In [52]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [53]:
```
data_train = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data', header = None)
data_test = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test', skiprows = 1, head
```

In [54]:
```
data_train.tail(3)
```

Out[54]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

In [55]:
```
data_test.tail(3)
```

Out[55]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
16278	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	United-States	<=50K.
16279	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455	0	40	United-States	<=50K.
16280	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	United-States	>50K.

Adding colum to dataset

In [56]:
```python
column = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status',
                'occupation','relationship', 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week',
                 'native_country', 'wage_class']
data_train.columns = column
data_test.columns = column
```

In [57]:
```python
data_train.head(3)
```

Out[57]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	

In [58]:
```python
df = pd.concat([data_train,data_test])
df['workclass'].value_counts()
```

Out[58]:
```
 Private             33906
 Self-emp-not-inc     3862
 Local-gov            3136
 ?                    2799
 State-gov            1981
 Self-emp-inc         1695
 Federal-gov          1432
 Without-pay            21
 Never-worked           10
Name: workclass, dtype: int64
```

In [59]:
```
for column in df.columns:
    print(f" value count for {column} : \n {df[column].value_counts()}")
```

```
208174     1
Name: fnlwgt, Length: 28523, dtype: int64
 value count for education :
  HS-grad         15784
 Some-college    10878
 Bachelors        8025
 Masters          2657
 Assoc-voc        2061
 11th             1812
 Assoc-acdm       1601
 10th             1389
 7th-8th           955
 Prof-school       834
 9th               756
 12th              657
 Doctorate         594
 5th-6th           509
 1st-4th           247
 Preschool          83
Name: education, dtype: int64
```

df.info()

In [60]:
```
df.describe()
```

Out[60]:

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

replacing ? from workclass column

In [61]:
```
df.replace('?',np.nan,inplace=True)
```

In [62]:
```
df.wage_class.unique()
```

Out[62]:
```
array([' <=50K', ' >50K', ' <=50K.', ' >50K.'], dtype=object)
```

In [66]:
```
df.replace({' <=50K':0,' >50K':1,' <=50K.':0,' >50K.':1}).head(3)
```

Out[66]:

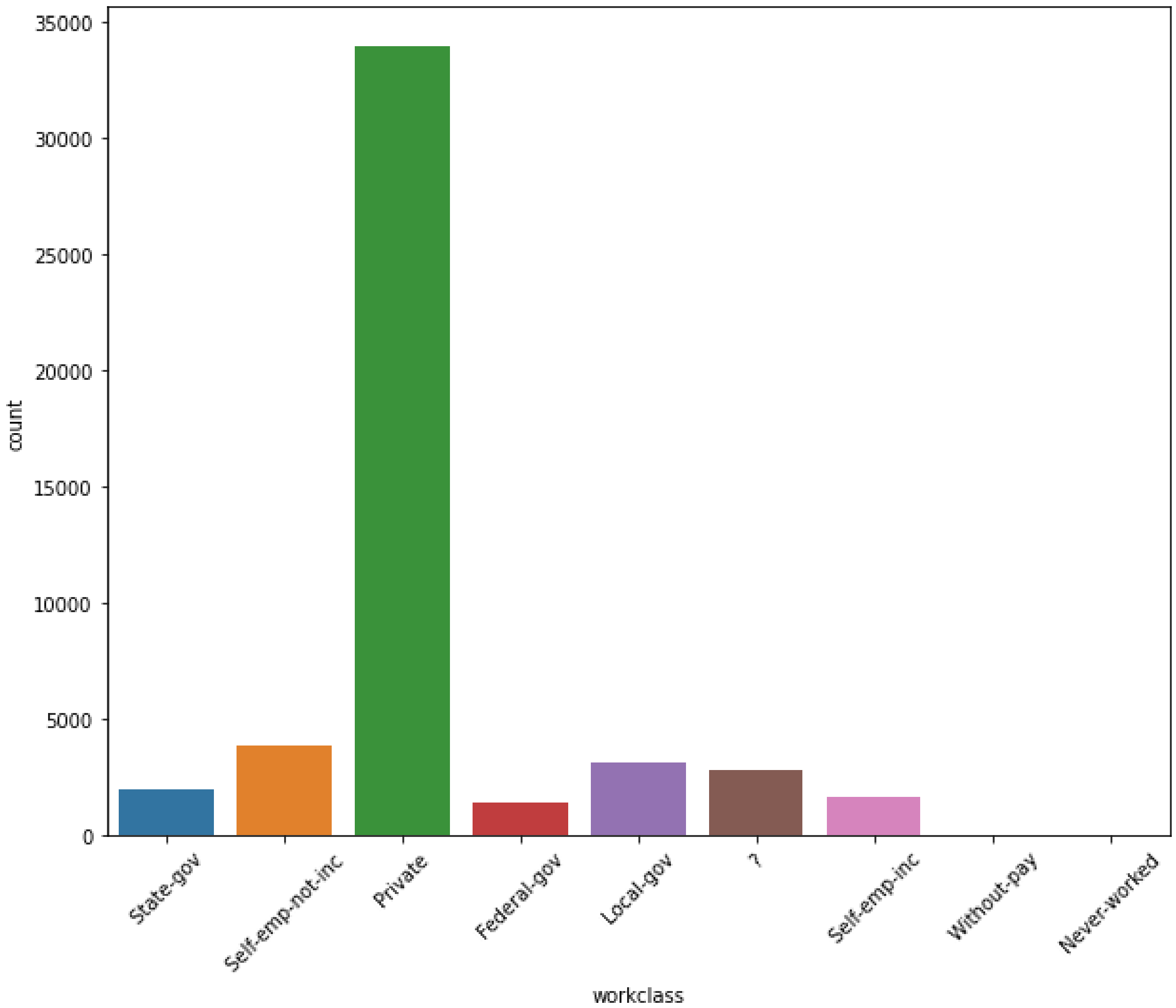
	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	

In [67]:
```
df['workclass'].fillna('0',inplace=True)
```

In [69]:

```
plt.figure(figsize=(10,8))
sns.countplot(df['workclass'])
plt.xticks(rotation = 45)
plt.show()
```

```
C:\Users\kants\AppData\Local\Programs\Python\Python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```


35000
30000
25000
20000
15000
10000
5000
0
count
State-gov
Self-emp-not-inc
Private
Federal-gov
Local-gov
?
Self-emp-inc
Without-pay
Never-worked
workclass

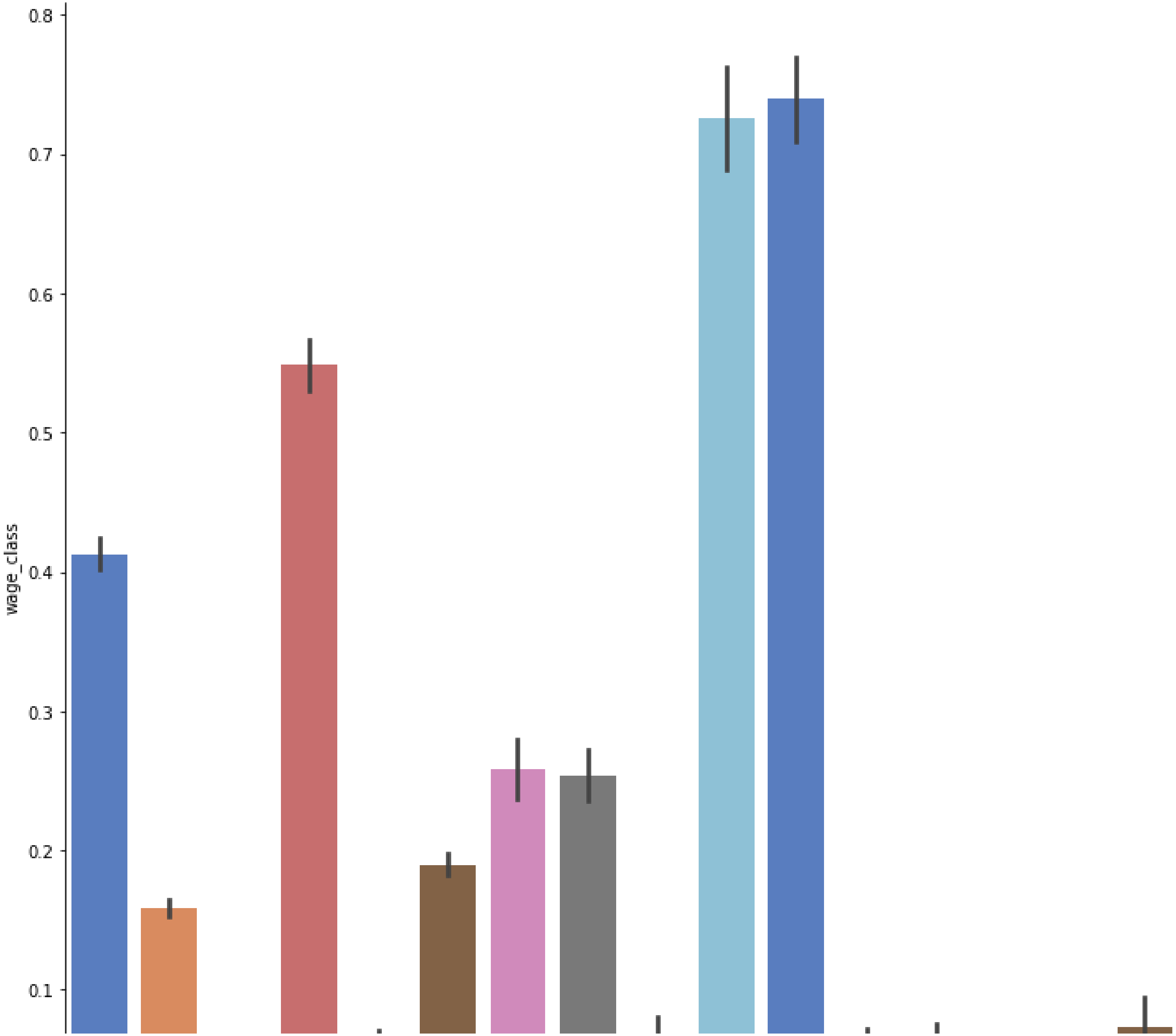
In [70]:
```
df['education'].value_counts()
```

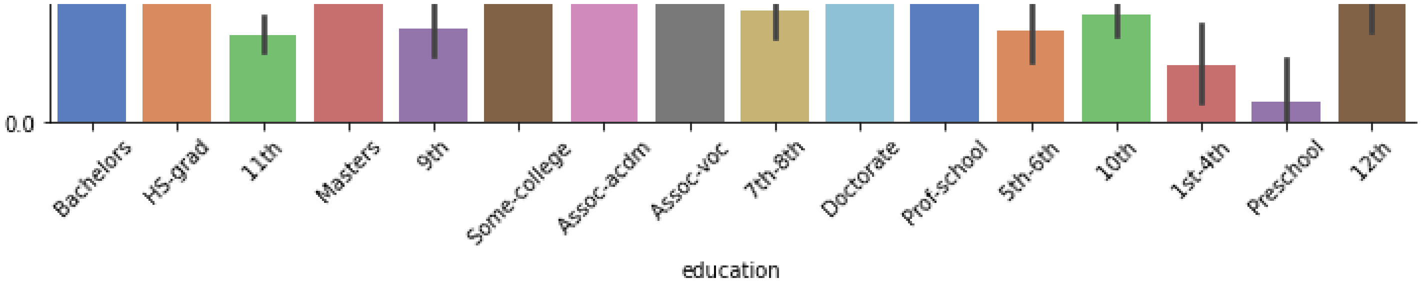
Out[70]:
```
 HS-grad         15784
 Some-college    10878
 Bachelors        8025
 Masters          2657
 Assoc-voc        2061
 11th             1812
 Assoc-acdm       1601
 10th             1389
 7th-8th           955
 Prof-school       834
 9th               756
 12th              657
 Doctorate         594
 5th-6th           509
 1st-4th           247
 Preschool          83
Name: education, dtype: int64
```

In [71]:
```
df.columns
```

Out[71]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'wage_class'],
      dtype='object')
```

In [73]:
```
sns.catplot(x='education',y='wage_class',data=df,height=10,palette='muted',kind='bar')
plt.xticks(rotation=45)
plt.show()
```

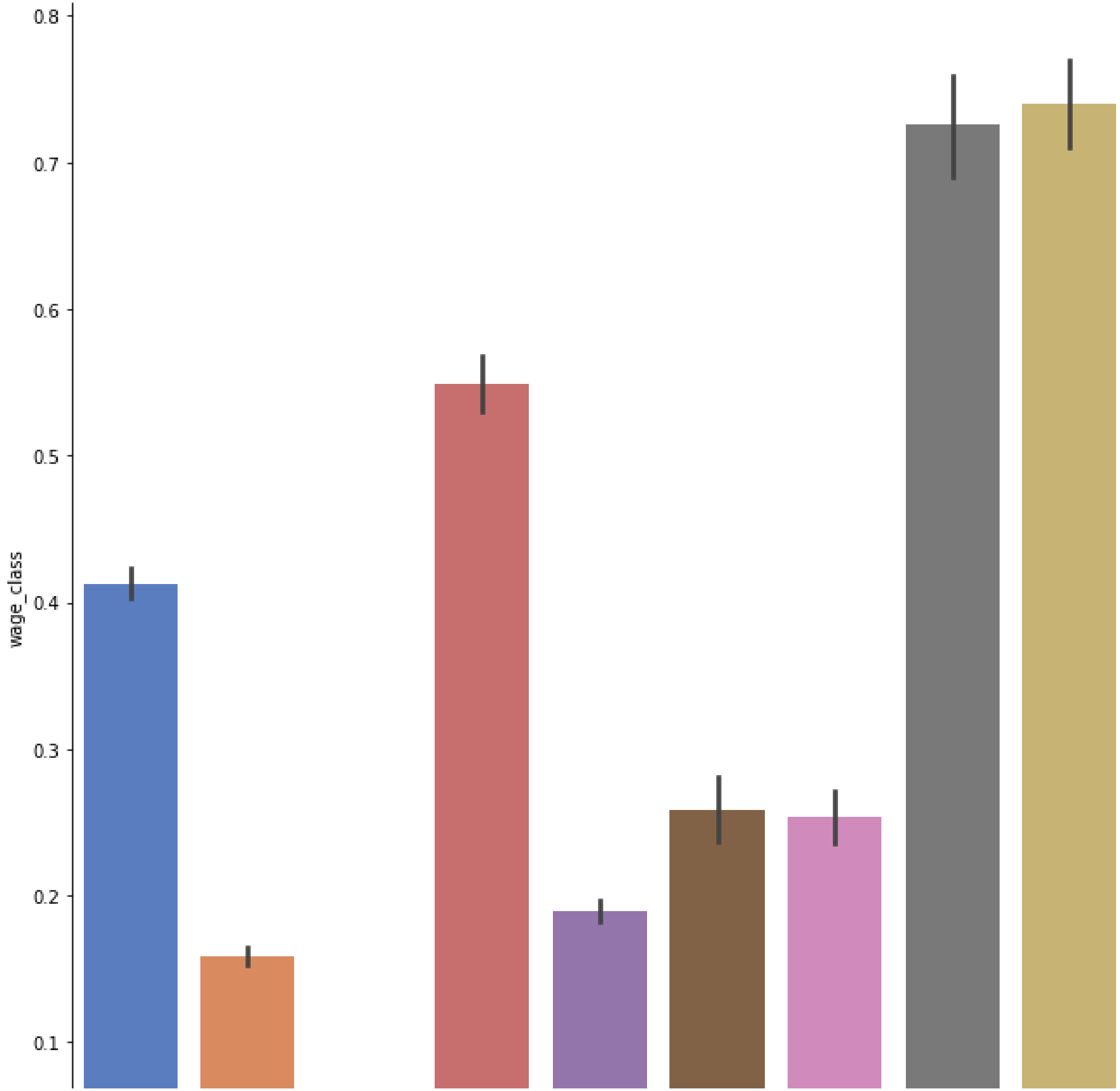


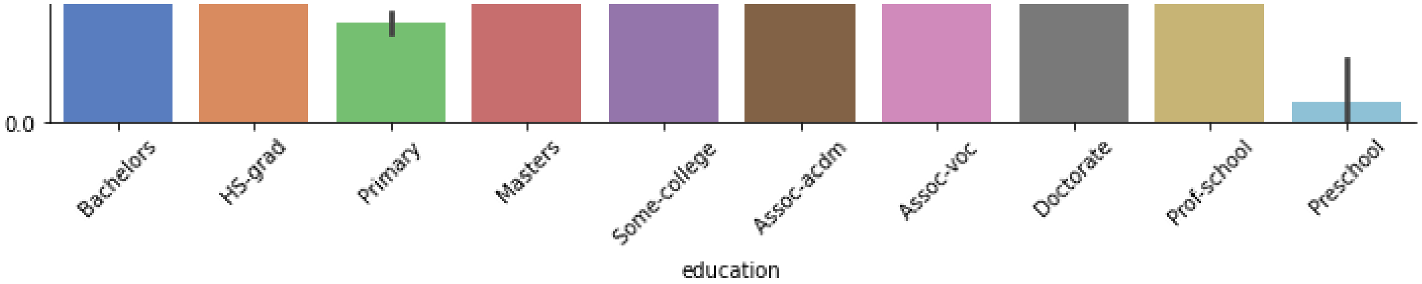


In [74]:
```
def primary(x):
    if x in [' 1st-4th', ' 5th-6th', ' 7th-8th', ' 9th', ' 10th', ' 11th', ' 12th']:
        return 'Primary'
    else:
        return x
```

In [75]:
```
df['education'] = df['education'].apply(primary)
```

In [76]:
```
sns.catplot(x='education',y='wage_class',data=df,height=10,palette='muted',kind='bar')
plt.xticks(rotation=45)
plt.show()
```

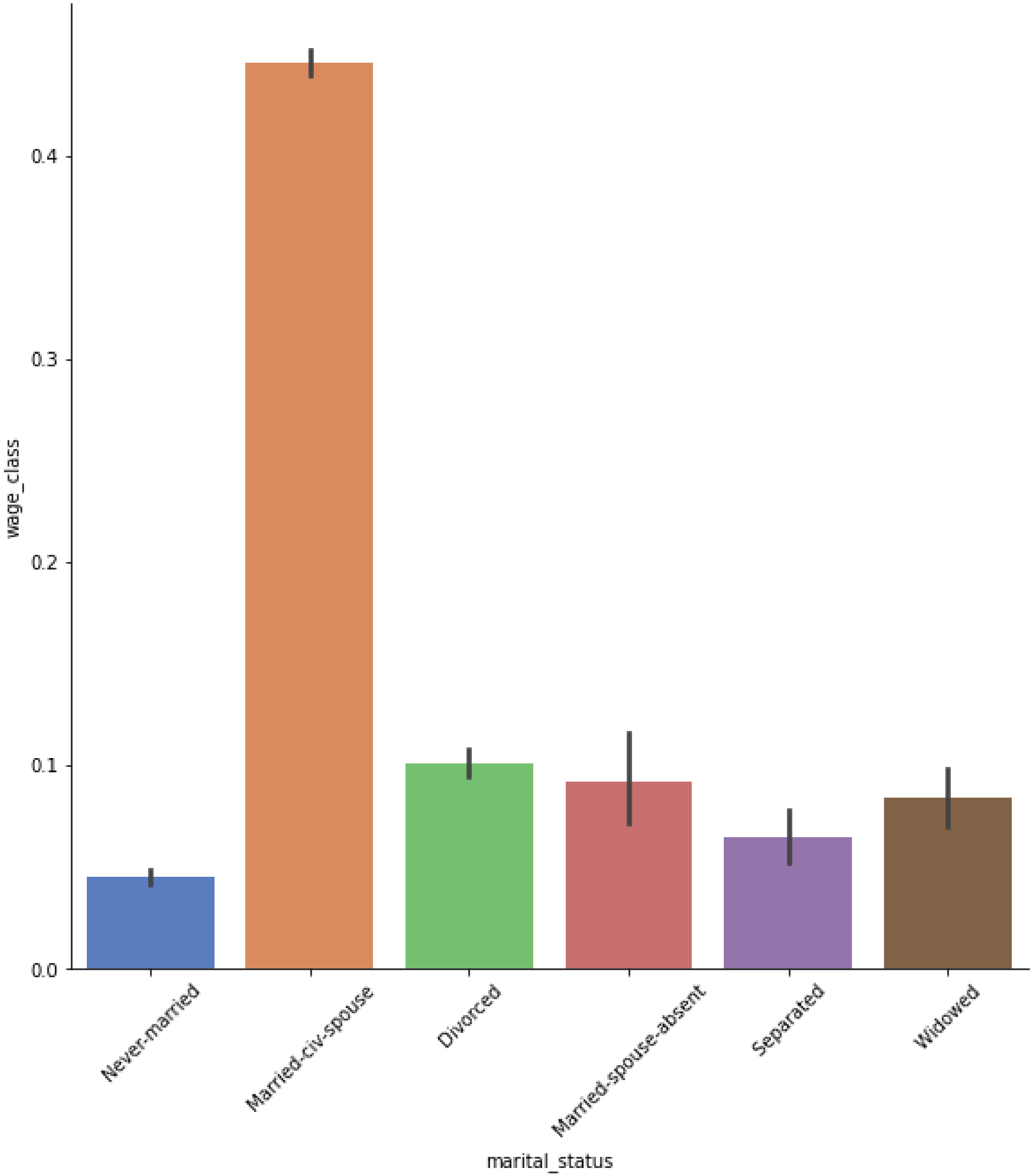




In [77]:
```
df['marital_status'].replace(' Married-AF-spouse', ' Married-civ-spouse',inplace=True)
```

In [78]:

```
sns.catplot(x='marital_status',y='wage_class',data=df,palette='muted',kind='bar',height=8)
plt.xticks(rotation=45)
plt.show()
```

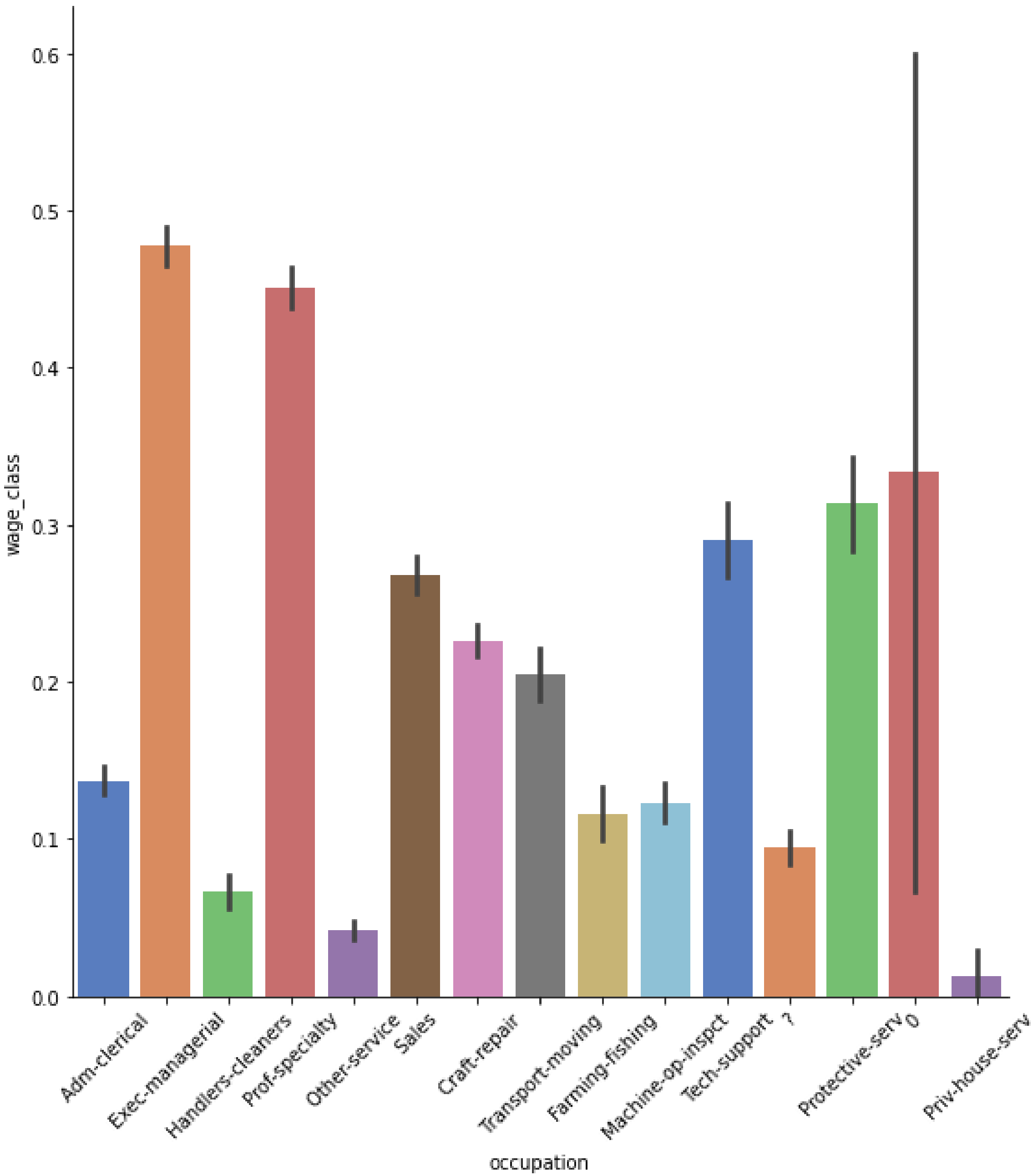

wage_class
0.0
0.1
0.2
0.3
0.4
Never-married
Married-civ-spouse
Divorced
Married-spouse-absent
Separated
Widowed
marital_status

In [79]:

```
df['occupation'].fillna('0',inplace=True)
df['occupation'].value_counts()
df['occupation'].replace(' Armed-Forces','0',inplace=True)
df['occupation'].value_counts()
sns.catplot(x='occupation',y='wage_class',data=df,palette='muted',kind='bar',height=8)
plt.xticks(rotation=45)
```

Out[79]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 [Text(0, 0, ' Adm-clerical'),
  Text(1, 0, ' Exec-managerial'),
  Text(2, 0, ' Handlers-cleaners'),
  Text(3, 0, ' Prof-specialty'),
  Text(4, 0, ' Other-service'),
  Text(5, 0, ' Sales'),
  Text(6, 0, ' Craft-repair'),
  Text(7, 0, ' Transport-moving'),
  Text(8, 0, ' Farming-fishing'),
  Text(9, 0, ' Machine-op-inspct'),
  Text(10, 0, ' Tech-support'),
  Text(11, 0, ' ?'),
  Text(12, 0, ' Protective-serv'),
  Text(13, 0, '0'),
  Text(14, 0, ' Priv-house-serv')])
```



wage_class
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Adm-clerical
Exec-managerial
Handlers-cleaners
Prof-specialty
Other-service
Sales
Craft-repair
Transport-moving
Farming-fishing
Machine-op-inspct
Tech-support
?
Protective-serv
0
Priv-house-serv
occupation

In [81]:
```python
df['relationship'].value_counts()
```

Out[81]:
```
 Husband           19716
 Not-in-family     12583
 Own-child          7581
 Unmarried          5125
 Wife               2331
 Other-relative     1506
Name: relationship, dtype: int64
```

In [82]:
```python
df['race'].value_counts()
```

Out[82]:
```
 White                 41762
 Black                  4685
 Asian-Pac-Islander     1519
 Amer-Indian-Eskimo      470
 Other                   406
Name: race, dtype: int64
```

In [83]:
```python
df.columns
```

Out[83]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'wage_class'],
      dtype='object')
```

In [84]:
```python
df['sex'].value_counts()
```

Out[84]:
```
 Male      32650
 Female    16192
Name: sex, dtype: int64
```

In [85]:
```python
df['native_country'].unique()
```

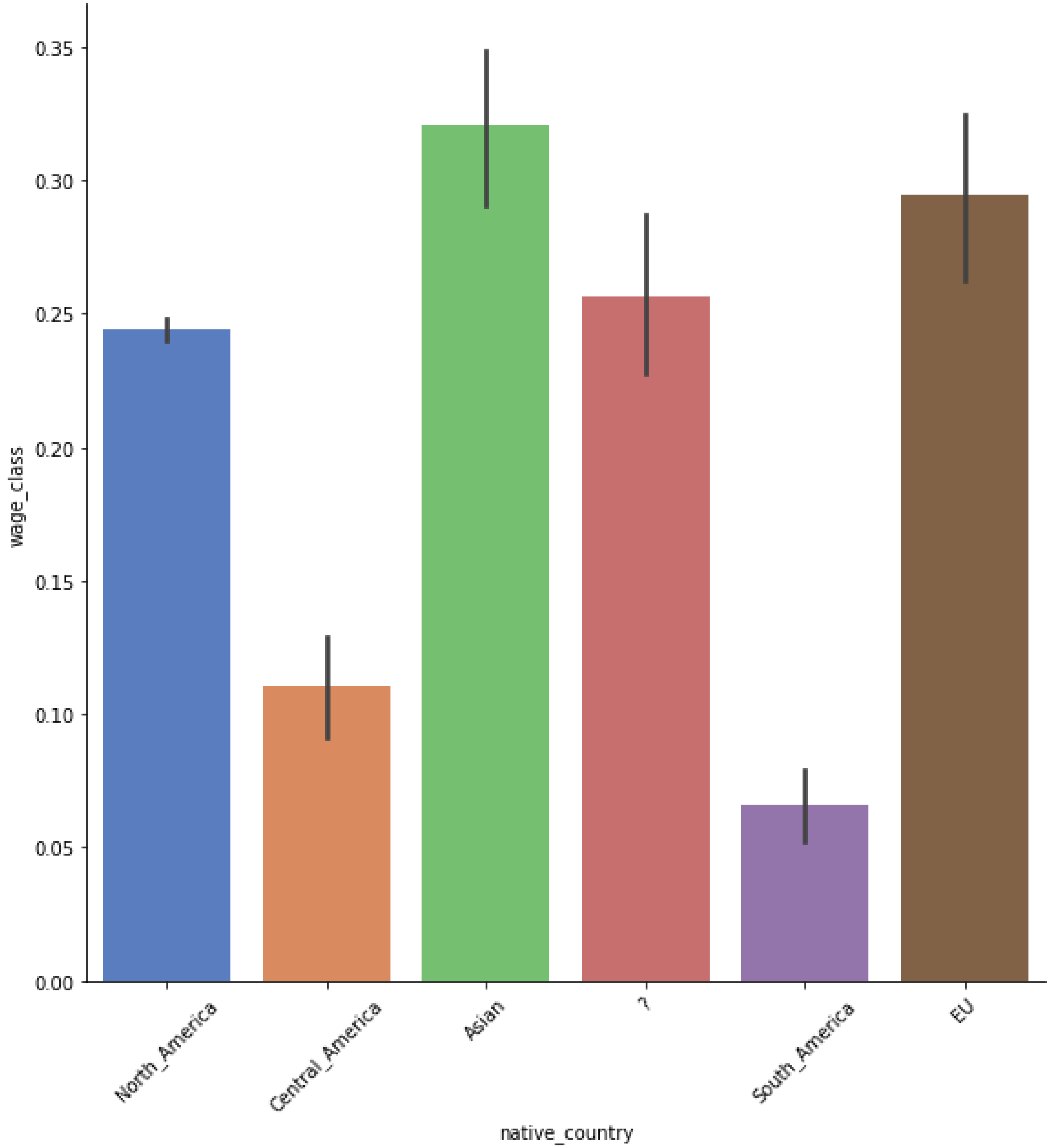
Out[85]:
```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' ?', ' Mexico',
       ' South', ' Puerto-Rico', ' Honduras', ' England', ' Canada',
       ' Germany', ' Iran', ' Philippines', ' Italy', ' Poland',
       ' Columbia', ' Cambodia', ' Thailand', ' Ecuador', ' Laos',
       ' Taiwan', ' 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'], dtype=object)
```

In [86]:
```python
def native(country):
    if country in [' United-States',' Canada']:
        return 'North_America'
    elif country in [' Puerto-Rico',' El-Salvador',' Cuba',' Jamaica',' Dominican-Republic',' Guatemala',' Haiti',' Nic
        return 'Central_America'
    elif country in [' Mexico',' Columbia',' Vietnam',' Peru',' Ecuador',' South',' Outlying-US(Guam-USVI-etc)']:
        return 'South_America'
    elif country in [' Germany',' England',' Italy',' Poland',' Portugal',' Greece',' Yugoslavia',' France',' Ireland',
        return 'EU'
    elif country in [' India',' Iran',' China',' Japan',' Thailand',' Hong',' Cambodia',' Laos',' Philippines',' Taiwan
        return 'Asian'
    else:
        return country
```

In [88]:
```python
df['native_country'] = df['native_country'].apply(native)
```

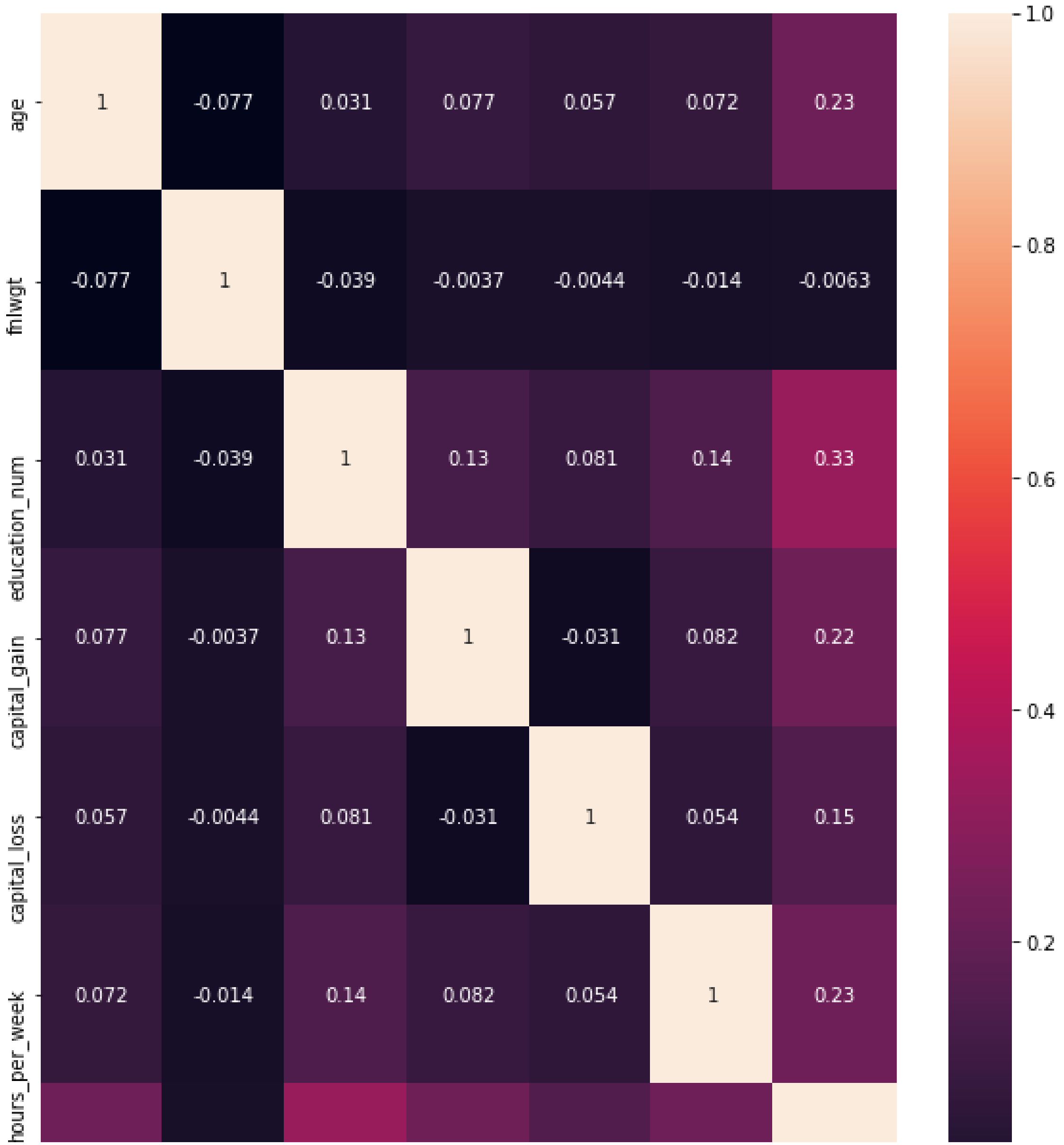
In [89]:
```
sns.catplot(x='native_country',y='wage_class',data=df,palette='muted',kind='bar',height=8)
plt.xticks(rotation=45)
```

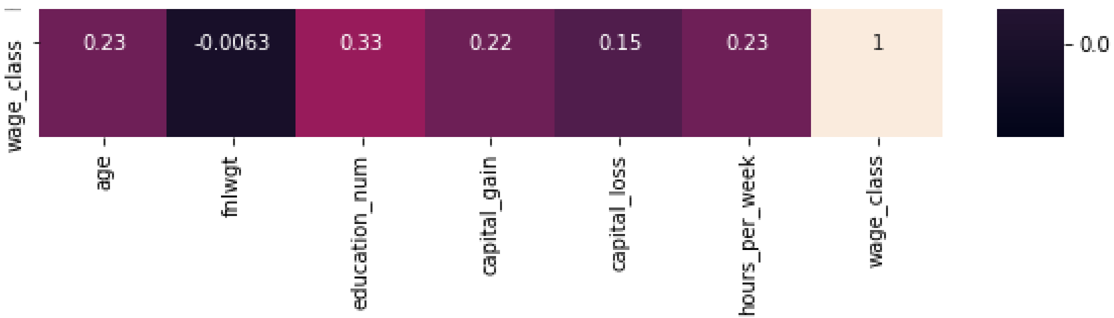
Out[89]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'North_America'),
  Text(1, 0, 'Central_America'),
  Text(2, 0, 'Asian'),
  Text(3, 0, ' ?'),
  Text(4, 0, 'South_America'),
  Text(5, 0, 'EU')])
```


0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
wage_class
North_America
Central_America
Asian
?
South_America
EU
native_country

In [90]:
```
corr = df.corr()
plt.figure(figsize=(10,12))
sns.heatmap(corr,annot=True)
```

Out[90]: <AxesSubplot:>





In [91]:
```
X = df.drop(['wage_class'],axis=1)
y = df['wage_class']
```

In [92]:
```
X.columns
```

Out[92]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country'],
      dtype='object')
```

In [94]:
```
X_dummy = pd.get_dummies(X)
```

In [95]:
```
X_dummy.head()
```

Out[95]:

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week	workclass_ ?	workclass_ Federal-gov	workclass_ Local-gov	workclass_ Never-worked	...	race_ Other	race_ White	Fe
0	39	77516	13	2174	0	40	0	0	0	0	...	0	1	
1	50	83311	13	0	0	13	0	0	0	0	...	0	1	
2	38	215646	9	0	0	40	0	0	0	0	...	0	1	
3	53	234721	7	0	0	40	0	0	0	0	...	0	0	
4	28	338409	13	0	0	40	0	0	0	0	...	0	0	

5 rows × 65 columns

In [96]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_d)
```

In [97]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.3,random_state=101)
```

In [98]:
```
parameters = [{ 'learning_rate':[0.01,0.001],
                'max_depth': [3,5,10],
                'n_estimators':[10,50,100,200]
              }
             ]
```

In [102]:
```python
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
Xbc = XGBClassifier()
Grid_cv = GridSearchCV(Xbc,parameters,scoring='accuracy',cv=5,n_jobs=3,verbose=3)
Grid_cv.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits

C:\Users\kants\AppData\Local\Programs\Python\Python37\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use
of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do th
e following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels
(y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[14:33:37] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBo
ost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[102]:
```
GridSearchCV(cv=5,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, gamma=None,
                                     gpu_id=None, importance_type='gain',
                                     interaction_constraints=None,
                                     learning_rate=None, max_delta_step=None,
                                     max_depth=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     n_estimators=100, n_jobs=None,
                                     num_parallel_tree=None, random_state=None,
                                     reg_alpha=None, reg_lambda=None,
                                     scale_pos_weight=None, subsample=None,
                                     tree_method=None, validate_parameters=None,
                                     verbosity=None),
             n_jobs=3,
             param_grid=[{'learning_rate': [0.01, 0.001],
                          'max_depth': [3, 5, 10],
                          'n_estimators': [10, 50, 100, 200]}],
             scoring='accuracy', verbose=3)
```

In [104]:
```
Grid_cv.best_params_
```
Out[104]:
```
{'learning_rate': 0.01, 'max_depth': 10, 'n_estimators': 200}
```
In [105]:
```
XBC = XGBClassifier(learning_rate=0.01,max_depth=10,n_estimators=200)
XBC.fit(x_train,y_train)
```
```
[14:38:05] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBo
ost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
```
Out[105]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.01, max_delta_step=0, max_depth=10,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=200, n_jobs=8, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```
In [106]:
```
XBC.score(x_test,y_test)
```
Out[106]:
```
0.8658295229645806
```
In [108]:
```
y_pred = XBC.predict(x_test)
```
In [109]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [112]:

```
print(f'Accuracy Score:{accuracy_score(y_test,y_pred)}')
print(f'Confusion Matrix:{confusion_matrix(y_test,y_pred)}')
print(f'Classification Report: {classification_report(y_test,y_pred)}')
```

```
Accuracy Score:0.8658295229645806
Confusion Matrix:[[10536   564]
 [ 1402  2151]]
Classification Report:               precision    recall  f1-score   support

           0       0.88      0.95      0.91     11100
           1       0.79      0.61      0.69      3553

    accuracy                           0.87     14653
   macro avg       0.84      0.78      0.80     14653
weighted avg       0.86      0.87      0.86     14653
```

In [ ]: