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

In [43]:
```
df=pd.read_csv('bank_nifty.csv')
```

In [44]:
```
df.head()
```

Out[44]:

	Date	Open	High	Low	Close	Volume
0	1/1/2018	25565.75	25588.00	25271.55	25318.10	57576913
1	1/2/2018	25382.20	25425.50	25232.80	25338.25	72033811
2	1/3/2018	25425.75	25454.90	25300.90	25318.60	59730356
3	1/4/2018	25367.65	25490.35	25310.30	25462.60	105995860
4	1/5/2018	25524.45	25643.35	25499.55	25601.85	123622612

In [45]:
```
df.describe()
```

Out[45]:

	Open	High	Low	Close
count	955.000000	955.000000	955.000000	955.000000
mean	28575.225969	28805.647958	28291.096178	28550.485812
std	4823.393891	4813.805396	4831.728274	4827.044525
min	16759.950000	17681.700000	16116.250000	16917.650000
25%	25487.200000	25651.575000	25252.175000	25443.475000
50%	27972.950000	28185.150000	27777.400000	28021.700000
75%	31523.500000	31742.175000	31240.750000	31515.000000
max	41234.550000	41829.600000	40829.150000	41238.300000

In [46]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 955 entries, 0 to 954
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    955 non-null    object 
 1   Open    955 non-null    float64
 2   High    955 non-null    float64
 3   Low     955 non-null    float64
 4   Close   955 non-null    float64
 5   Volume  955 non-null    object 
dtypes: float64(4), object(2)
memory usage: 44.9+ KB
```

In [47]:
```
df.isna().sum()  #no missing values
```

Out[47]:
```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

In [80]:
```
df.corr()
```

Out[80]:

	Open	High	Low	Close
Open	1.000000	0.998461	0.998387	0.996952
High	0.998461	1.000000	0.997471	0.998621
Low	0.998387	0.997471	1.000000	0.998409
Close	0.996952	0.998621	0.998409	1.000000

All the variables have a very high correlation with each other

In [81]:
```
x=df['High']
y=df['Close']
```

Linear Regression Model

In [82]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

In [83]:
```
from sklearn.linear_model import LinearRegression
```

In [84]:
```
model = LinearRegression()
```

In [85]:
```
m=model.fit(np.array(x_train).reshape(len(np.array(x_train)),-1),y_train)
```

In [90]:
```
y_pred=m.predict(np.array(x_test).reshape(len(np.array(x_test)),-1))
```

In [91]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[91]:
```
0.9964581228659305
```

The model is a very good fit giving an accuracy of 99.6458%

Multiple Linear Model

In [129]:
```
x=df.iloc[:,1:4]
y=df['Close']
```

In [130]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

In [132]:
```
model = LinearRegression().fit(np.array(x_train).reshape(len(np.array(x_train)),-1),y_train)
```

In [134]:
```
y_pred=model.predict(np.array(x_test).reshape(len(np.array(x_test)),-1))
```

In [135]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[135]:
```
0.9991564063177207
```

The model has an accuracy of 99.9156% and is a better fit than linear regression model

Taking a new dataset

In [75]:
```
data=pd.read_csv('adult.csv')
```

In [76]:
```
data.head()
```

Out[76]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States

In [77]:
```
data.isnull().sum() #No null values
```

Out[77]:
```
age                0
workclass          0
fnlwgt             0
education          0
educational-num    0
marital-status     0
occupation         0
relationship       0
race               0
gender             0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country     0
income             0
dtype: int64
```

In [78]:
```
data.info()
```

```
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              48842 non-null  int64 
 1   workclass        48842 non-null  object
 2   fnlwgt           48842 non-null  int64 
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64 
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64 
 11  capital-loss     48842 non-null  int64 
 12  hours-per-week   48842 non-null  int64 
 13  native-country   48842 non-null  object
 14  income           48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

In [79]:
```
data['income']=data['income'].map({data.income.unique()[0]:0,data.income.unique()[1]:1})
```

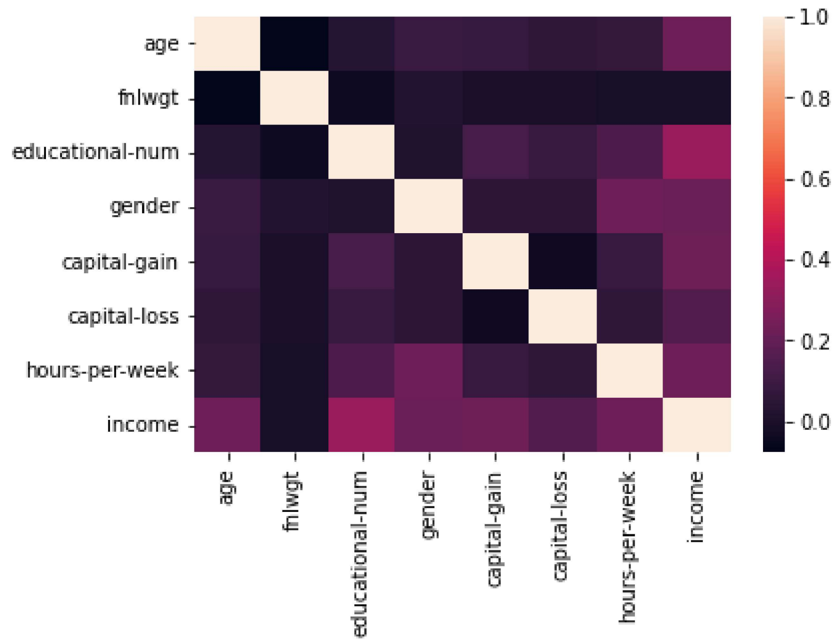
In [82]:
```
data['gender']=data['gender'].map({data.gender.unique()[0]:1,data.gender.unique()[1]:0})
```

In [83]:
```
data.gender.unique()[0]
```

Out[83]:
```
1
```

In [84]:
```
sns.heatmap(data.corr())
```

Out[84]: <AxesSubplot:>

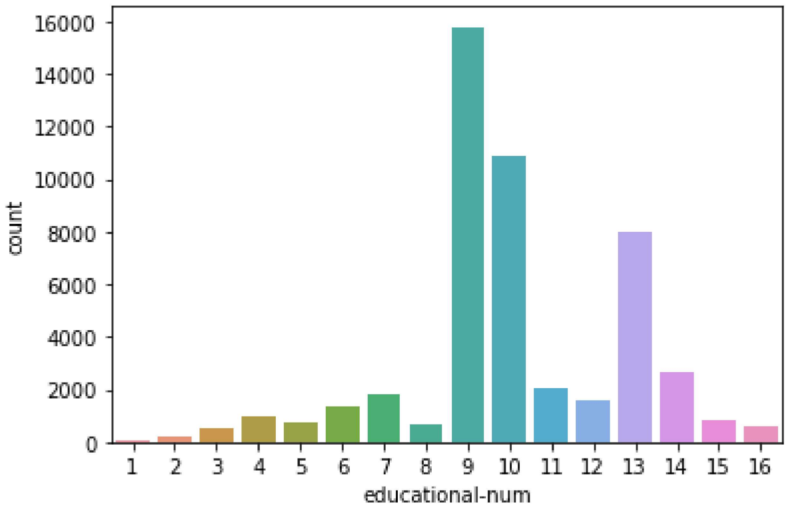


We can observe from the data that income is weekly correlated with gender.Hence the variable doesn't have a major impact on the variable.

In [6]:
```
sns.countplot(data['educational-num'])
```

```
C:\Users\Admin\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variab
le as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[6]:
```
<AxesSubplot:xlabel='educational-num', ylabel='count'>
```



In [7]:
```
d=pd.get_dummies(data,drop_first=True)
d.head()
```

Out[7]:

native-y_South	native-country_Taiwan	native-country_Thailand	native-country_Trinadad&Tobago	native-country_United-States	native-country_Vietnam	native-country_Yugoslavia	income_>50K
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	1
0	0	0	0	1	0	0	1
0	0	0	0	1	0	0	0

In [52]:
```
d.isnull().sum()   #no missing values
```

Out[52]:
```
age                               0
fnlwgt                            0
educational-num                   0
capital-gain                      0
capital-loss                      0
                                 ..
native-country_Trinadad&Tobago    0
native-country_United-States      0
native-country_Vietnam            0
native-country_Yugoslavia         0
income_>50K                       0
Length: 101, dtype: int64
```

Logistic Model

In [14]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
model = LogisticRegression(random_state=0)
```

In [32]:
```
x=d.iloc[:,0:100]
y=d.iloc[:,-1]
```

In [36]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [38]:
```
m2=model.fit(x_train, y_train)
```

In [40]:
```
y_pr=m2.predict(x_test)
```

In [43]:
```
confusion_matrix(y_test,y_pr)
```

Out[43]:
```
array([[11942,   374],
       [ 2785,  1017]], dtype=int64)
```

In [48]:
```
classification_report(y_test,y_pr)
```

Out[48]:
```
'              precision    recall  f1-score   support\n\n           0       0.81      0.97      0.88     1231
6\n           1       0.73      0.27      0.39      3802\n\n    accuracy                           0.80     16
118\n   macro avg       0.77      0.62      0.64     16118\nweighted avg       0.79      0.80      0.77     16
118\n'
```

In [51]:
```
(11942+1017)/(11942+374+2785+1017)
```

Out[51]:
```
0.8040079414319394
```

Hence the accuracy of the logistic regression model is 80.4%