In [1]:

```
import numpy as np
import pandas as pd
df=pd.read_csv("/content/insurance_premium.csv")
df
```

Out[1]:

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86
...	...	...	...	...	...	...	...
1333	50	male	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83
1336	21	female	25.8	0	no	southwest	2007.95
1337	61	female	29.1	0	yes	northwest	29141.36

1338 rows × 7 columns

In [2]:

```
df.shape
```

Out[2]:

```
(1338, 7)
```

In [3]:

```
df.size
```

Out[3]:

```
9366
```

In [4]:

```
df.head()
```

Out[4]:

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86

In [5]:

```
df.tail()
```

Out[5]:

	age	sex	bmi	children	smoker	region	expenses
1333	50	male	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83
1336	21	female	25.8	0	no	southwest	2007.95
1337	61	female	29.1	0	yes	northwest	29141.36

In [6]:
```
df.columns
```

Out[6]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'expenses'], dt
ype='object')
```

In [7]:
```
df.dtypes
```

Out[7]:
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
expenses    float64
dtype: object
```

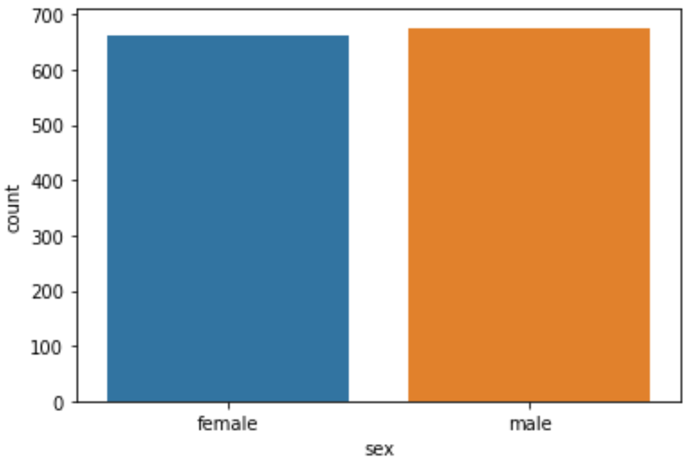
In [8]:
```
df.isna().sum()
```

Out[8]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
expenses    0
dtype: int64
```

In [19]:
```
# USING SEABORN TO VISUALIZE
import seaborn as sns
sns.countplot('sex',data=df)
```

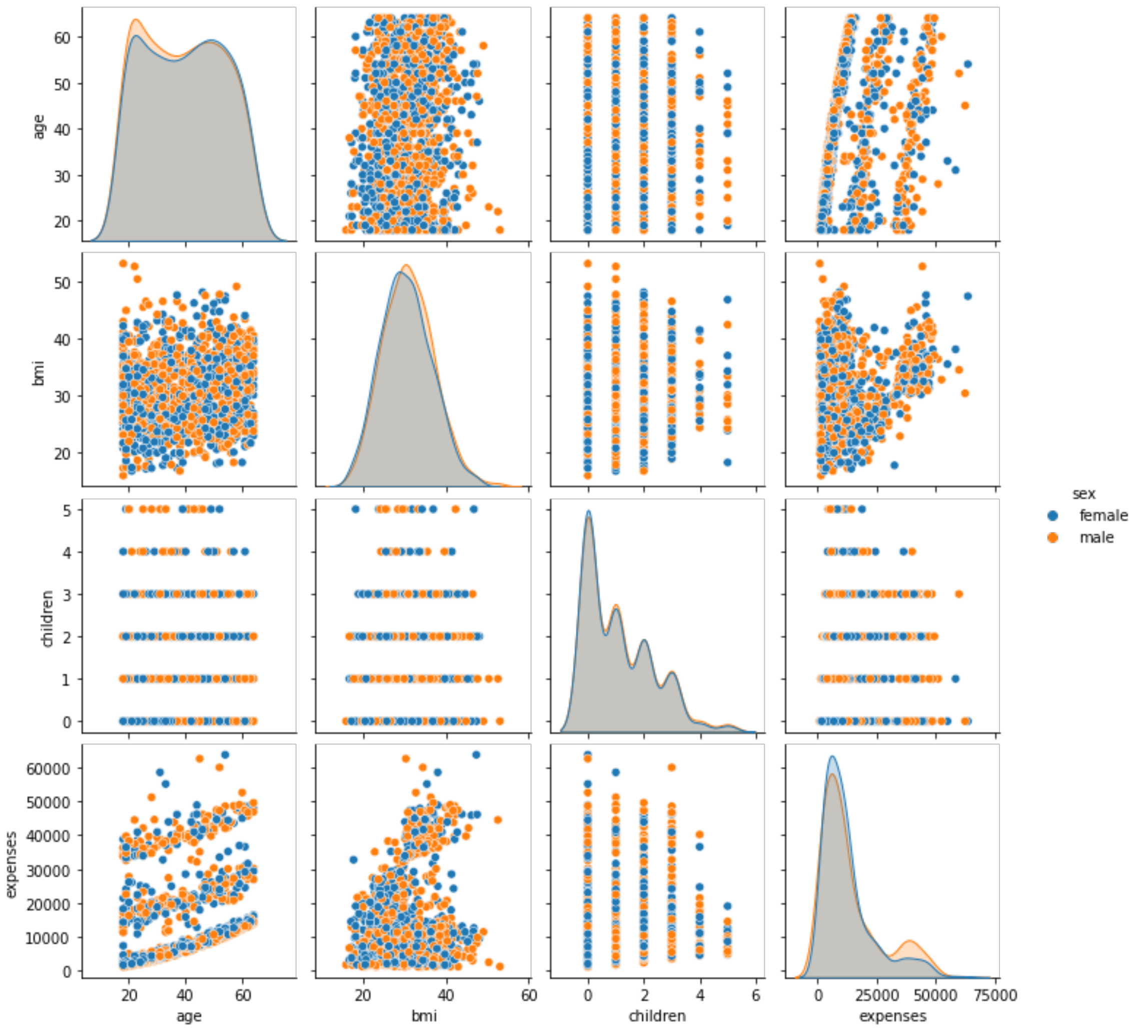
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbf8a6692e0>
```



```
In [35]: sns.pairplot(df,hue='sex')
```

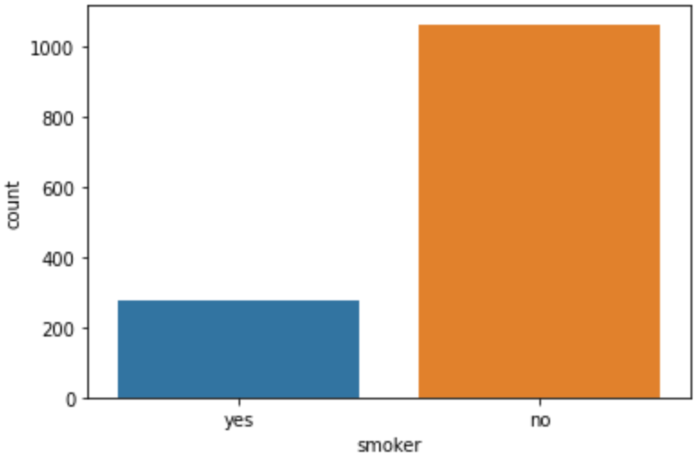
```
Out[35]: <seaborn.axisgrid.PairGrid at 0x7fbf880573a0>
```



```
In [36]: sns.countplot('smoker',data=df)
```

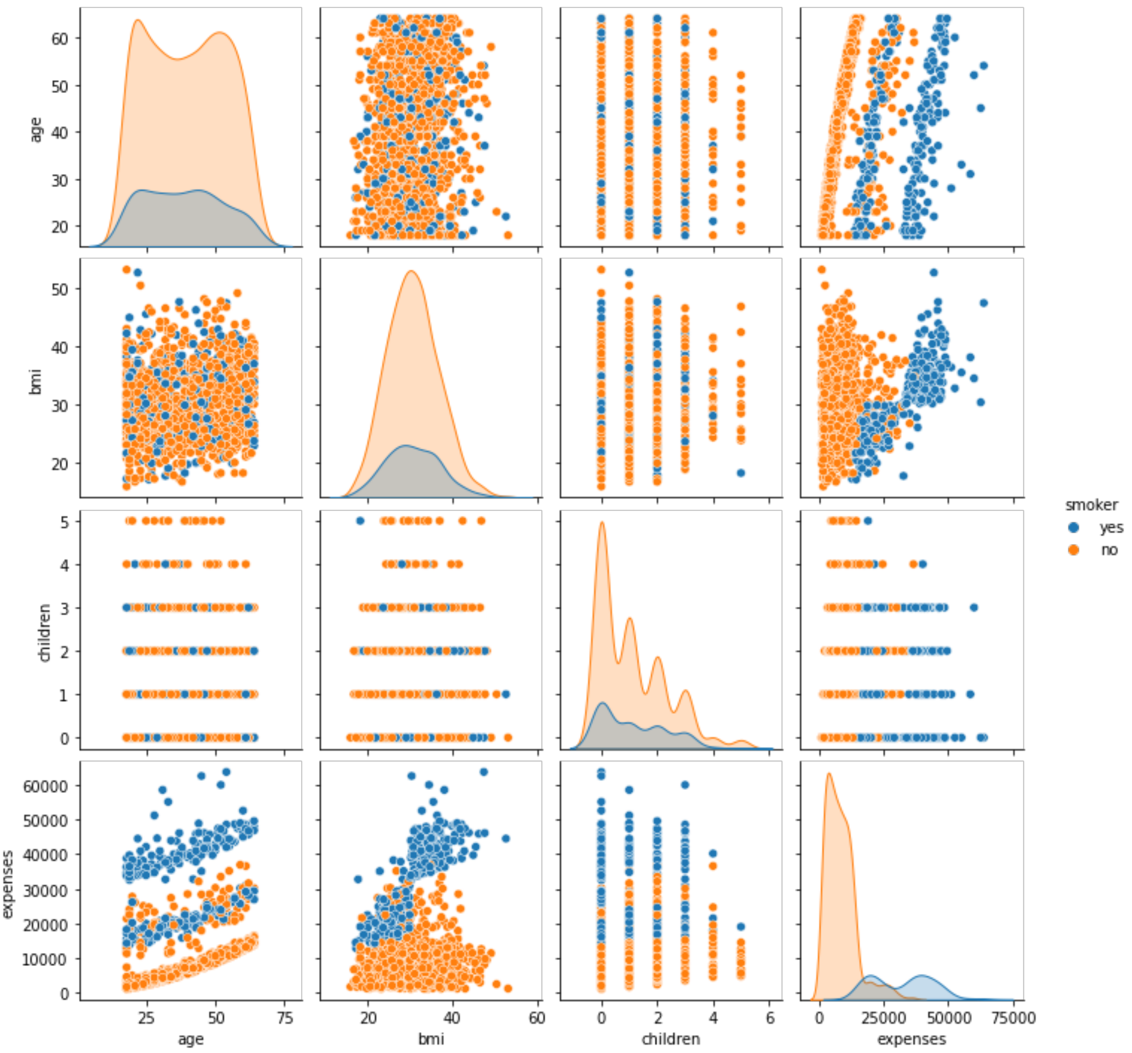
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbf85a62b80>
```



```
In [37]: sns.pairplot(df,hue='smoker')
```

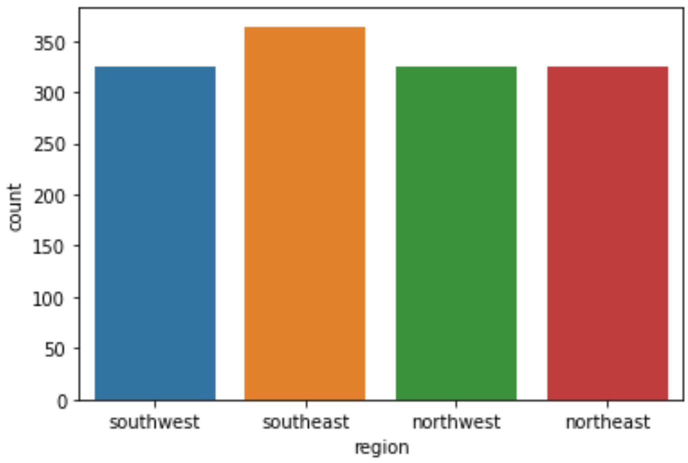
```
Out[37]: <seaborn.axisgrid.PairGrid at 0x7fbf85a0c550>
```



```
In [21]: sns.countplot('region',data=df)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbf8c69cc70>
```

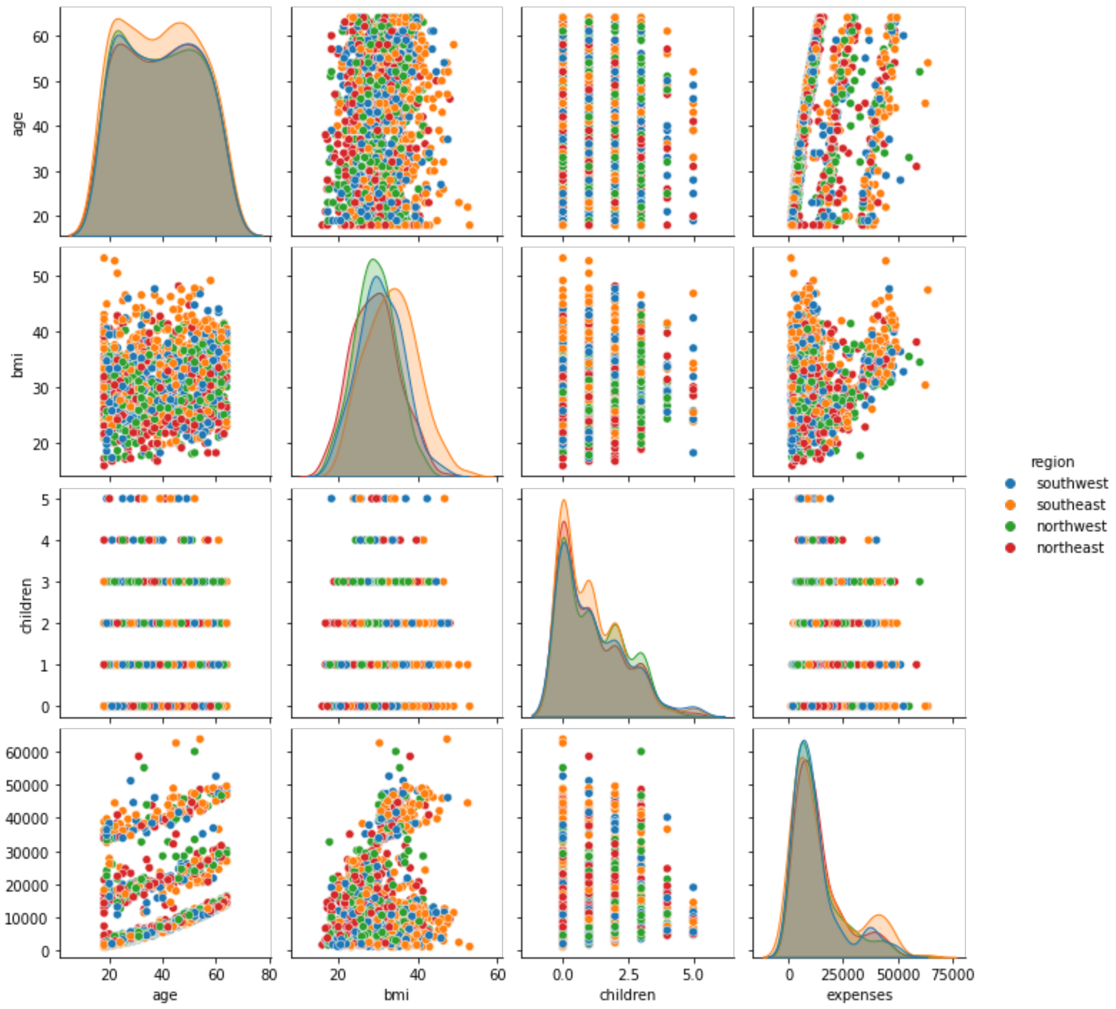


In [38]:

```
sns.pairplot(df,hue='region')
```

Out[38]:

```
<seaborn.axisgrid.PairGrid at 0x7fbf8548ac10>
```



In [23]:

```
# CONVERTING THE COLUMNS WITH STRING AS NUMERICAL VALUES
dummy1=pd.get_dummies(df[['sex','region','smoker']],drop_first=True)
dummy1
```

Out[23]:

	sex_male	region_northwest	region_southeast	region_southwest	smoker_yes
0	0	0	0	1	1
1	1	0	1	0	0
2	1	0	1	0	0
3	1	1	0	0	0
4	1	1	0	0	0
...	...	...	...	...	...
1333	1	1	0	0	0
1334	0	0	0	0	0
1335	0	0	1	0	0
1336	0	0	0	1	0
1337	0	1	0	0	1

1338 rows × 5 columns

In [25]:
```
df1=pd.concat([df,dummy1],axis=1)
df1
```

Out[25]:

	age	sex	bmi	children	smoker	region	expenses	sex_male	region_northwest	region_so
0	19	female	27.9	0	yes	southwest	16884.92	0	0	
1	18	male	33.8	1	no	southeast	1725.55	1	0	
2	28	male	33.0	3	no	southeast	4449.46	1	0	
3	33	male	22.7	0	no	northwest	21984.47	1	1	
4	32	male	28.9	0	no	northwest	3866.86	1	1	
...	...	...	...	...	...	...	...	...	...	
1333	50	male	31.0	3	no	northwest	10600.55	1	1	
1334	18	female	31.9	0	no	northeast	2205.98	0	0	
1335	18	female	36.9	0	no	southeast	1629.83	0	0	
1336	21	female	25.8	0	no	southwest	2007.95	0	0	
1337	61	female	29.1	0	yes	northwest	29141.36	0	1	

1338 rows × 12 columns

In [26]:
```
# DROP UNWANTED COLUMNS
df1=df1.drop(['sex','smoker','region'],axis=1)
df1
```

Out[26]:

	age	bmi	children	expenses	sex_male	region_northwest	region_southeast	region_southwest
0	19	27.9	0	16884.92	0	0	0	1
1	18	33.8	1	1725.55	1	0	1	0
2	28	33.0	3	4449.46	1	0	1	0
3	33	22.7	0	21984.47	1	1	0	0
4	32	28.9	0	3866.86	1	1	0	0
...	...	...	...	...	...	...	...	...
1333	50	31.0	3	10600.55	1	1	0	0
1334	18	31.9	0	2205.98	0	0	0	0
1335	18	36.9	0	1629.83	0	0	1	0
1336	21	25.8	0	2007.95	0	0	0	1
1337	61	29.1	0	29141.36	0	1	0	0

1338 rows × 9 columns

In [28]:
```
df1.columns
```

Out[28]:
```
Index(['age', 'bmi', 'children', 'expenses', 'sex_male', 'region_northwest
',
       'region_southeast', 'region_southwest', 'smoker_yes'],
      dtype='object')
```

```
In [29]: df1.dtypes
```

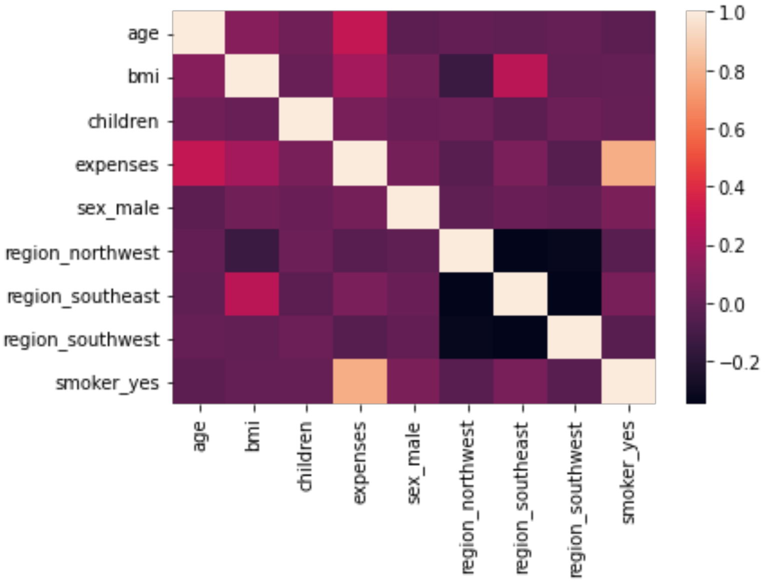
```
Out[29]: age                    int64
         bmi                  float64
         children               int64
         expenses             float64
         sex_male               uint8
         region_northwest       uint8
         region_southeast       uint8
         region_southwest       uint8
         smoker_yes             uint8
         dtype: object
```

```
In [31]: df1.isna().sum()
```

```
Out[31]: age                 0
         bmi                 0
         children            0
         expenses            0
         sex_male            0
         region_northwest    0
         region_southeast    0
         region_southwest    0
         smoker_yes          0
         dtype: int64
```

```
In [33]: sns.heatmap(df1.corr())
```

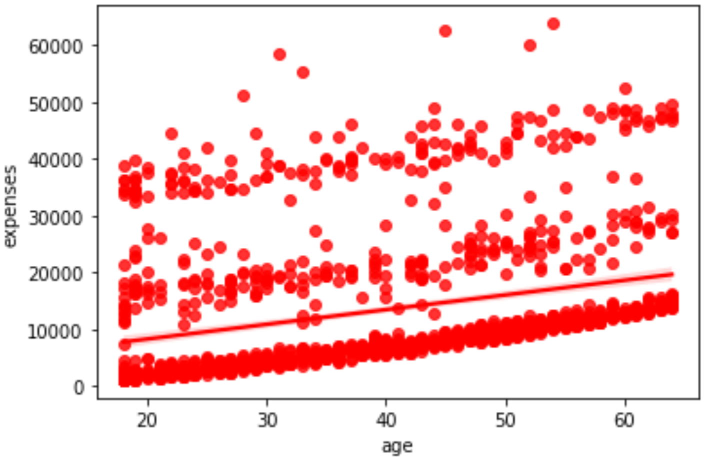
```
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbf89b1c730>
```



```
In [44]: # SEPERATING INPUT X AND OUTPUT Y
         x=df1.drop(['expenses'],axis=1)
         y=df1['expenses']
```

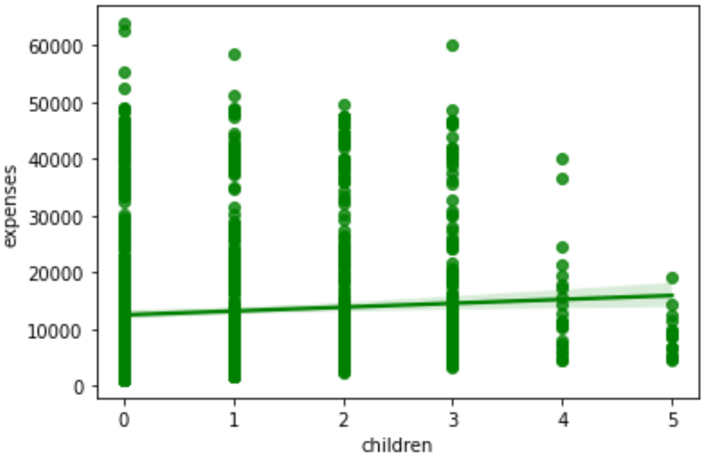
In [55]:
```
sns.regplot(x=df['age'],y=y,color='red')
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbf83dbb640>
```



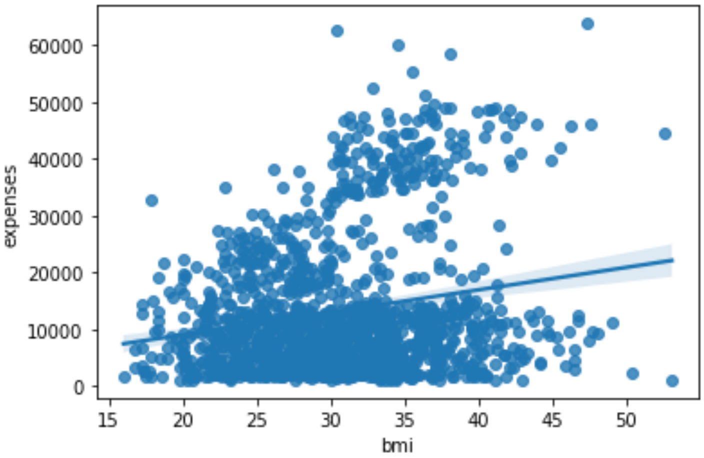
In [56]:
```
sns.regplot(x=df['children'],y=y,color='green')
```

Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbf83d27460>
```



In [47]:
```
sns.regplot(x=df['bmi'],y=y)
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbf84959b80>
```



In [48]:
```
# SEPERATING TRAIN AND TEST DATA
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_st
ate=42)
```

In [49]:

```
# MODEL CREATION
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```

In [50]:

```
# CALCULATE SLOPE:COEFFICIENT AND CONSTANT:INTERCEPT
print("SLOPE IS:",model.coef_)
list(zip(x,model.coef_))
```

```
SLOPE IS: [  261.28251281   348.96600937   424.41067944   104.99524716
  -486.46995207  -970.61815579  -925.06307896 23627.8945956 ]
```

Out[50]:

```
[('age', 261.2825128136763),
 ('bmi', 348.96600937445476),
 ('children', 424.41067943856666),
 ('sex_male', 104.99524716230803),
 ('region_northwest', -486.4699520736093),
 ('region_southeast', -970.6181557875361),
 ('region_southwest', -925.063078956111),
 ('smoker_yes', 23627.894595596354)]
```

In [51]:

```
print("CONSTANT IS:",model.intercept_)
```

```
CONSTANT IS: -12376.785237284628
```

In [52]:

```
# DATAFRAME HAVING ACTUAL VALUE AND PREDICTED VALUE
df2=pd.DataFrame({'ACTUAL_VALUE':y_test,'PREDICTED_VALUE':y_pred})
df2
```

Out[52]:

	ACTUAL_VALUE	PREDICTED_VALUE
764	9095.07	9023.692634
887	5272.18	7011.895553
890	29330.98	36873.905878
1293	9301.89	9502.394126
259	33750.29	26966.018096
...	...	...
701	9541.70	16286.121022
672	4399.73	6528.052794
1163	2200.83	2167.141458
1103	11363.28	14509.650536
1295	1964.78	130.460073

402 rows × 2 columns

In [54]:

```
## PERFORMANCE MEASUREMENT

from sklearn.metrics import  mean_absolute_error,mean_absolute_percentage_error,mean_squared_error,r2_score
print("ERROR IS:",mean_absolute_error(y_test,y_pred))
print("PERCENTAGE ERROR:",mean_absolute_percentage_error(y_test,y_pred))
print("SQUARED ERROR:",mean_squared_error(y_test,y_pred))
print("ROOT MEAN SQUARED ERROR:",np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2_SCORE:",r2_score(y_test,y_pred))
```

```
ERROR IS: 4144.88640999345
PERCENTAGE ERROR: 0.4358069585830062
SQUARED ERROR: 33777093.10084606
ROOT SQUARED ERROR: 5811.806354382952
R2_SCORE: 0.7696351080608885
```