Mini project - Insurance

In [ ]:
```
# problem  statements

# age - age of policyholder
# sex - male(1)/female(0)
# bmi - body mass index (kg/m2)
# children - number of children/ dependents of policyholder
# smoker - smoking state nonsmoke(0)/ smoker(1)
# region - residental area northeast(0)/northwest(1)/southeast(2)/southwest(3)
# charges - medical cost
# insuranceclaim - yes(1)/no(0)
```

Load libraries

In [18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [8]:
```
# loading data

data= pd.read_csv('insurance.csv')
data.head()
```

Out[8]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
0	19	0	27.900	0	1	3	16884.92400	1
1	18	1	33.770	1	0	2	1725.55230	1
2	28	1	33.000	3	0	2	4449.46200	0
3	33	1	22.705	0	0	1	21984.47061	0
4	32	1	28.880	0	0	1	3866.85520	1

In [ ]:
```
# here insurance is our - target variable
# others are categorical features(such as sex,smoker , region )
```

In [ ]:
```
# EDA (Exploratory Data Analysis)-
```

In [3]:
```
# shape - it shows the no of rows and columns in dataset
data.shape                      # 1338 rows and 8 columns
```

Out[3]:
```
(1338, 8)
```

In [4]:
```
# size - product of no of rows and no of column
data.size
```

Out[4]:
```
10704
```

In [5]:
```
# Data types
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             1338 non-null   int64
 1   sex             1338 non-null   int64
 2   bmi             1338 non-null   float64
 3   children        1338 non-null   int64
 4   smoker          1338 non-null   int64
 5   region          1338 non-null   int64
 6   charges         1338 non-null   float64
 7   insuranceclaim  1338 non-null   int64
dtypes: float64(2), int64(6)
memory usage: 83.8 KB
```

In [6]:
```
# unique = it shows different data types present in the data
data.dtypes.unique()
```

Out[6]:
```
array([dtype('int64'), dtype('float64')], dtype=object)
```

In [7]:
```
# unique values for children
sorted(data['children'].unique())
```

Out[7]:
```
[0, 1, 2, 3, 4, 5]
```

In [8]:
```
sorted(data['sex'].unique())
```

Out[8]:
```
[0, 1]
```

In [9]:
```
# column = it shows column

data.columns
```

Out[9]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges',
       'insuranceclaim'],
      dtype='object')
```

In [10]:
```
# Data cleaning - check missing or null values
data.isnull().sum()

# (here dont have any missing values )
```

Out[10]:
```
age               0
sex               0
bmi               0
children          0
smoker            0
region            0
charges           0
insuranceclaim    0
dtype: int64
```

In [11]:
```
# describe - gives statistical measure of dataset
data.describe()
```

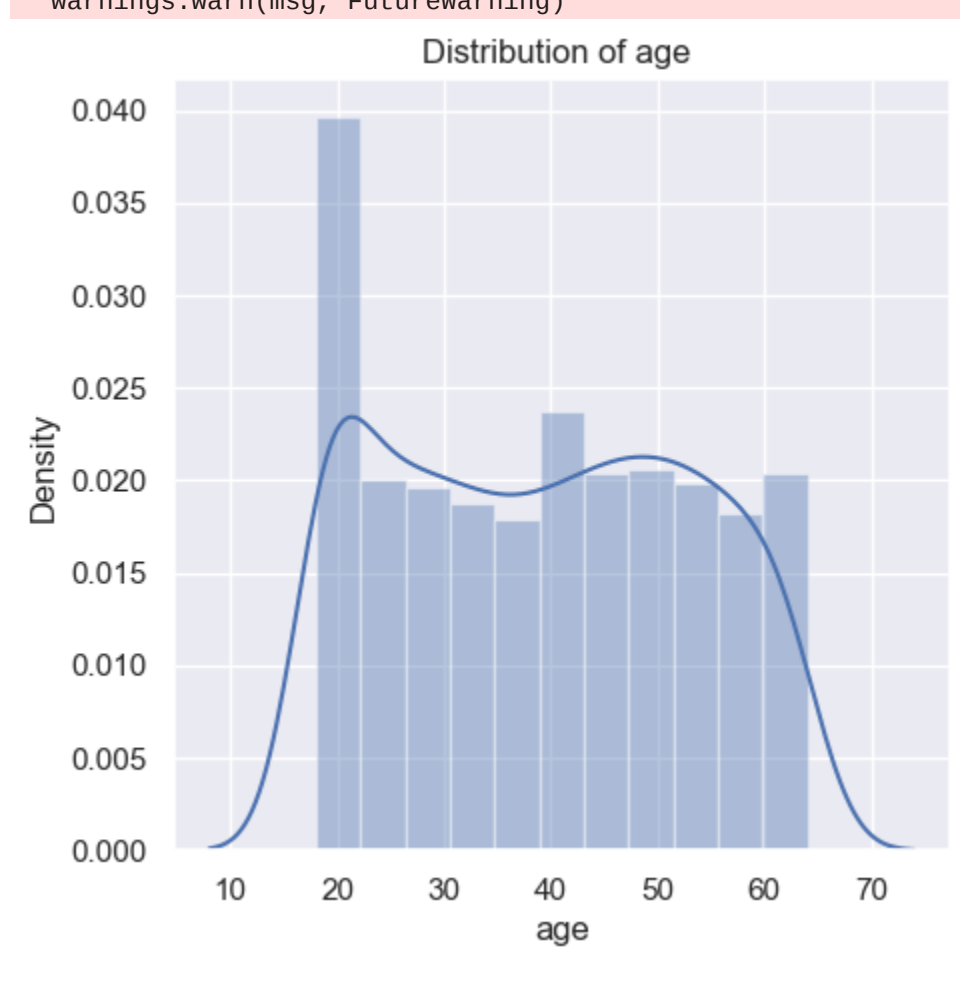
Out[11]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	1.515695	13270.422265	0.585202
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237	0.492871
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900	0.000000
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150	0.000000
50%	39.000000	1.000000	30.400000	1.000000	0.000000	2.000000	9382.033000	1.000000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515	1.000000
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010	1.000000

In [ ]:
```
# above values for only numerical value not for categorical features
# 25%, 50% , 75% are percentiles
```

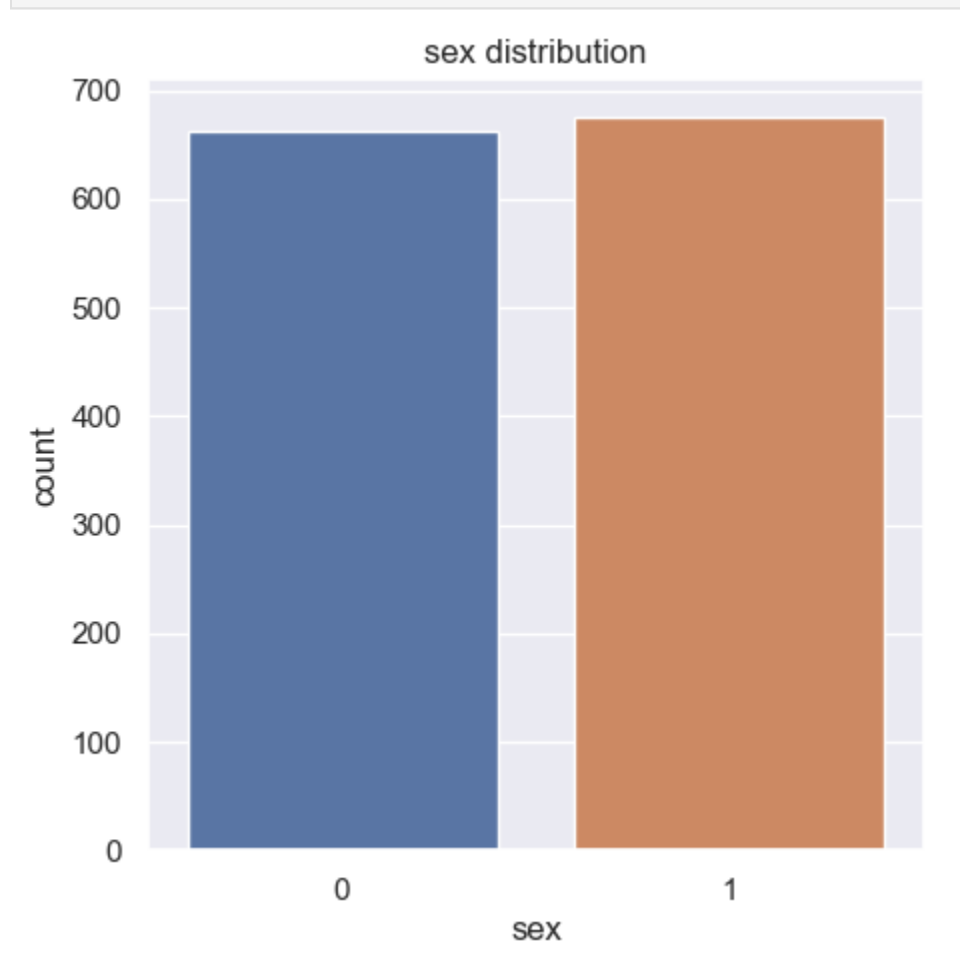
In [17]:
```
# distribution of age value -

sns.set()
plt.figure(figsize=(5,5))
sns.distplot(data['age'])
plt.title('Distribution of age ')
plt.show()
```

```
C:\Users\Vaibhav\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [ ]:
```
# gender - male or female
# count - it gives no of males and no of females


plt.figure(figsize=(5,5))
sns.countplot(x='sex', data=data)
```

In [28]:
```
plt.title('sex distribution')
plt.show()
```

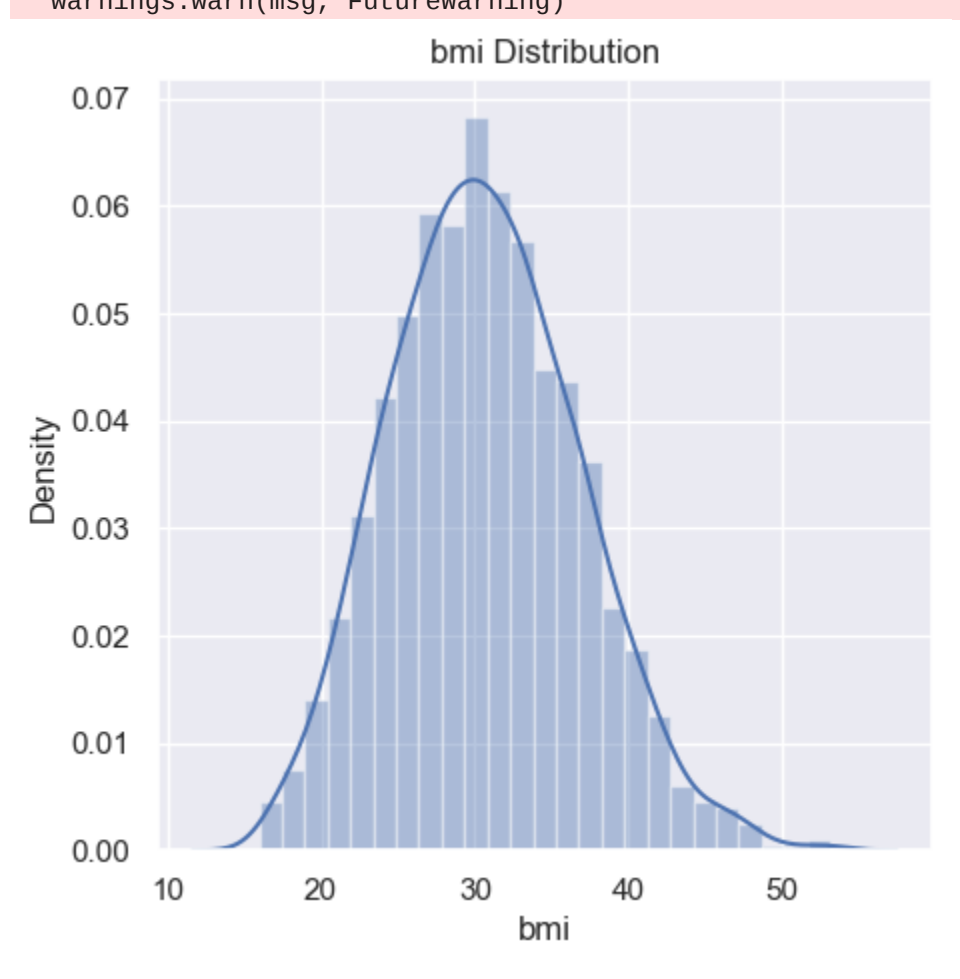


In [31]:
```
# here we can find count of male(1) and female(0)

data['sex'].value_counts()
```

Out[31]:
```
1    676
0    662
Name: sex, dtype: int64
```

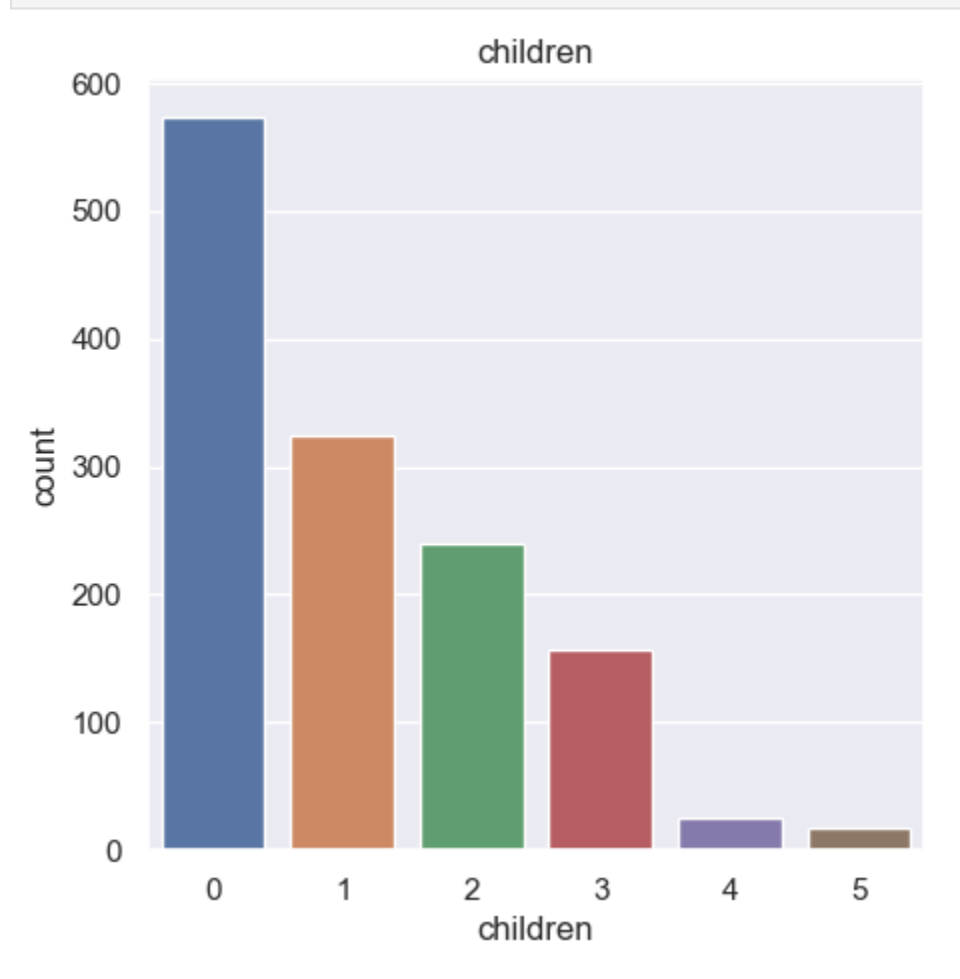
In [36]:
```
# bmi ( body mass index )= It gives person is over weight, under weight or normal weight

plt.figure(figsize=(5,5))
sns.distplot(data['bmi'])
plt.title('bmi Distribution')
plt.show()
```

```
C:\Users\Vaibhav\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [ ]:
```
# here we get normal distribution curve = its normal range is = approxi 18 to 25
# below 18 = we can say person is under weight
# above 25 = we can say person is over weight
# more no of values betweeen 25 to 40 hence we can say more no of people might be over weighted
```

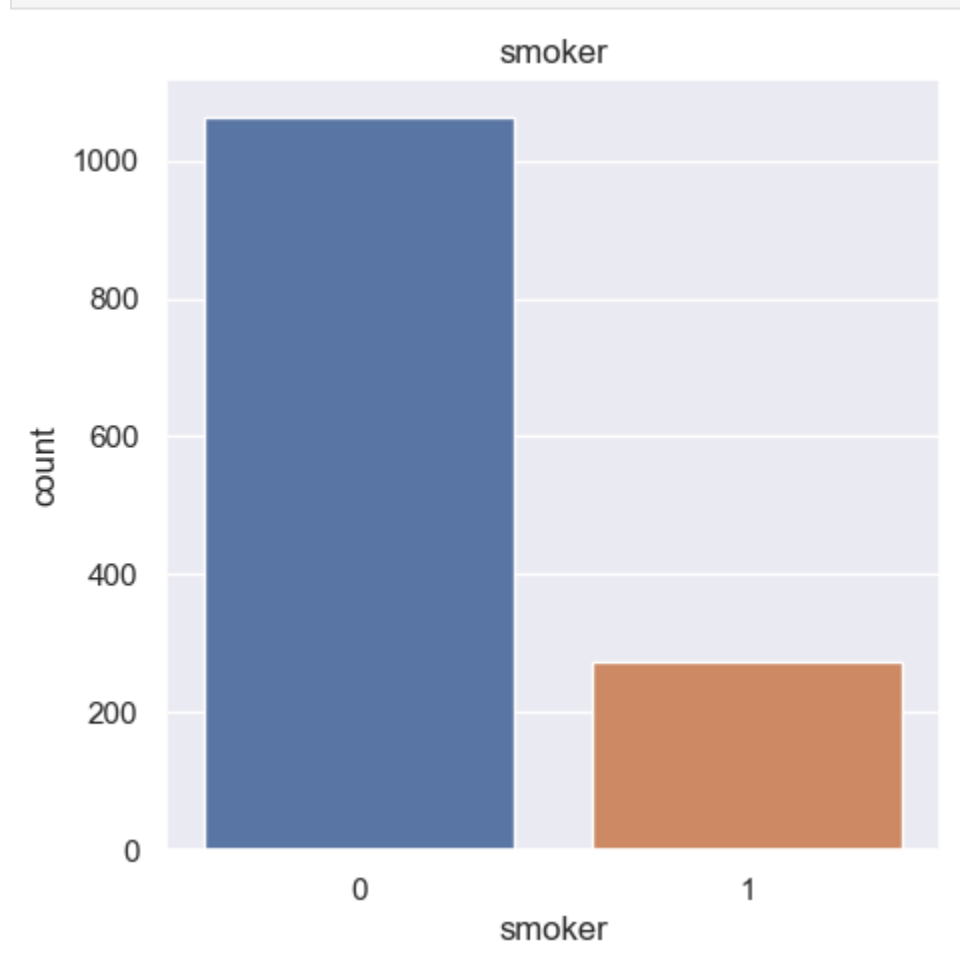
In [39]:
```
# children -

plt.figure(figsize=(5,5))
sns.countplot(x='children', data=data)
plt.title('children')
plt.show()
```



In [41]:
```
data['children'].value_counts()
```

Out[41]:
```
0    574
1    324
2    240
3    157
4     25
5     18
Name: children, dtype: int64
```

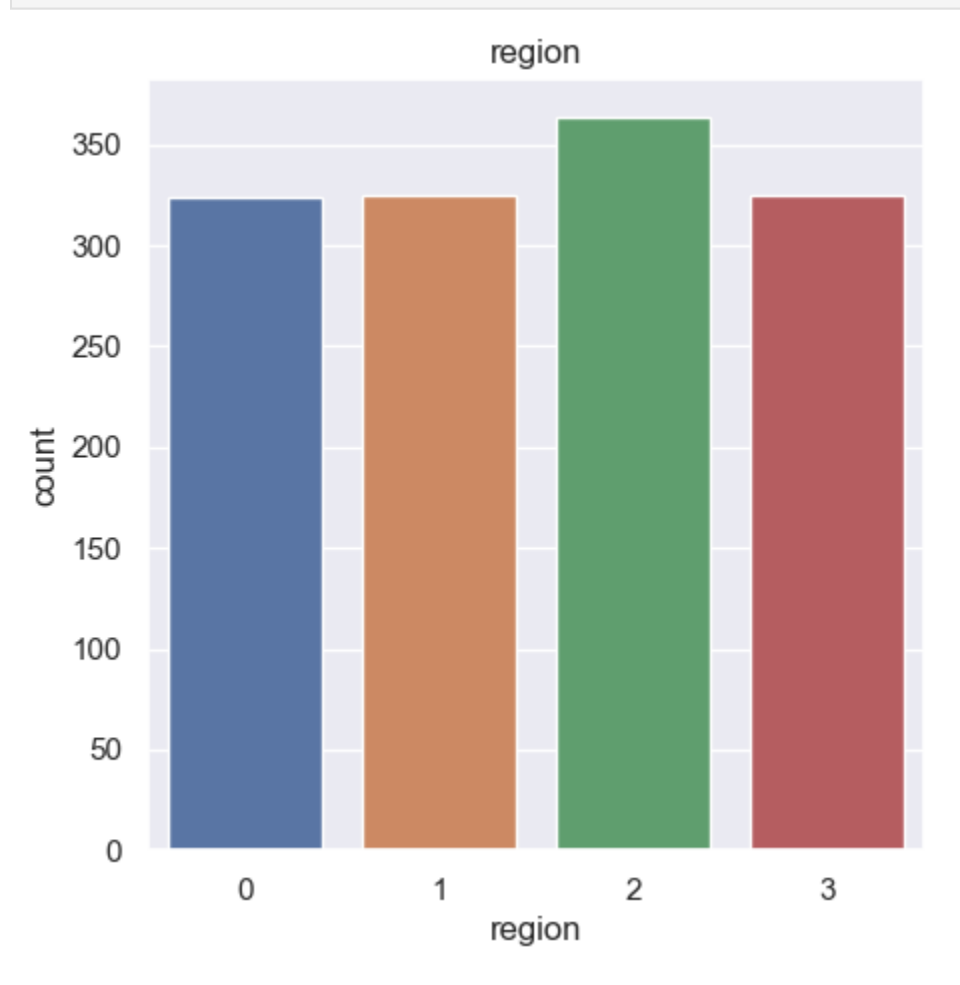
In [49]:
```
# smoker -

plt.figure(figsize=(5,5))
sns.countplot(x='smoker', data=data)
plt.title('smoker')
plt.show()
```



In [50]:
```
# count of smoker [if yes(1)= smoker and no(0)= non-smoker]

data['smoker'].value_counts()
```

Out[50]:
```
0    1064
1     274
Name: smoker, dtype: int64
```

In [51]:
```
# region

plt.figure(figsize=(5,5))
sns.countplot(x='region', data=data)
plt.title('region')
plt.show()
```

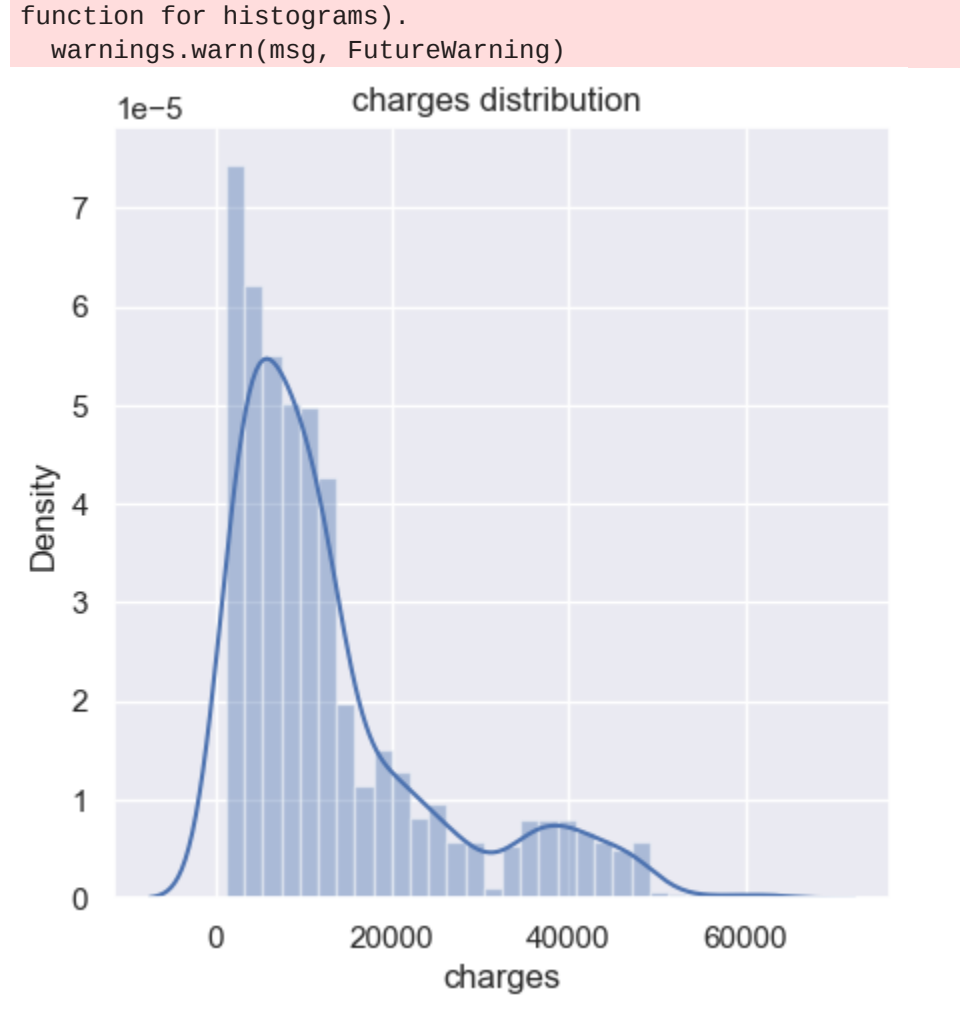


In [52]:
```
# northeast(0), northwest(1), southeast(2), southwest(3)
data['region'].value_counts()
```

Out[52]:
```
2    364
3    325
1    325
0    324
Name: region, dtype: int64
```

In [15]:
```
# charges -
sns.set()
plt.figure(figsize=(5,5))
sns.distplot(data['charges'])
plt.title('charges distribution ')
plt.show()
```

```
C:\Users\Vaibhav\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



spliting features and target variable

In [4]:
```
# here remove target variable ( insuranceclaim)

x = data.drop(columns='insuranceclaim',axis=1)          # take axis = 1 to delete column
y = data['insuranceclaim']
```

In [5]:
```
print(y)                                    # shows only target varible column
```

```
0       1
1       1
2       0
3       0
4       1
       ..
1333    0
1334    1
1335    1
1336    0
1337    1
Name: insuranceclaim, Length: 1338, dtype: int64
```

In [29]:
```
print(x)                                # insuranceclaim column is deleted
```

```
      age  sex     bmi  children  smoker  region      charges
0      19    0  27.900         0       1       3  16884.92400
1      18    1  33.770         1       0       2   1725.55230
2      28    1  33.000         3       0       2   4449.46200
3      33    1  22.705         0       0       1  21984.47061
4      32    1  28.880         0       0       1   3866.85520
...   ...  ...     ...       ...     ...     ...          ...
1333   50    1  30.970         3       0       1  10600.54830
1334   18    0  31.920         0       0       0   2205.98080
1335   18    0  36.850         0       0       2   1629.83350
1336   21    0  25.800         0       0       3   2007.94500
1337   61    0  29.070         0       1       1  29141.36030

[1338 rows x 7 columns]
```

Spliting the data into Traing and Testing data

In [14]:
```
x_train, x_test, y_train, y_test =train_test_split(x,y, test_size=0.2, random_state=2)
```

In [15]:
```
print(x.shape, x_train.shape, x_test.shape)
```

```
(1338, 7) (1070, 7) (268, 7)
```

Model training

In [12]:
```
# classification by using Decision tree

classifier = DecisionTreeClassifier()
```

In [16]:
```
classifier.fit(x_train, y_train)
```

Out[16]:
```
DecisionTreeClassifier()
```

Model Evaluation

In [ ]:
```
# prediction of Training data
```

In [17]:
```
training_data_prediction=classifier.predict(x_train)
```

In [21]:
```
# accuracy score
acc=accuracy_score(y_train, training_data_prediction)
print("training" ,acc)
```

```
training 1.0
```

In [ ]:
```
# prediction of Testing data
```

In [23]:
```
testing_data_prediction=classifier.predict(x_test)
```

In [24]:
```
# accuracy score
acc=accuracy_score(y_test, testing_data_prediction)
print("testing" ,acc)
```

```
testing 0.9589552238805971
```

In [30]:
```
# Building a predictive system
```

In [34]:
```
input_data=(28,1,33.000,3,0,2,4449.4620)
```

In [33]:
```
# changing input_data to numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction= classifier.predict(input_data_reshaped)
if prediction[0]==1:
    print("Insurance will be claimed")
else:
    print("Insurance will not be claimed")
```

```
Insurance will not be claimed
C:\Users\Vaibhav\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [35]:
```
input_data=(18,1,33.770,1,0,2,1725.55230)
```

In [37]:
```
input_data_as_numpy_array= np.asarray(input_data)

# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction= classifier.predict(input_data_reshaped)
if prediction[0]==1:
    print("Insurance will be claimed")
else:
    print("Insurance will not be claimed")
```

```
Insurance will be claimed
C:\Users\Vaibhav\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [ ]:

conclusion

In [ ]:
```
# if my prediction value is equal to zero then insurance will be claimed and
# if my prediction is equal to one then it will not be claimed
```