In [82]:

```
from sklearn.ensemble import GradientBoostingClassifier
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [69]:

```
df = pd.read_csv("C:/Users/HP/Downloads/income_evaluation.csv")
df.columns
```

Out[69]:

```
Index(['age', ' workclass', ' fnlwgt', ' education', ' education-num',
       ' marital-status', ' occupation', ' relationship', ' race', ' se
x',
       ' capital-gain', ' capital-loss', ' hours-per-week', ' native-coun
try',
       ' income'],
      dtype='object')
```

In [70]:

```
df.head()
```

Out[70]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black

In [71]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              32561 non-null  int64 
 1    workclass       32561 non-null  object
 2    fnlwgt          32561 non-null  int64 
 3    education       32561 non-null  object
 4    education-num   32561 non-null  int64 
 5    marital-status  32561 non-null  object
 6    occupation      32561 non-null  object
 7    relationship    32561 non-null  object
 8    race            32561 non-null  object
 9    sex             32561 non-null  object
 10   capital-gain    32561 non-null  int64 
 11   capital-loss    32561 non-null  int64 
 12   hours-per-week  32561 non-null  int64 
 13   native-country  32561 non-null  object
 14   income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
# Data Preprocessing
```

In [74]:

```
df.head()
```

Out[74]:

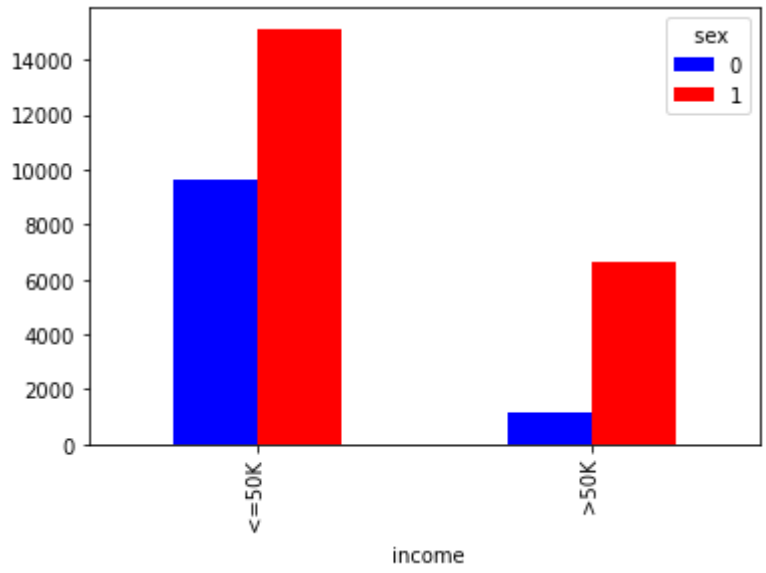
	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black

In [75]:

```
# plot of sex vs income
pd.crosstab(df[' income'],df[' sex']).plot(kind='bar',color=['blue','red'])
```

Out[75]:

```
<AxesSubplot:xlabel=' income'>
```



In [76]:

```
np.unique(df[' marital-status'])
```

Out[76]:

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

In [77]:

```
np.unique(df[' workclass'])
```

Out[77]:

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

In [78]:

```
df.drop(columns=' fnlwgt',inplace=True)
```

In [79]:

```
column_names = ['age', 'workclass', 'education', 'education_num', 'marital_status',
             'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'nati

df.columns = column_names
df.columns
```

Out[79]:

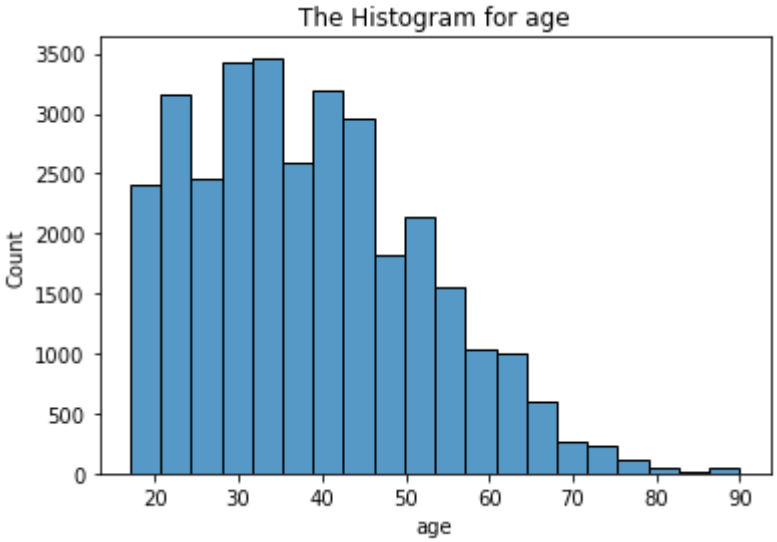
```
Index(['age', 'workclass', 'education', 'education_num', 'marital_statu
s',
       'occupation', 'relationship', 'race', 'sex', 'capital_gain',
       'capital_loss', 'hours_per_week', 'native_country', 'income'],
      dtype='object')
```

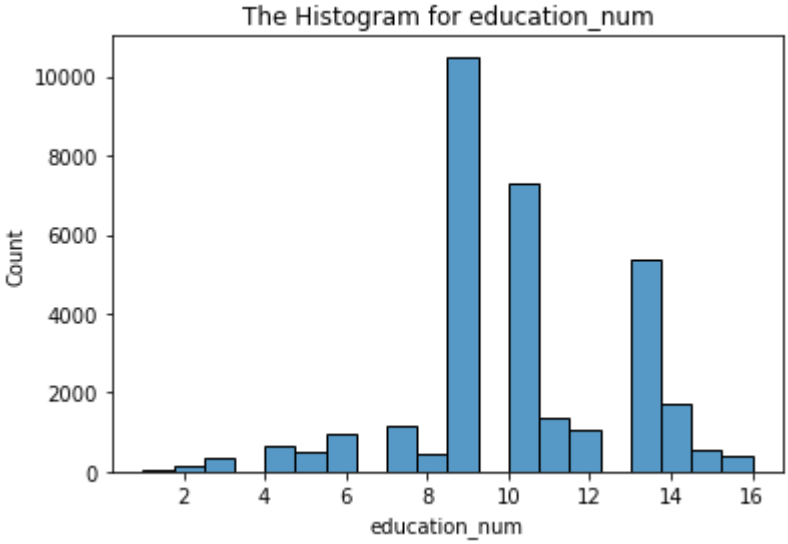
In [80]:

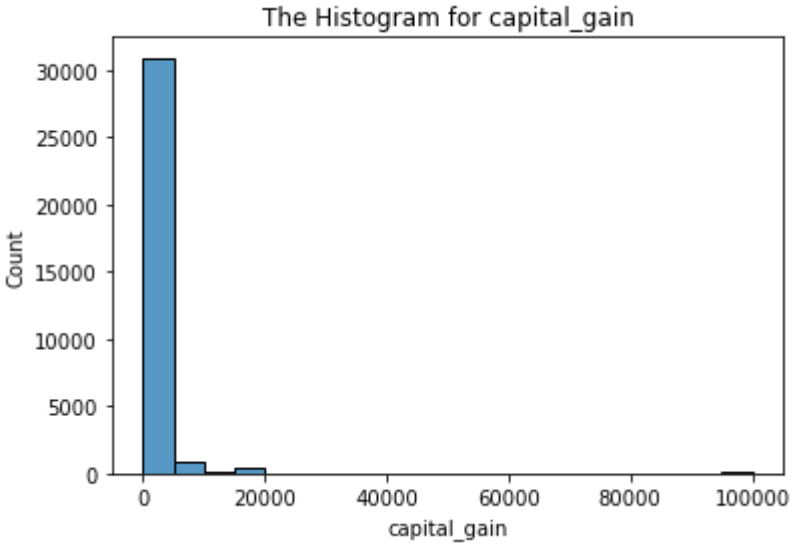
```
categorical = [var for var in df.columns if df[var].dtype == 'o' ]
numerical = [num for num in df.columns if df[num].dtype == 'int64']
```

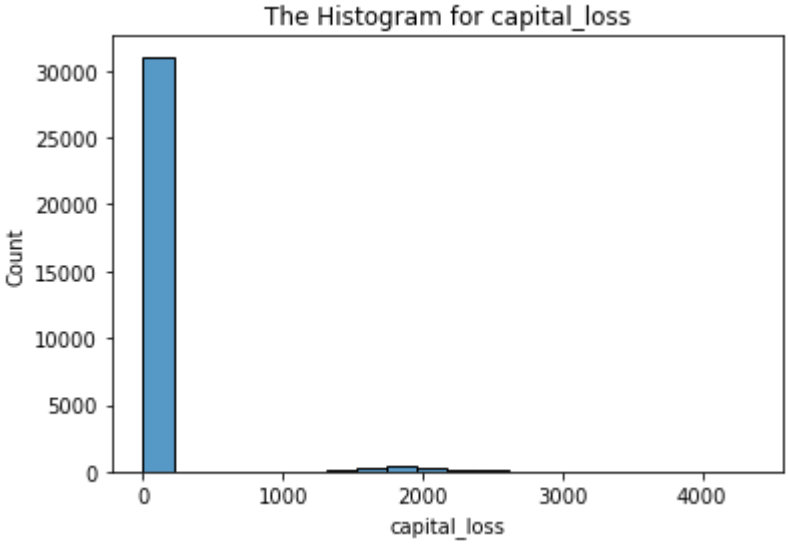
In [83]:

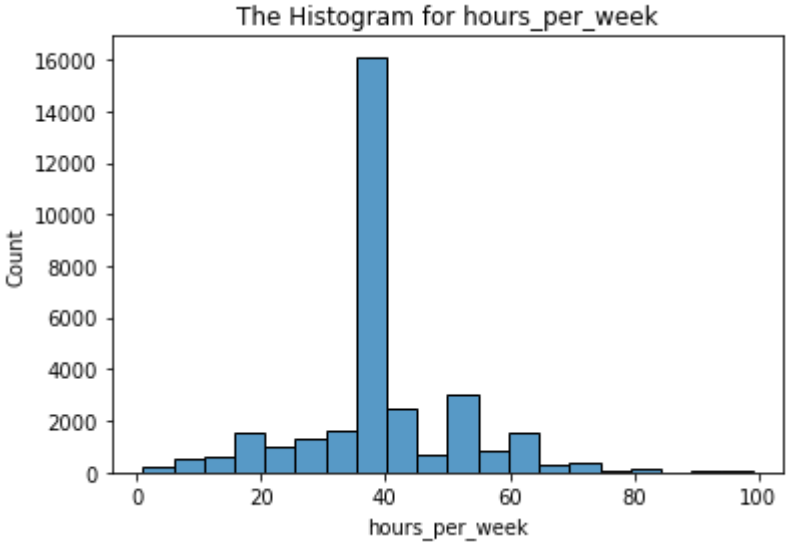
```
for i in numerical:
    sns.histplot(x=df[i], palette='Set1',bins=20)
    plt.title("The Histogram for {}".format(i))
    plt.show()
```





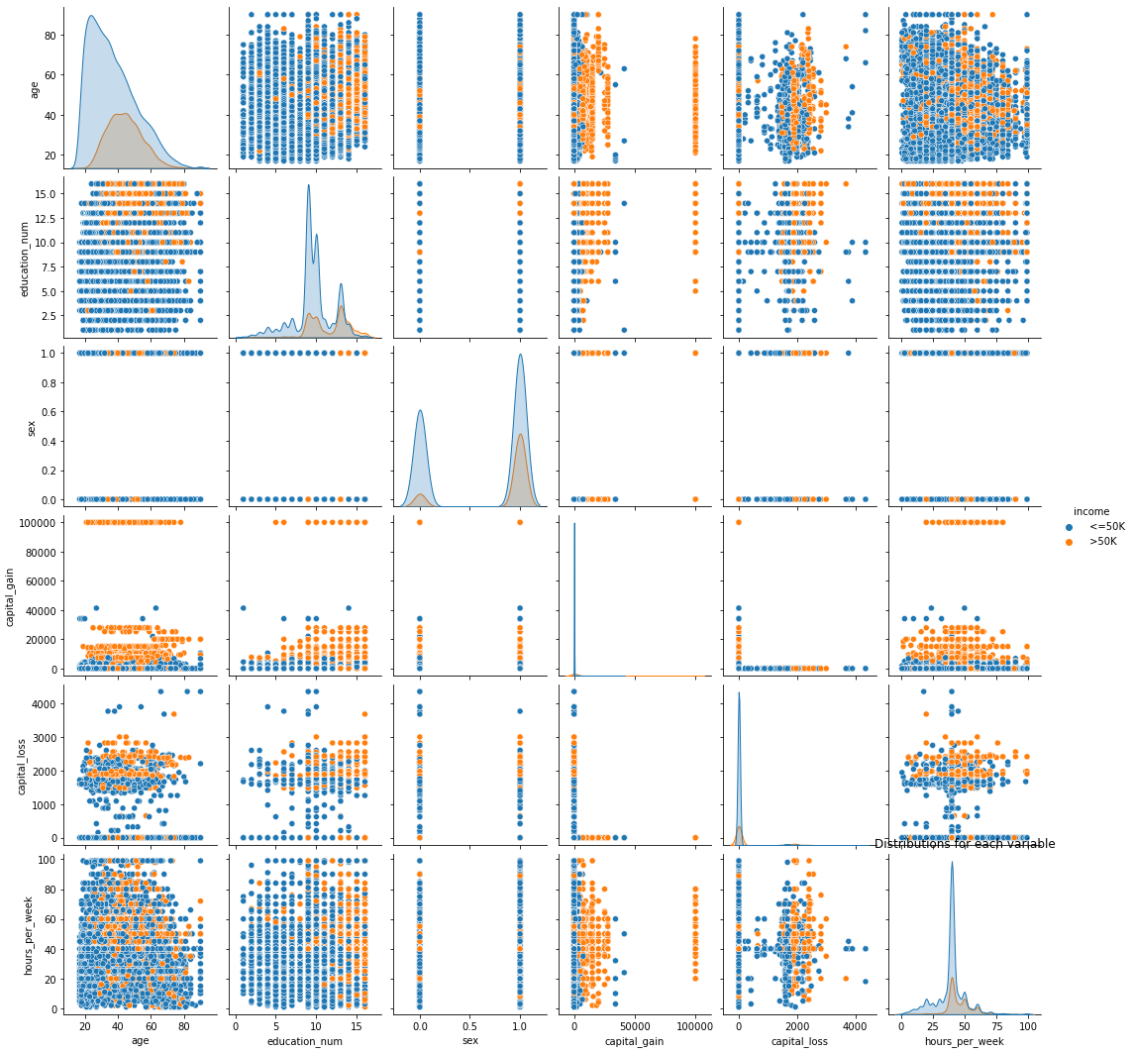



The Histogram for capital_loss
Count
capital_loss
0
5000
10000
15000
20000
25000
30000
0
1000
2000
3000
4000


The Histogram for hours_per_week
Count
hours_per_week
0
2000
4000
6000
8000
10000
12000
14000
16000
0
20
40
60
80
100

In [93]:

```
sns.pairplot(data=df, hue="income")
plt.title('Distributions for each variable')
plt.show()
```



In [101]:

```
def label_encoder(a):
    le = LabelEncoder()
    df[a] = le.fit_transform(df[a])

label_list = ['workclass', 'education','marital_status',
       'occupation', 'relationship', 'race', 'sex','native_country', 'income']

for i in label_list:
    label_encoder(i)
```

In [104]:

```
x = df.drop('income',axis=1)
y=df['income']
```

In [105]:

```
X_train, X_test, y_train, y_test = train_test_split( x,y, test_size=0.2)
```

In [110]:

```
# Define Gradient Boosting Classifier with hyperparameters
GBA=GradientBoostingClassifier(n_estimators=500,learning_rate=0.05,random_state=100
GBA.fit(X_train,y_train)
print("Test Score:",GBA.score(X_test,y_test))
```

```
Test Score: 0.8694917856594503
```

In [111]:

```
print(confusion_matrix(y_test, GBA.predict(X_test)))
```

```
[[4644  249]
 [ 601 1019]]
```

Hyperparameter tunning

In [118]:

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

grid = {
    'learning_rate':[0.01,0.05,0.1],
    'n_estimators':np.arange(100,500,100),
}

gb = GradientBoostingClassifier()
gb_cv = GridSearchCV(gb, grid, cv = 4)
gb_cv.fit(X_train,y_train)
print("Best Parameters:",gb_cv.best_params_)
print("Train Score:",gb_cv.best_score_)
print("Test Score:",gb_cv.score(X_test,y_test))
```

```
Best Parameters: {'learning_rate': 0.1, 'n_estimators': 300}
Train Score: 0.8707002457002456
Test Score: 0.8727161062490404
```