male', ' Married-civ-spouse'): 189,
 (' United-States', ' Male', ' Married-civ-spouse'): 133,
 (' White', ' <=50K', ' Private', ' Male', ' Married-civ-spouse'): 445,
 (' White', ' Male', ' Married-civ-spouse'): 117,
 (' White', ' <=50K', ' Male', ' Married-civ-spouse'): 136,
 (' <=50K', ' Male', ' Married-civ-spouse'): 71,
 (' United-States', ' White', ' <=50K', ' Private', ' Male'): 4,
 (' United-States', ' White', ' <=50K', ' Private', ' Married-civ-spouse'): 1,
 (' United-States', ' White', ' <=50K', ' Male'): 1,
 (' United-States', ' White', ' Private', ' Male'): 2,
 (' United-States', ' White', ' Male'): 2},
' Married-civ-spouse': {(' United-States', ' White', ' <=50K', ' Male'): 2112,
 (' United-States', ' <=50K', ' Private', ' Male'): 361,
 (' <=50K', ' Private'): 57,
 (' United-States', ' White', ' <=50K', ' Private'): 415,
 (' United-States', ' White', ' Male'): 1855,
 (' United-States', ' White', ' Private', ' Male'): 3281,
 (' United-States', ' Private', ' Male'): 172,
 (' Male',): 64,
 (' Private', ' Male'): 125,
 (' <=50K', ' Private', ' Male'): 198,
 (' United-States', ' White', ' <=50K', ' Private', ' Male'): 3877,
 (' United-States', ' <=50K', ' Male'): 156,
 (' White', ' Private', ' Male'): 191,
 (' United-States', ' Male'): 133,
 (' United-States', ' <=50K', ' Private'): 82,
 (' White', ' Private'): 29,
 (' White', ' <=50K', ' Private', ' Male'): 463,
 (' United-States', ' White', ' Private'): 369,
 (' White', ' Male'): 117,
 (' United-States', ' White'): 246,
 (' United-States', ' White', ' <=50K'): 214,
 (' <=50K',): 17,
 (' White', ' <=50K', ' Private'): 59,
 (' White', ' <=50K', ' Male'): 140,
 (' United-States', ' <=50K'): 34,
 (' United-States', ' Private'): 34,
```

```
 (' White', ' <=50K'): 25,
 (' <=50K', ' Male'): 74,
 (' White',): 17,
 (): 13,
 (' Private',): 19,
 (' United-States',): 27},
' Male': {(' United-States', ' White', ' <=50K'): 3632,
 (' United-States', ' White', ' <=50K', ' Private'): 8281,
 (' United-States', ' <=50K', ' Private'): 1049,
 (' United-States', ' White'): 2065,
 (' United-States', ' White', ' Private'): 3675,
 (' United-States', ' Private'): 204,
 (): 74,
 (' Private',): 140,
 (' <=50K', ' Private'): 428,
 (' United-States', ' <=50K'): 427,
 (' White', ' <=50K', ' Private'): 949,
 (' White', ' Private'): 218,
 (' United-States',): 155,
 (' White', ' <=50K'): 223,
 (' White',): 131,
 (' <=50K',): 139},
' Private': {(' United-States', ' White', ' <=50K'): 13452,
 (' United-States', ' <=50K'): 2142,
 (' <=50K',): 719,
 (' United-States', ' White'): 4276,
 (' United-States',): 265,
 (): 166,
 (' White', ' <=50K'): 1420,
 (' White',): 256},
' <=50K': {(' United-States', ' White'): 18917,
 (' United-States',): 3082,
 (): 939,
 (' White',): 1782},
' White': {(' United-States',): 25621, (): 2195},
' United-States': {(): 29170}}
```

In [682]:
```
frequentitemsetsFP23, itemsets3 = ConditionalFPtrees(df1, Flist, FPtree, minsup = 0.
```

In [683]:
```
frequentitemsetsFP23
```

Out[683]:
```
{(' <=50K', ' Never-married'): 0.3130124996161053,
 (' United-States', ' Husband'): 0.3642701391234914,
 (' White', ' Husband'): 0.366696354534566,
 (' Male', ' Husband'): 0.4051472620619761,
 (' Married-civ-spouse', ' Husband'): 0.4049015693621204,
 (' United-States', ' White', ' Husband'): 0.33945517643807016,
 (' United-States', ' Male', ' Husband'): 0.3642394275360094,
```

```
 (' United-States', ' Married-civ-spouse', ' Husband'): 0.3639937348361537,
 (' White', ' Male', ' Husband'): 0.36666564294708404,
 (' White', ' Married-civ-spouse', ' Husband'): 0.36641995024722834,
 (' Male', ' Married-civ-spouse', ' Husband'): 0.40487085777463844,
 (' United-States', ' Married-civ-spouse'): 0.4105525014588004,
 (' White', ' Married-civ-spouse'): 0.41184238813304264,
 (' Male', ' Married-civ-spouse'): 0.40904763367218455,
 (' United-States', ' White', ' Married-civ-spouse'): 0.3798716255643254,
 (' United-States', ' Male', ' Married-civ-spouse'): 0.36691133564693956,
 (' White', ' Male', ' Married-civ-spouse'): 0.36964466693283377,
 (' United-States', ' Male'): 0.5985074168483769,
 (' White', ' Male'): 0.5888639783790424,
 (' <=50K', ' Male'): 0.46460489542704464,
 (' Private', ' Male'): 0.45895396333036453,
 (' United-States', ' White', ' Male'): 0.5421516538189859,
 (' United-States', ' <=50K', ' Male'): 0.4111974447959216,
 (' United-States', ' Private', ' Male'): 0.4056693590491693,
 (' White', ' <=50K', ' Male'): 0.4018611222014066,
 (' White', ' Private', ' Male'): 0.40302816252572093,
 (' <=50K', ' Private', ' Male'): 0.328828967169313,
 (' United-States', ' Private'): 0.6183778139492031,
 (' White', ' Private'): 0.5959276434998925,
 (' <=50K', ' Private'): 0.5446085808175425,
 (' United-States', ' White', ' Private'): 0.5444550228801327,
 (' United-States', ' <=50K', ' Private'): 0.4789164951936366,
 (' White', ' <=50K', ' Private'): 0.4567427290316637,
 (' United-States', ' <=50K'): 0.6756242130155707,
 (' White', ' <=50K'): 0.6356991492890267,
 (' United-States', ' White', ' <=50K'): 0.5809711003961795,
 (' United-States', ' White'): 0.7868615828752188}
```

In [684]:
```
itemsets3
```

Out[684]:
```
[[' Husband', [' United-States', ' White']],
 [' Husband', [' United-States', ' Male']],
 [' Husband', [' United-States', ' Married-civ-spouse']],
 [' Husband', [' White', ' Male']],
 [' Husband', [' White', ' Married-civ-spouse']],
 [' Husband', [' Male', ' Married-civ-spouse']],
 [' Married-civ-spouse', [' United-States', ' White']],
 [' Married-civ-spouse', [' United-States', ' Male']],
 [' Married-civ-spouse', [' White', ' Male']],
 [' Male', [' United-States', ' White']],
 [' Male', [' United-States', ' <=50K']],
 [' Male', [' United-States', ' Private']],
 [' Male', [' White', ' <=50K']],
 [' Male', [' White', ' Private']],
 [' Male', [' <=50K', ' Private']],
```

```
           [' Private', [' United-States', ' White']],
           [' Private', [' United-States', ' <=50K']],
           [' Private', [' White', ' <=50K']],
           [' <=50K', [' United-States', ' White']]]
```

```
In [672]: frequentitemsetsFP4, itemsets4 = lenkfrequentitemsets(itemsets = itemsets3, lenk = 3
                                                                FPtree = FPtree, minsup = 0.3)
```

```
In [673]: frequentitemsetsFP4
```

```
Out[673]: {(' White', ' United-States', ' Male', ' Husband'): 0.3394244648505882,
           (' Married-civ-spouse',
            ' White',
            ' United-States',
            ' Husband'): 0.3391787721507325,
           (' Married-civ-spouse',
            ' United-States',
            ' Male',
            ' Husband'): 0.36396302324867175,
           (' Married-civ-spouse', ' White', ' Male', ' Husband'): 0.3663892386597464,
           (' White',
            ' United-States',
            ' Male',
            ' Married-civ-spouse'): 0.34166641073677095,
           (' <=50K', ' White', ' United-States', ' Male'): 0.3658671416725531,
           (' Private', ' White', ' United-States', ' Male'): 0.3671877399342772,
           (' <=50K', ' White', ' United-States', ' Private'): 0.4131322748072848}
```

```
In [674]: itemsets4
```

```
Out[674]: [[' Husband', [' White', ' United-States', ' Male']],
           [' Husband', [' Married-civ-spouse', ' White', ' United-States']],
           [' Husband', [' Married-civ-spouse', ' United-States', ' Male']],
           [' Husband', [' Married-civ-spouse', ' White', ' Male']],
           [' Married-civ-spouse', [' White', ' United-States', ' Male']],
           [' Male', [' <=50K', ' White', ' United-States']],
           [' Male', [' Private', ' White', ' United-States']],
           [' Male', [' <=50K', ' Private', ' United-States']],
           [' Male', [' <=50K', ' Private', ' White']],
           [' Private', [' <=50K', ' White', ' United-States']]]
```

```
In [676]: frequentitemsetsFP5, itemsets5 = lenkfrequentitemsets(itemsets = itemsets4, lenk = 4
                                                                FPtree = FPtree, minsup = 0.3)
```

```
In [677]: frequentitemsetsFP5
```

```
Out[677]: {(' Married-civ-spouse',
            ' United-States',
            ' White',
            ' Male',
            ' Husband'): 0.33914806056325053}
```

```
In [678]: itemsets5

Out[678]: [[' Husband', [' Married-civ-spouse', ' United-States', ' White', ' Male']],
           [' Male', [' United-States', ' Private', ' <=50K', ' White']]]

In [679]: frequentitemsetsFP6, itemsets6 = lenkfrequentitemsets(itemsets = itemsets5, lenk = 5
                                                                FPtree = FPtree, minsup = 0.3)

In [680]: frequentitemsetsFP6

Out[680]: {}

In [693]: time_start = time.perf_counter()

          Flist = FList(df1, 0.3)
          df_list = Cleandf(df1, Flist)
          FPtree = FPTREE(df_list, Flist)
          frequentitemsetsFP23, itemsets3 = ConditionalFPtrees(df1, Flist, FPtree, minsup = 0.3
          frequentitemsetsFP4, itemsets4 = lenkfrequentitemsets(itemsets = itemsets3, lenk = 3
                                                                FPtree = FPtree, minsup = 0.3)
          frequentitemsetsFP5, itemsets5 = lenkfrequentitemsets(itemsets = itemsets4, lenk = 4
                                                                FPtree = FPtree, minsup = 0.3)

          print(frequentitemsetsFP23)
          print(frequentitemsetsFP4)
          print(frequentitemsetsFP5)

          time_elapsed = (time.perf_counter() - time_start)
          memMb=resource.getrusage(resource.RUSAGE_SELF).ru_maxrss/1024.0/1024.0
          print ("%5.1f secs %5.1f MByte" % (time_elapsed,memMb))

{(' <=50K', ' Never-married'): 0.3130124996161053, (' United-States', ' Husband'): 0.364270139
{(' White', ' United-States', ' Male', ' Husband'): 0.3394244648505882, (' Married-civ-spouse'
{(' Married-civ-spouse', ' United-States', ' White', ' Male', ' Husband'): 0.33914806056325053
  7.7 secs 189.6 MByte


In [ ]:
```