In [ ]:

```
Problem Statement -  To analyze the Appolo dataset and to find out the variables that are
                   significant in predicting the reason for hospitalization charges in dif

Insights:
1. Age and hospitalalization charges are highly correlated 0.48 (Pearson)
2. Age and hospitalalization charges are highly correlated 0.53 (Spearman)
3. Age 18 and 19 has count >65
4.  Min and Max for age range for female and male is the same
5. There are more male smokers than female
6.  Severity level 0,1,2 are have highest counts
7. In all the regions severity level 0,1,2,3 are the highest
8.  Southeast region has the highest number of count
9.  Viral load in southeast asia is the highest. Median hospital bills for a non smoker is
northwest has the very less number of severity 5
10. Hospitalization charges are highest for 0-25000 and 25000-50000
11.  Viral loads are highest for 10-12, 8-10, 12-14
12.  All the age groups have equal number of counts
13.  Age group 10-20 has the least number of count
14.  Severity level of 0 is highest for 10-20, 20-30 and 50-60 age groups
15.  Hospital charges Median for southeast is the highest
16. viral load Median for southeast is the highest an lowest for northeast and northwest
17. viral load Difference between Median for southeast between male and female is higher
18.  All smokers have greater bills than non smokers
19.  Viral load median for regions remains the almost the same for smoker and a non smoker
 ALL TESTS, P VALUE, ALPHA VALUE, ASSUMPTIONS, CONCLUTION STATEMENTS ARE WRITTEN IN THE NOT
20. The sample means of hospitalization charges of Smoking person is > than Non-Smoking
21. Ststistical evidence shows that the viral load of femals is same as that of male
22. Propotion of msoking and non smoking is independent of region
23. The mean viral load of women with 0 Severity level , 1 Severity level, and 2 Severity l


Recommendations :

1. Increase the health insurance premium amount for those who have the habit of smoking
2. Treat the covid +ve person when the severity level is low
3. Incorroporate covid guidelines mainly is southest region
4. Viral load, severity level and smoker affect the hospital charges the most
5. Let older age group remain indoors
6. Have more number of beds in regions which have more cases (Southest)
7. Females between age group 50-60 have more count whencompared with males hence immediatel
8. Have good stock of medicines in all hospitals
```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from  scipy import stats
from statsmodels.stats.weightstats import ztest as ztest
from statsmodels.formula.api import ols
import statsmodels.api as sm
import pingouin as pg
import datetime
import math
import scipy
import warnings
warnings.filterwarnings('ignore')
import re
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
```

In [2]:

```
df=pd.read_csv('scaler_apollo_hospitals.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0	age	sex	smoker	region	viral load	severity level	hospitalization charges
0	0	19	female	yes	southwest	9.30	0	42212
1	1	18	male	no	southeast	11.26	1	4314
2	2	28	male	no	southeast	11.00	3	11124
3	3	33	male	no	northwest	7.57	0	54961
4	4	32	male	no	northwest	9.63	0	9667

In [4]:

```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [5]:

```
df.info()
# Datatypes of all columns are shown below
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   age                     1338 non-null   int64  
 1   sex                     1338 non-null   object 
 2   smoker                  1338 non-null   object 
 3   region                  1338 non-null   object 
 4   viral load              1338 non-null   float64
 5   severity level          1338 non-null   int64  
 6   hospitalization charges  1338 non-null   int64  
dtypes: float64(1), int64(3), object(3)
memory usage: 73.3+ KB
```

In [6]:

```
df.shape
# Shape of data
```

Out[6]:

```
(1338, 7)
```

In [7]:

```
df.isnull().sum()
# There is no null values
```

Out[7]:

```
age                       0
sex                       0
smoker                    0
region                    0
viral load                0
severity level            0
hospitalization charges    0
dtype: int64
```

In [8]:

```
df.describe(include='object')
# Descriptive summary inclusive of object
```

Out[8]:

	sex	smoker	region
count	1338	1338	1338
unique	2	2	4
top	male	no	southeast
freq	676	1064	364

In [9]:

```
df.describe()
# Descriptive summary for numerical features
```

Out[9]:

	age	viral load	severity level	hospitalization charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	10.221233	1.094918	33176.058296
std	14.049960	2.032796	1.205493	30275.029296
min	18.000000	5.320000	0.000000	2805.000000
25%	27.000000	8.762500	0.000000	11851.000000
50%	39.000000	10.130000	1.000000	23455.000000
75%	51.000000	11.567500	2.000000	41599.500000
max	64.000000	17.710000	5.000000	159426.000000

In [10]:

```
df.columns
```

Out[10]:

```
Index(['age', 'sex', 'smoker', 'region', 'viral load', 'severity level',
       'hospitalization charges'],
      dtype='object')
```

In [11]:

```
df
```

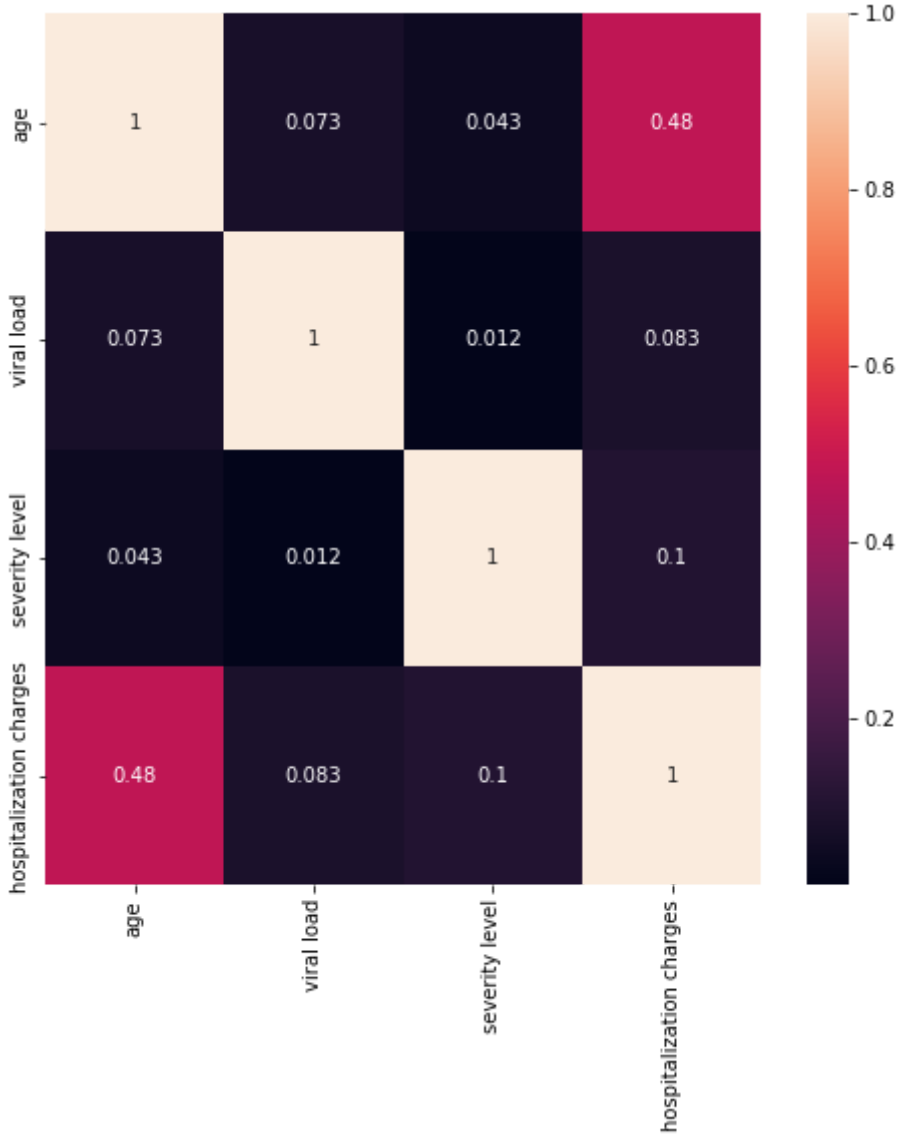
Out[11]:

	age	sex	smoker	region	viral load	severity level	hospitalization charges
0	19	female	yes	southwest	9.30	0	42212
1	18	male	no	southeast	11.26	1	4314
2	28	male	no	southeast	11.00	3	11124
3	33	male	no	northwest	7.57	0	54961
4	32	male	no	northwest	9.63	0	9667
...	...	...	...	...	...	...	...
1333	50	male	no	northwest	10.32	3	26501
1334	18	female	no	northeast	10.64	0	5515
1335	18	female	no	southeast	12.28	0	4075
1336	21	female	no	southwest	8.60	0	5020
1337	61	female	yes	northwest	9.69	0	72853

1338 rows × 7 columns

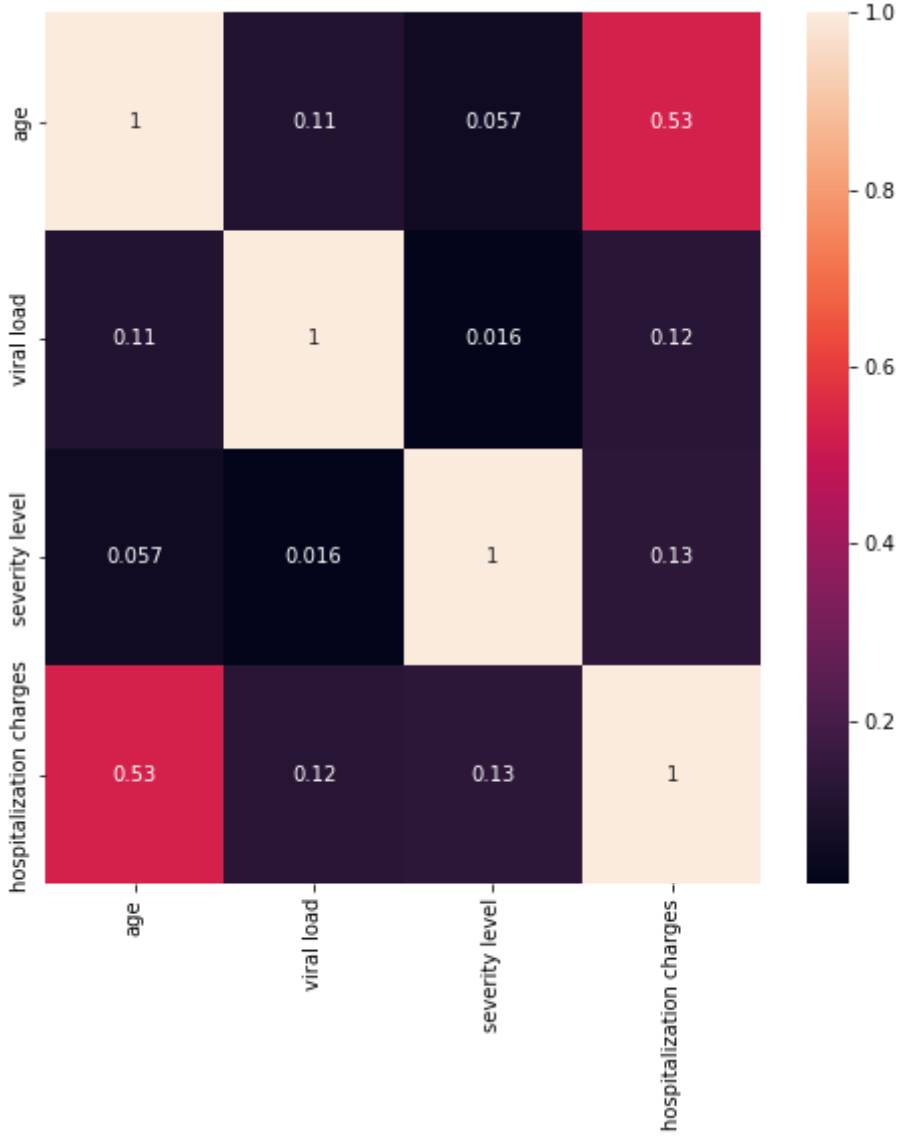
In [12]:

```
fig,ax=plt.subplots(figsize=(8,8))
sns.heatmap(df.corr(method ='kendall'),annot=True
            ,ax=ax)
plt.show()

# Kendall Correlation
# Age and hospitalalization charges are highly correlated 0.48
```



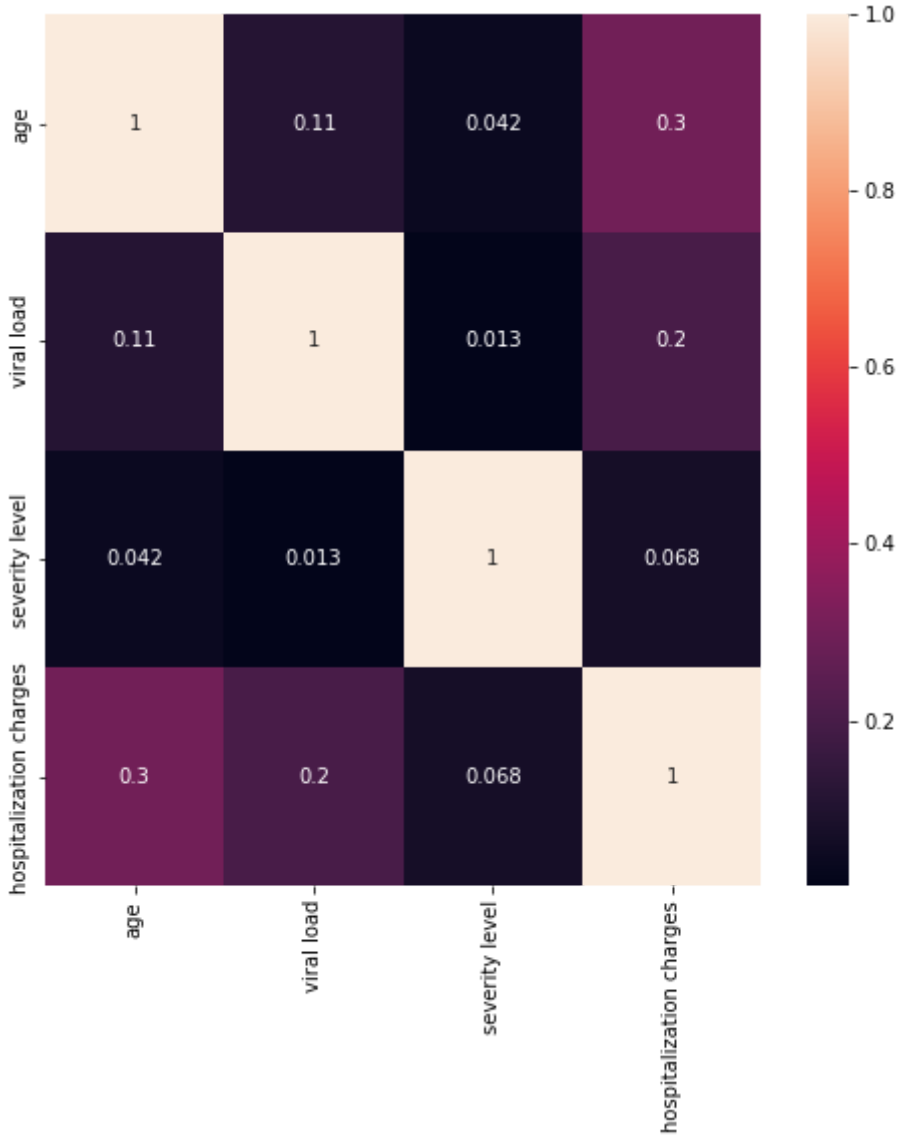
In [13]:

```
fig,ax=plt.subplots(figsize=(8,8))
sns.heatmap(df.corr(method ='spearman'),annot=True
            ,ax=ax)
plt.show()
# Spearman Correlation
# Age and hospitalalization charges are highly correlated 0.53
```



In [14]:

```
fig,ax=plt.subplots(figsize=(8,8))
sns.heatmap(df.corr(method ='pearson'),annot=True
            ,ax=ax)
plt.show()
# Pearson Correlation
# Age and hospitalalization charges are highly correlated 0.3
```



In [15]:

```
def outliners(x,col):
    Q1 = np.percentile(x[col], 25)
    Q3 = np.percentile(x[col], 75)
    IQR = Q3 - Q1
    upper = Q3 +1.5*IQR
    lower = Q1 - 1.5*IQR
    #print(upper,lower)
    ls=list(x.iloc[((x[col]<lower) | (x[col]>upper)).values].index)
    return ls
```

In [16]:

```
outliners(df,'viral load')
```

Out[16]:

```
[116, 286, 401, 543, 847, 860, 1047, 1088, 1317]
```

In [17]:

```
len(outliners(df,'viral load'))*100/len(df)
# 0.67 % values are outliners in 'viral load' feature
```

Out[17]:

```
0.672645739910314
```

In [18]:

```
df.drop(outliners(df,'viral load'),axis=0,inplace=True)
# Droped all the outliners
```

In [19]:

```
len(outliners(df,'age'))*100/len(df)
```

Out[19]:

```
0.0
```

In [20]:

```
len(outliners(df,'hospitalization charges'))*100/len(df)
# 10.38 % are outliners in hospitalization charges
```

Out[20]:

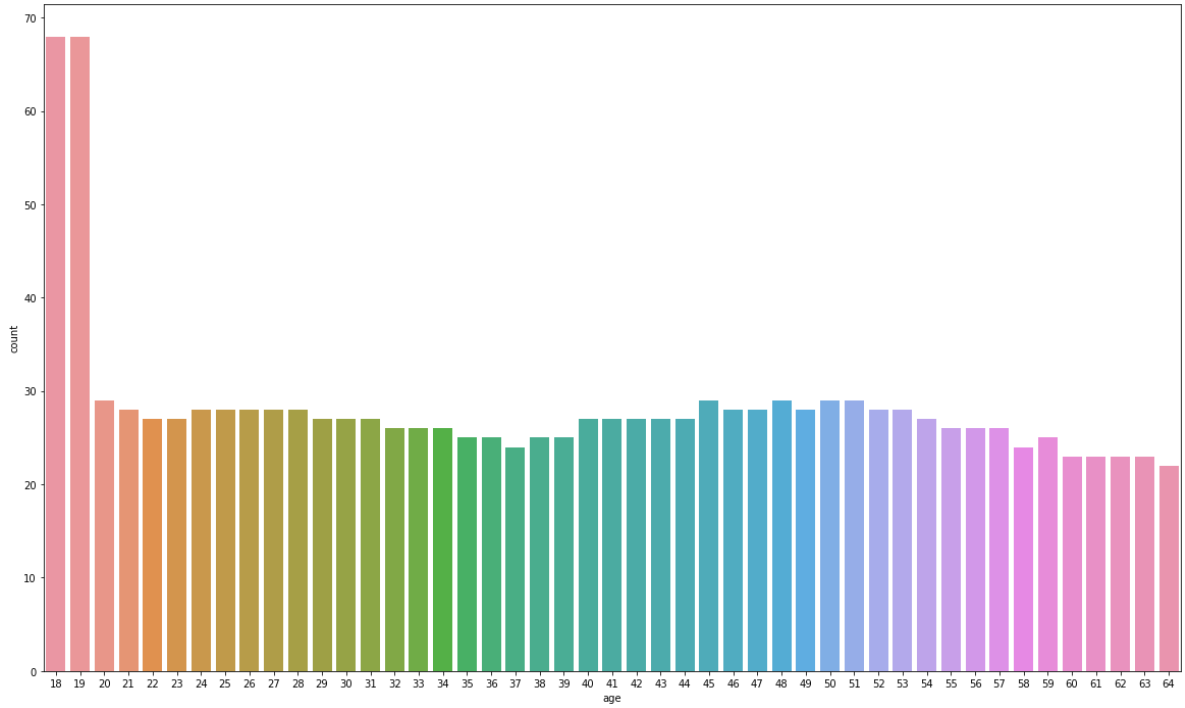
```
10.383747178329571
```

In [21]:

```
fig,ax=plt.subplots(figsize=(20,12))
sns.countplot(df['age'])
# Below is the count plot for age
# Age 18 and 19 has count >65
```

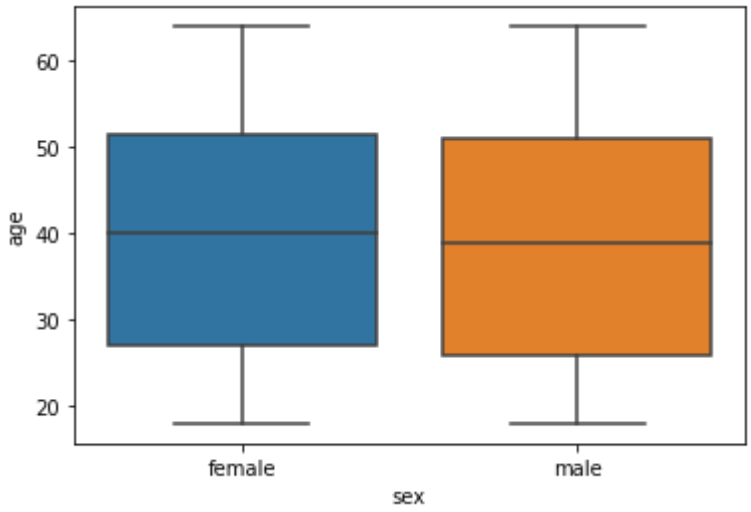
Out[21]:

```
<AxesSubplot:xlabel='age', ylabel='count'>
```



In [22]:

```
sns.boxplot(x=df['sex'],y=df['age'])
plt.show()
# Boxplot for sex and age
# Min and Max for age range for female and male is the same
```

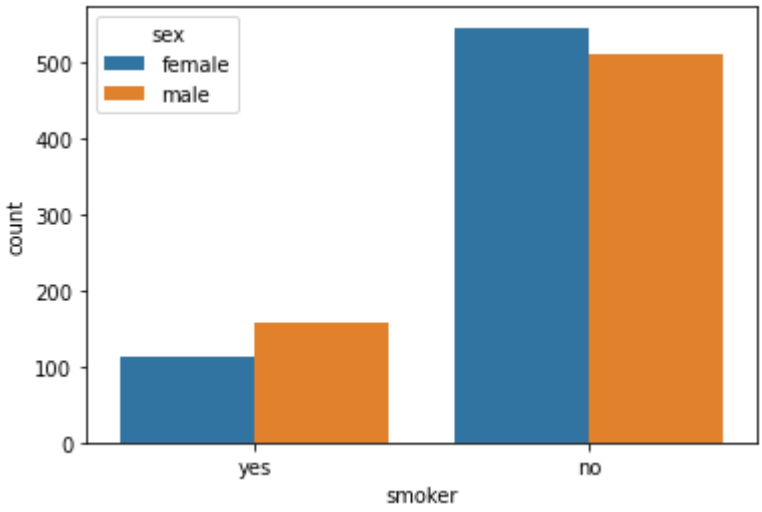


In [23]:

```
sns.countplot(df['smoker'],hue=df['sex'])
# Count plot for smoker with hue as sex
# There are more male smokers than female
```

Out[23]:

```
<AxesSubplot:xlabel='smoker', ylabel='count'>
```

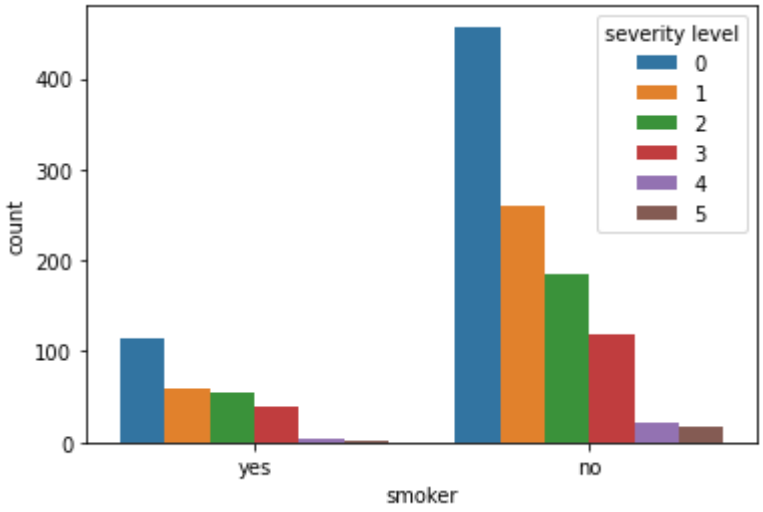


In [24]:

```
sns.countplot(df['smoker'],hue=df['severity level'])
# Count plot for smokers with severity level as hue
# Severity level 0,1,2 are have highest counts
```

Out[24]:

```
<AxesSubplot:xlabel='smoker', ylabel='count'>
```

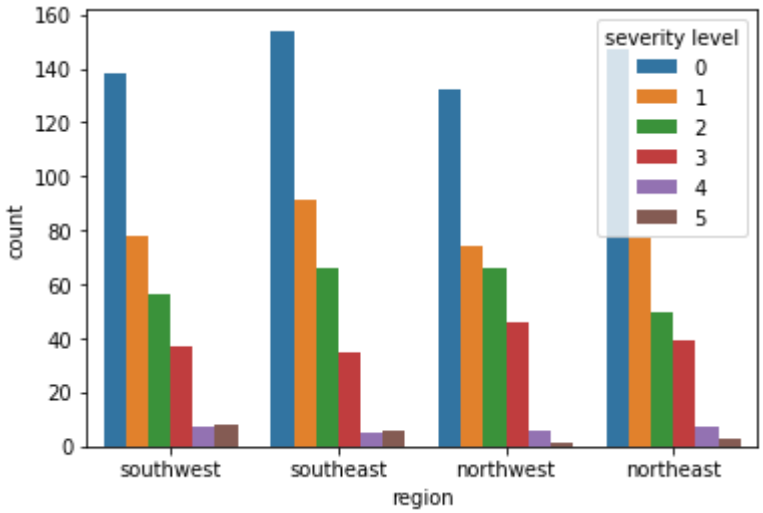


In [25]:

```
sns.countplot(df['region'],hue=df['severity level'])
# Count plot for region with hue as severity level
# In all the regions severity level 0,1,2,3 are the highest
```

Out[25]:

```
<AxesSubplot:xlabel='region', ylabel='count'>
```

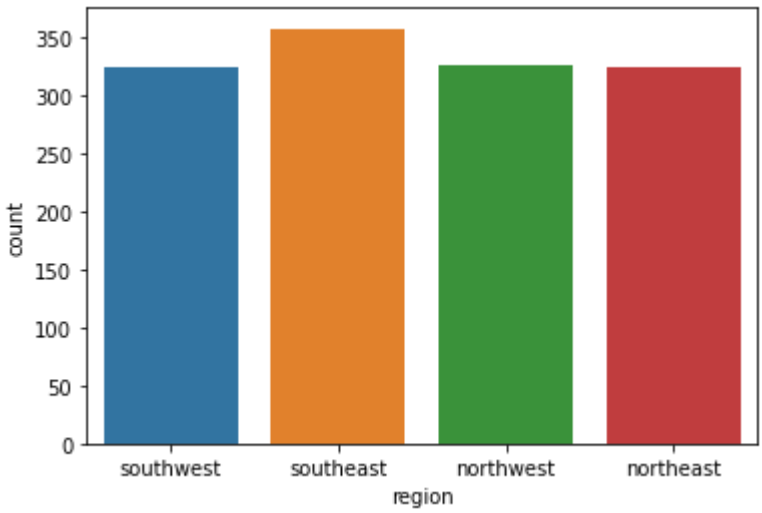


In [26]:

```
sns.countplot(df['region'])
# Southeast region has the highest number of count
```

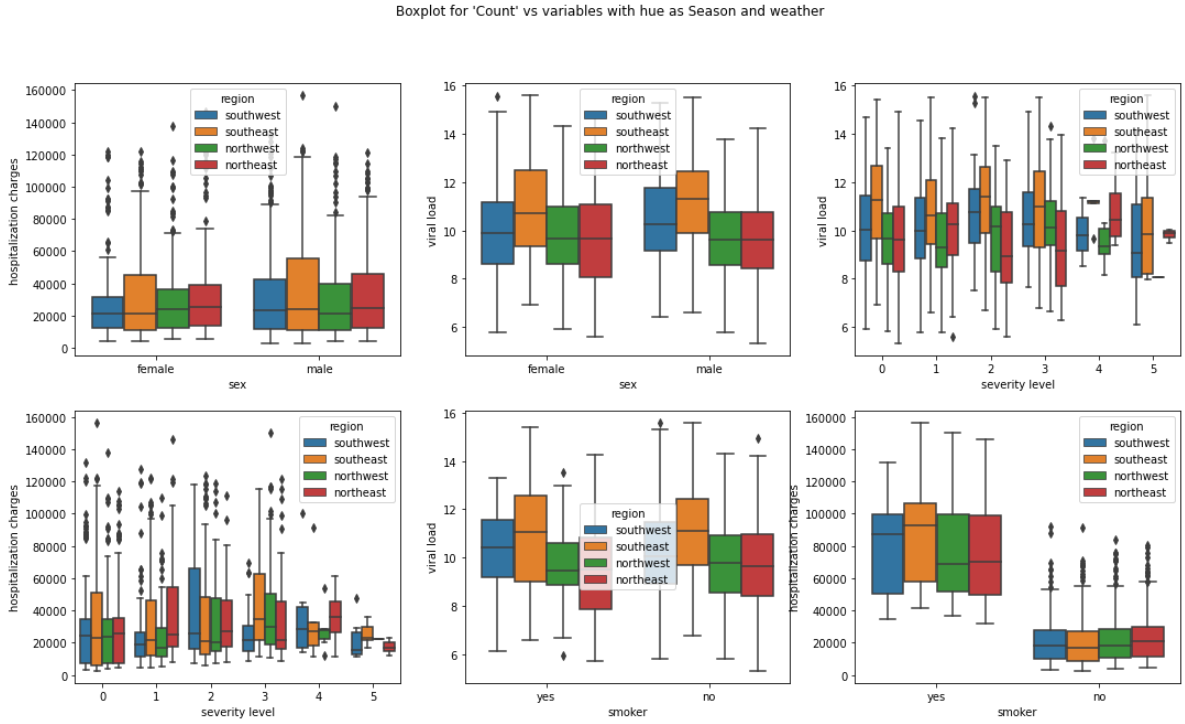
Out[26]:

```
<AxesSubplot:xlabel='region', ylabel='count'>
```



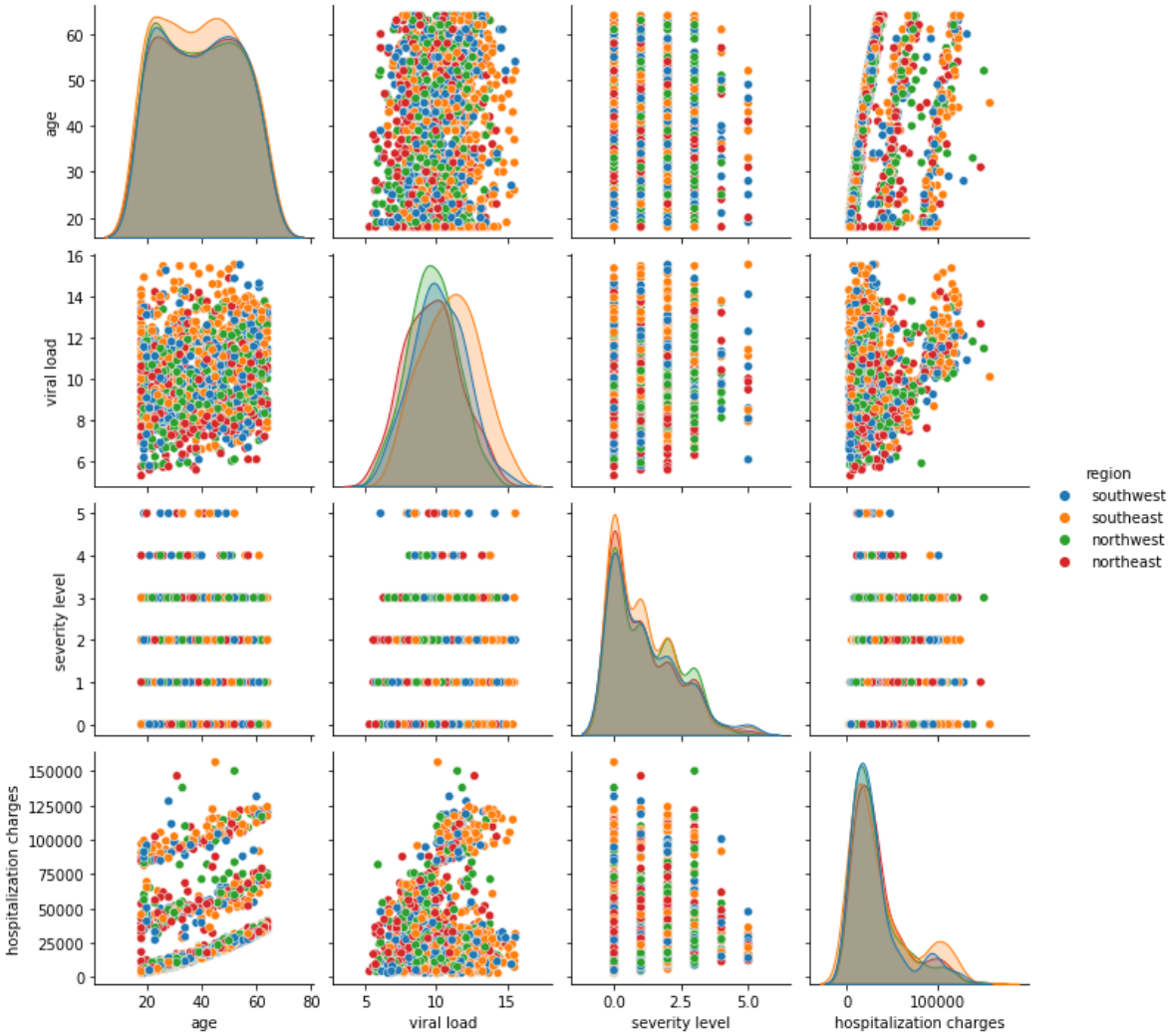
In [27]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('Boxplot for \'Count\' vs variables with hue as Season and weather')
sns.boxplot(ax=axes[0, 0], data=df, x='sex', y='hospitalization charges',hue='region' )
sns.boxplot(ax=axes[0, 1], data=df, x='sex', y='viral load',hue='region' )
sns.boxplot(ax=axes[0, 2], data=df, x='severity level', y='viral load',hue='region' )
sns.boxplot(ax=axes[1, 0], data=df, x='severity level', y='hospitalization charges',hue='re
sns.boxplot(ax=axes[1, 1], data=df, x='smoker', y='viral load',hue='region')
sns.boxplot(ax=axes[1, 2], data=df, x='smoker', y='hospitalization charges',hue='region')
plt.show()

# shows the boxplot for all numerical variables
# Viral load in southeast asia is the highest
# Median hospital bills for a non smoker is always lesser than a smoker
# northwest has the very less number of severity 5
```



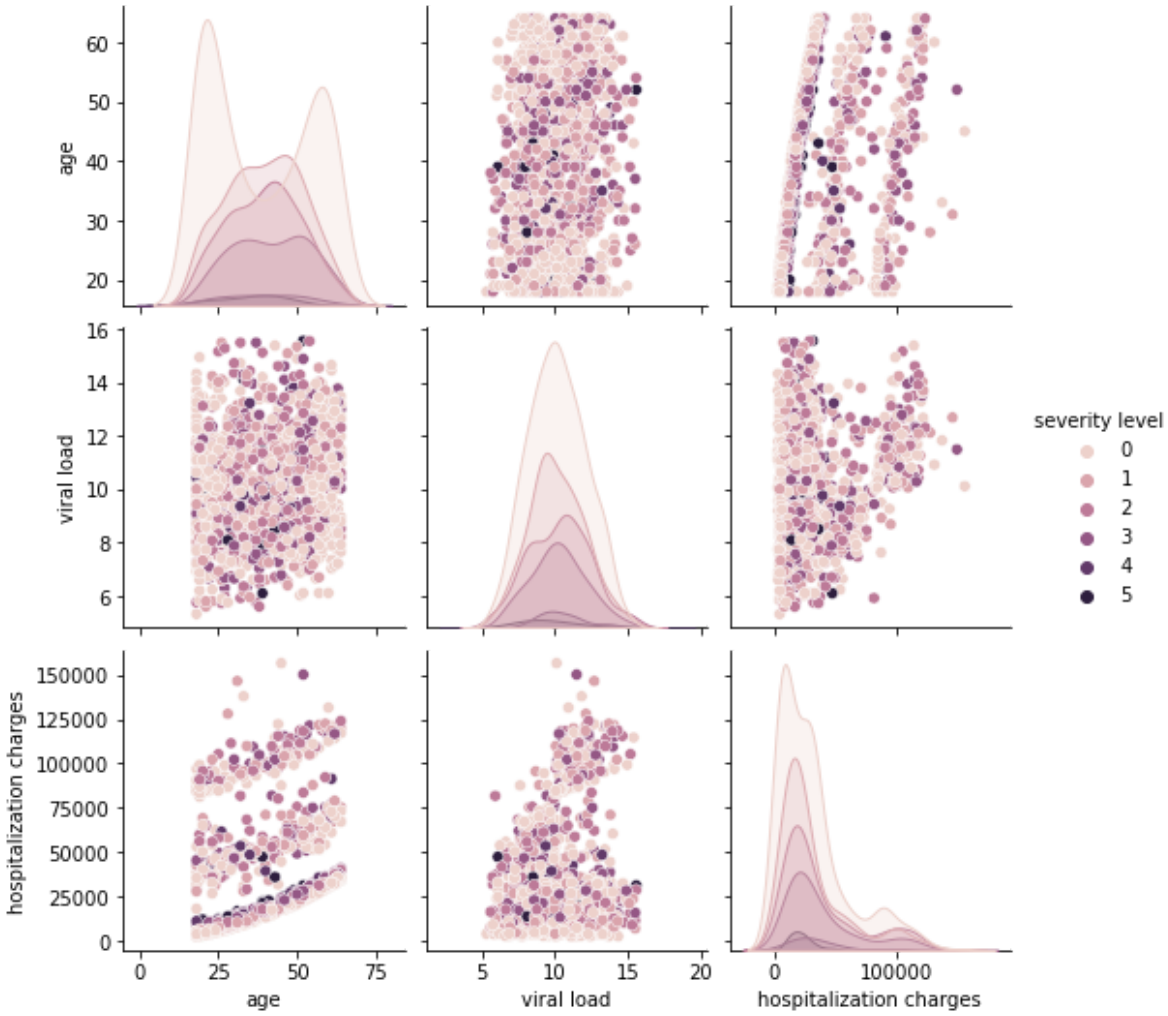
In [28]:

```
sns.pairplot(df, hue = "region")
plt.show()
# Below shows the pairplot for all the numerical variables with hue as region
```



In [29]:

```
sns.pairplot(df, hue = "severity level")
plt.show()
# Below shows the pairplot for all the numerical variables with hue as severity level
```

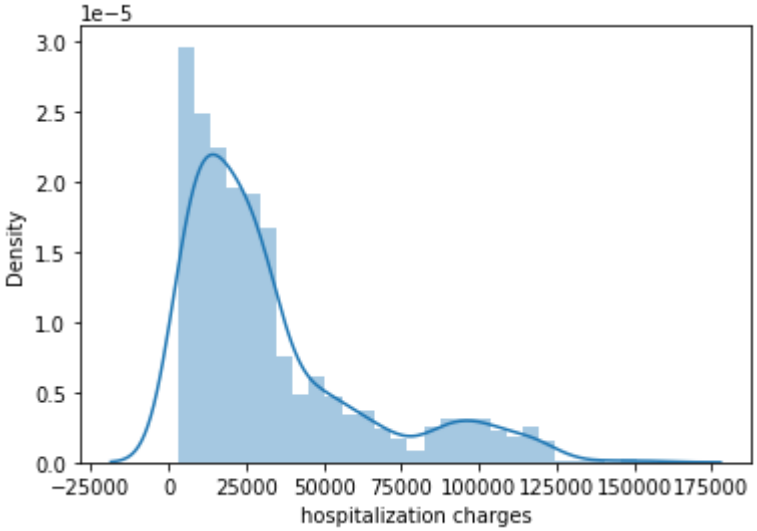


In [30]:

```
sns.distplot(df['hospitalization charges'])
# Distribution plot for 'hospitalization charges'
```

Out[30]:

```
<AxesSubplot:xlabel='hospitalization charges', ylabel='Density'>
```

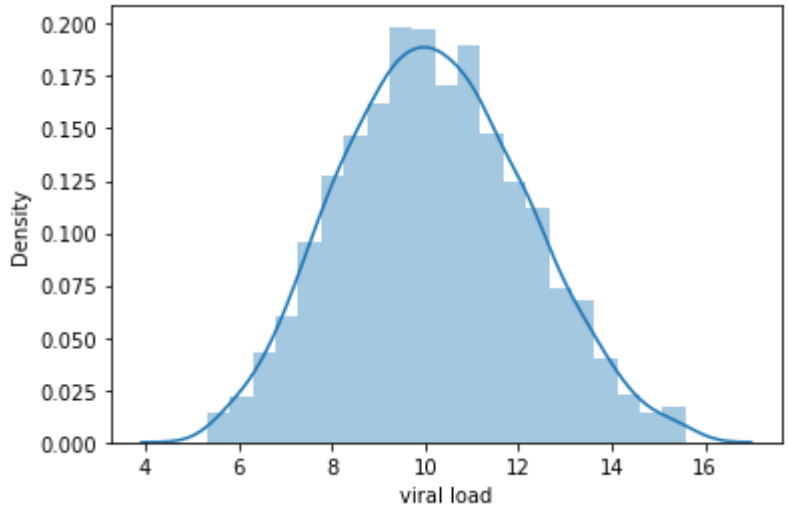


In [31]:

```
sns.distplot(df['viral load'])
# Distribution plot for 'viral load'
```

Out[31]:

```
<AxesSubplot:xlabel='viral load', ylabel='Density'>
```

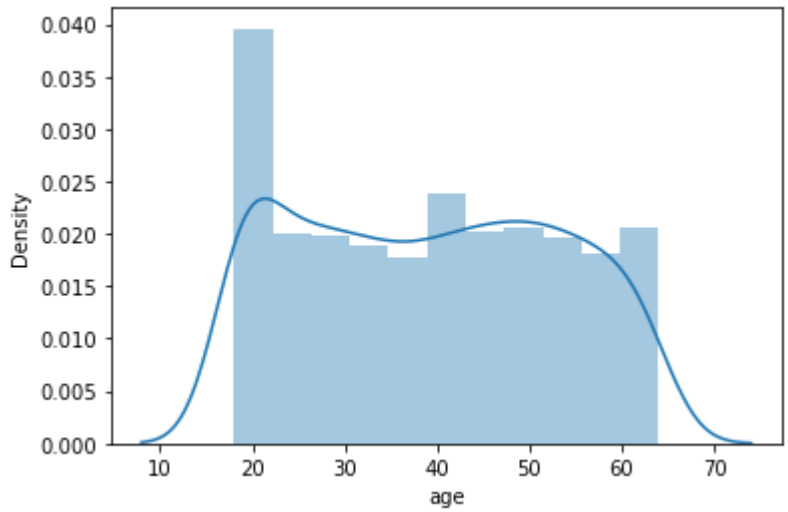


In [32]:

```
sns.distplot(df['age'])
# Distribution plot for'age'
```

Out[32]:

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



In [33]:

```
bins=[0,25000,50000,75000,100000,125000,150000,175000]
labels=['0-25000','25000-50000','50000-75000','75000-100000','100000-125000','125000-150000
df['hospitalization charges_bins']=pd.cut(df['hospitalization charges'], bins=bins, labels=
# Created bins for hospitalization charges
```

In [34]:

```
bins=[0,4,6,8,10,12,14,16,18]
labels=['0-4','4-6','6-8','8-10','10-12','12-14','14-16','16-18']
df['viral load_bins']=pd.cut(df['viral load'], bins=bins, labels=labels)
# Created bins for viral load
```

In [35]:

```
df['hospitalization charges_bins'].value_counts()
# Hospitalization charges are highest for 0-25000 and 25000-50000
```

Out[35]:

```
0-25000          707
25000-50000      352
50000-75000      111
75000-100000      83
100000-125000     70
125000-150000      4
150000-175000      2
Name: hospitalization charges_bins, dtype: int64
```

In [36]:

```
df['viral load_bins'].value_counts()
# Viral loads are highest for 10-12, 8-10, 12-14
```

Out[36]:

```
10-12    449
8-10     445
12-14    207
6-8      173
14-16     40
4-6       15
0-4        0
16-18      0
Name: viral load_bins, dtype: int64
```

In [37]:

```
bins=[0,10,20,30,40,50,60]
labels=['0-10','10-20','20-30','30-40','40-50','50-60']
df['age_bins']=pd.cut(df['age'], bins=bins, labels=labels)
```

In [38]:

```
df['age_bins'].value_counts()
# All the age groups have equal number of counts
```

Out[38]:

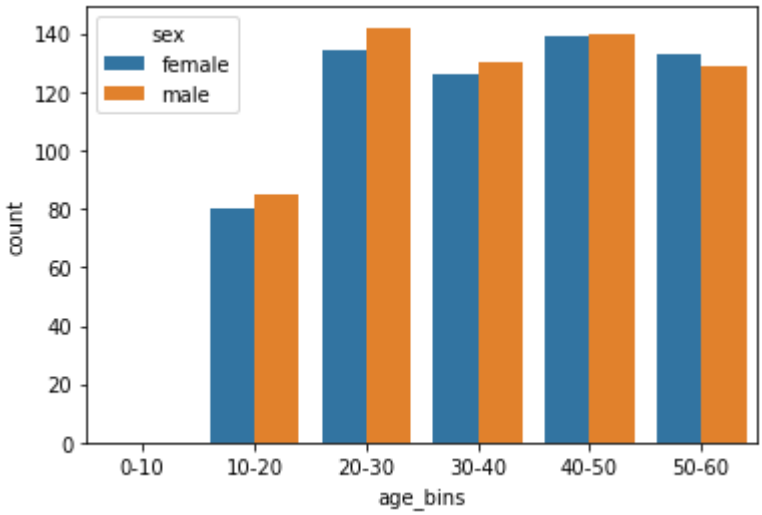
```
40-50    279
20-30    276
50-60    262
30-40    256
10-20    165
0-10       0
Name: age_bins, dtype: int64
```

In [39]:

```
sns.countplot(df['age_bins'],hue=df['sex'])
# Age group 10-20 has the least number of count
```

Out[39]:

```
<AxesSubplot:xlabel='age_bins', ylabel='count'>
```

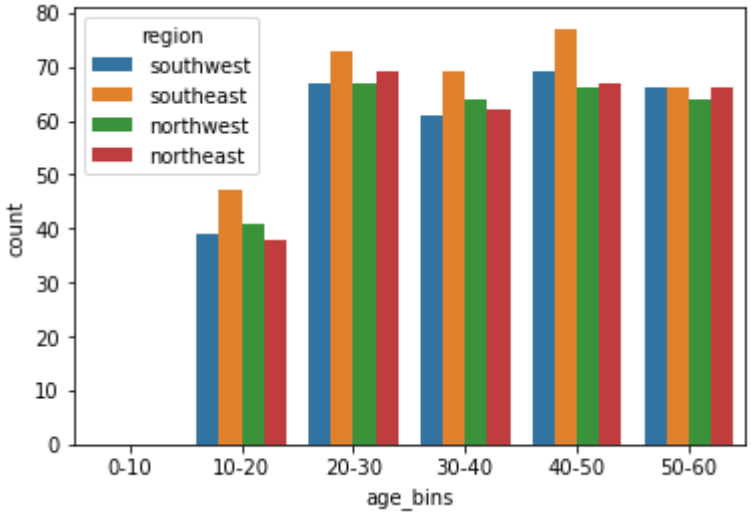


In [40]:

```
sns.countplot(df['age_bins'],hue=df['region'])
# Below is the age_bins counts with hue as region
```

Out[40]:

```
<AxesSubplot:xlabel='age_bins', ylabel='count'>
```

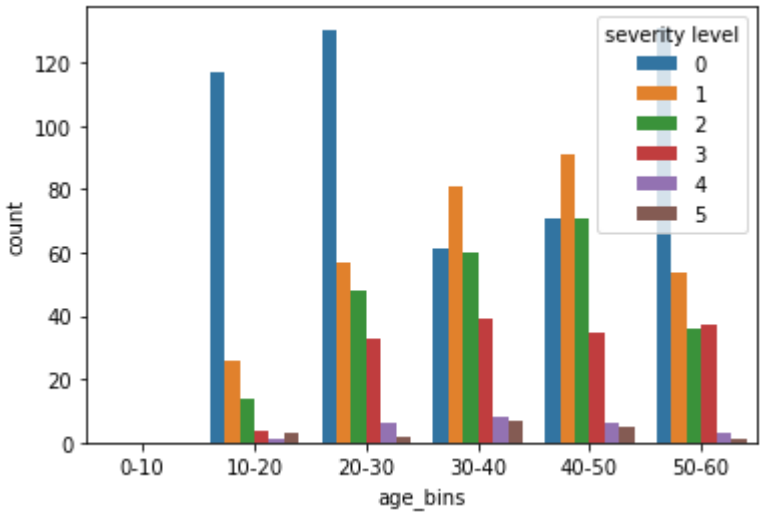


In [41]:

```
sns.countplot(df['age_bins'],hue=df['severity level'])
# Below is the count plot for age bins with hue as severity level
# Severity level of 0 is highest for 10-20, 20-30 and 50-60 age groups
```

Out[41]:

```
<AxesSubplot:xlabel='age_bins', ylabel='count'>
```

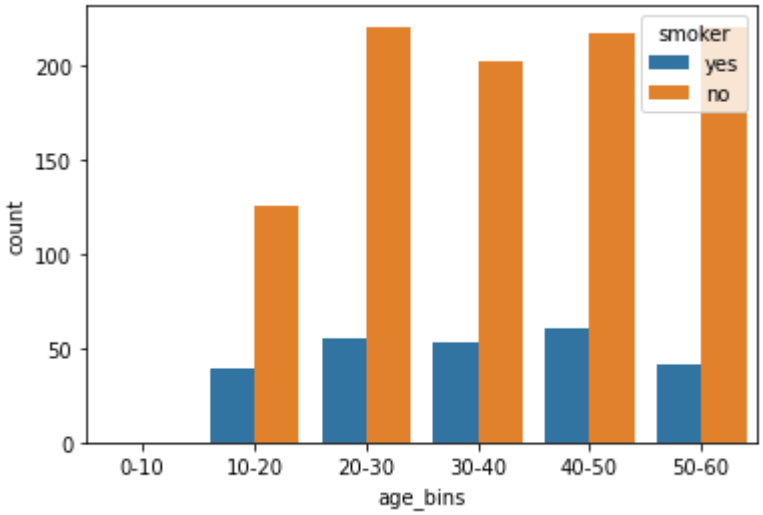


In [42]:

```
sns.countplot(df['age_bins'],hue=df['smoker'])
# Count of age groups with hue as smoker
```

Out[42]:

```
<AxesSubplot:xlabel='age_bins', ylabel='count'>
```

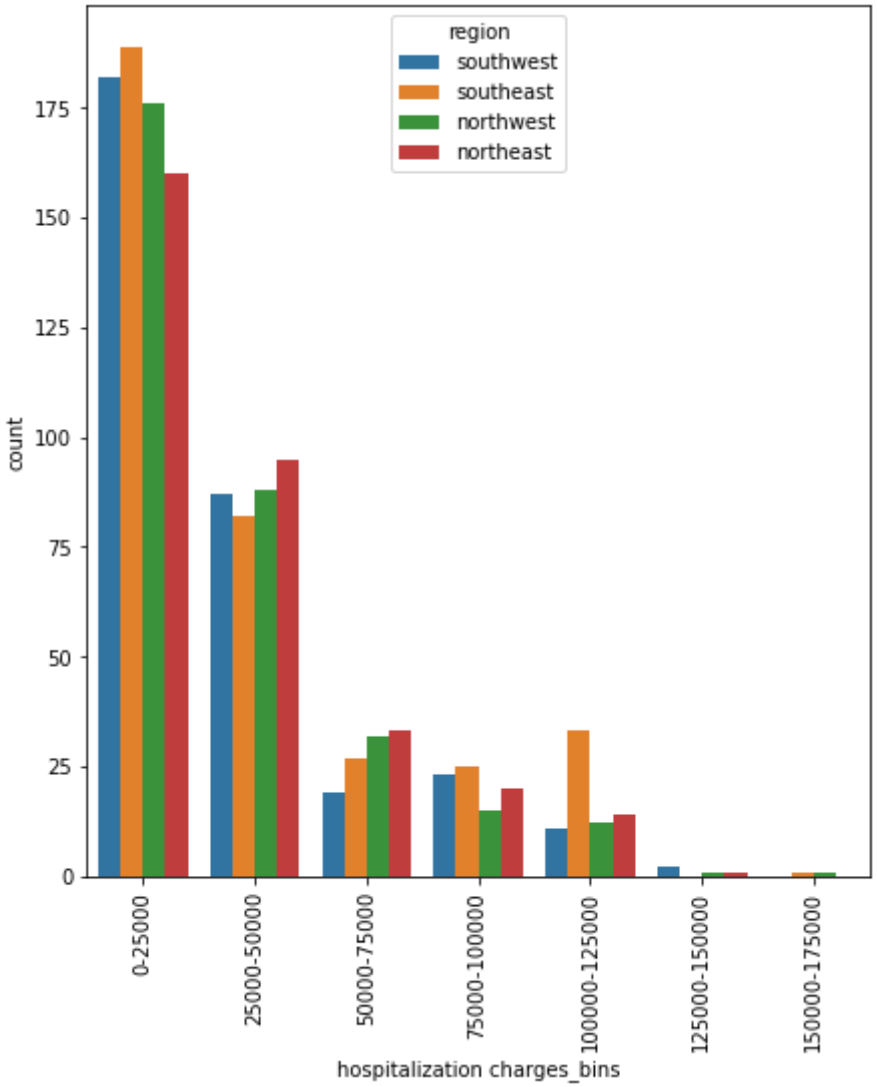


In [43]:

```
fig,ax=plt.subplots(figsize=(7,8))
plt.xticks(rotation=90)
sns.countplot(df['hospitalization charges_bins'],hue=df['region'])
# Count plot for hospital bils with hue as region
```

Out[43]:

```
<AxesSubplot:xlabel='hospitalization charges_bins', ylabel='count'>
```

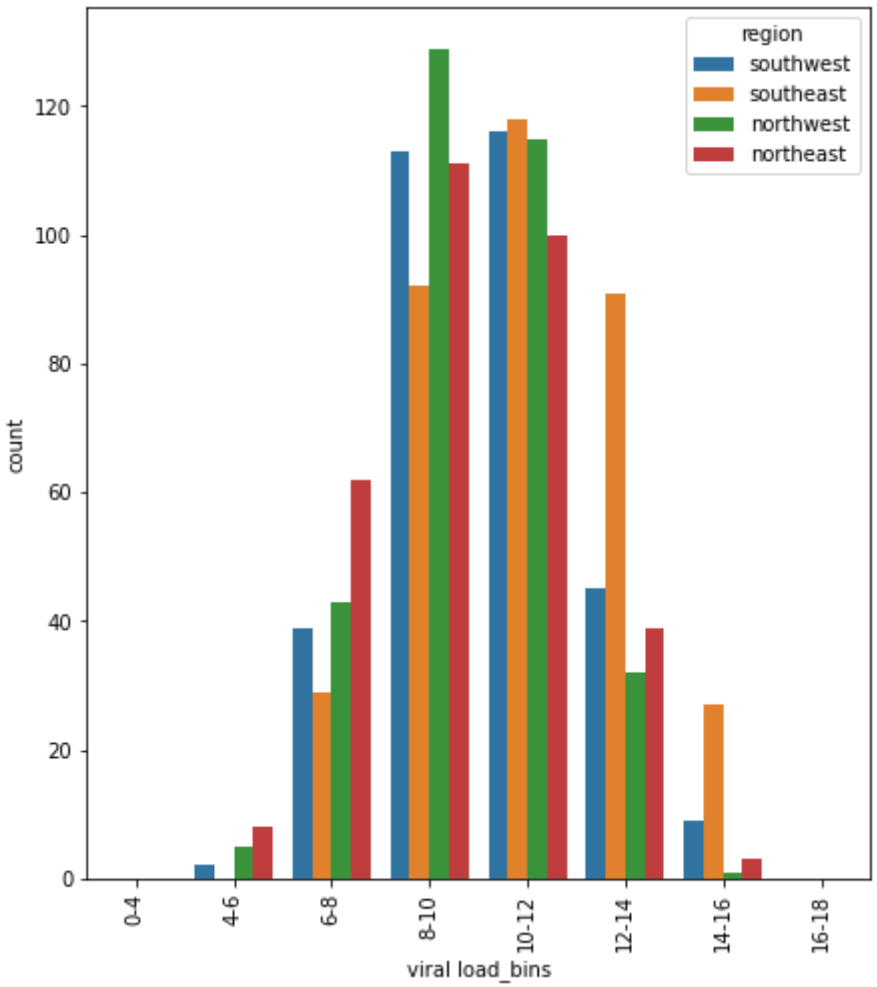


In [44]:

```
fig,ax=plt.subplots(figsize=(7,8))
plt.xticks(rotation=90)
sns.countplot(df['viral load_bins'],hue=df['region'])
# Count plot for viral load bins with hue as region
```

Out[44]:

```
<AxesSubplot:xlabel='viral load_bins', ylabel='count'>
```

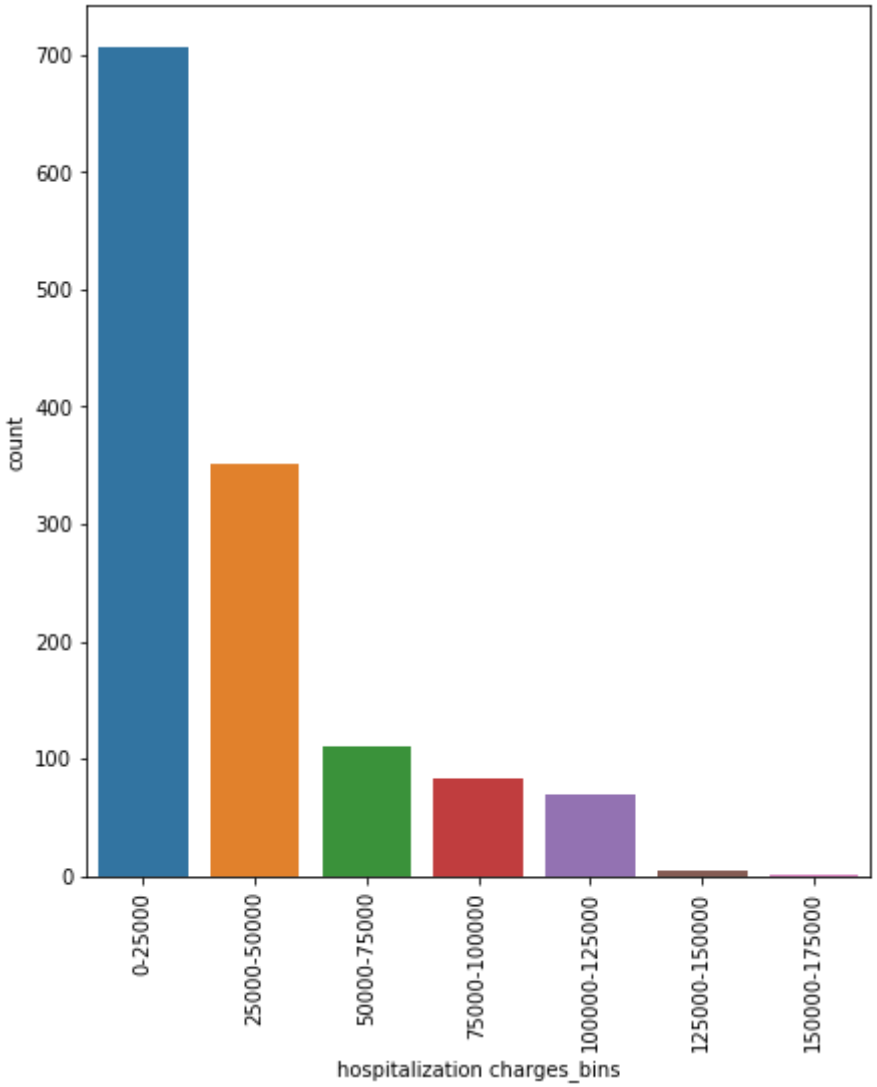


In [45]:

```
fig,ax=plt.subplots(figsize=(7,8))
plt.xticks(rotation=90)
sns.countplot(df['hospitalization charges_bins'])
# Count plot for hospitalization charges_bins
```

Out[45]:

```
<AxesSubplot:xlabel='hospitalization charges_bins', ylabel='count'>
```

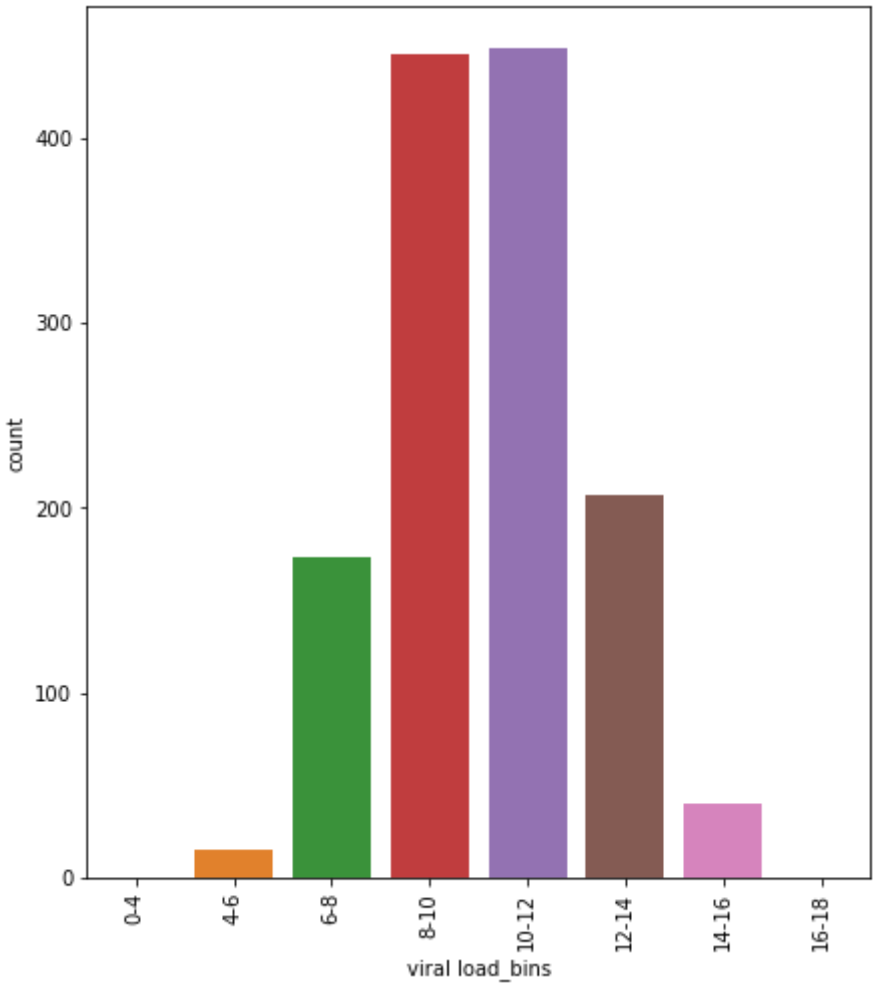


In [46]:

```
fig,ax=plt.subplots(figsize=(7,8))
plt.xticks(rotation=90)
sns.countplot(df['viral load_bins'])
# Count plot for viral load bins
```

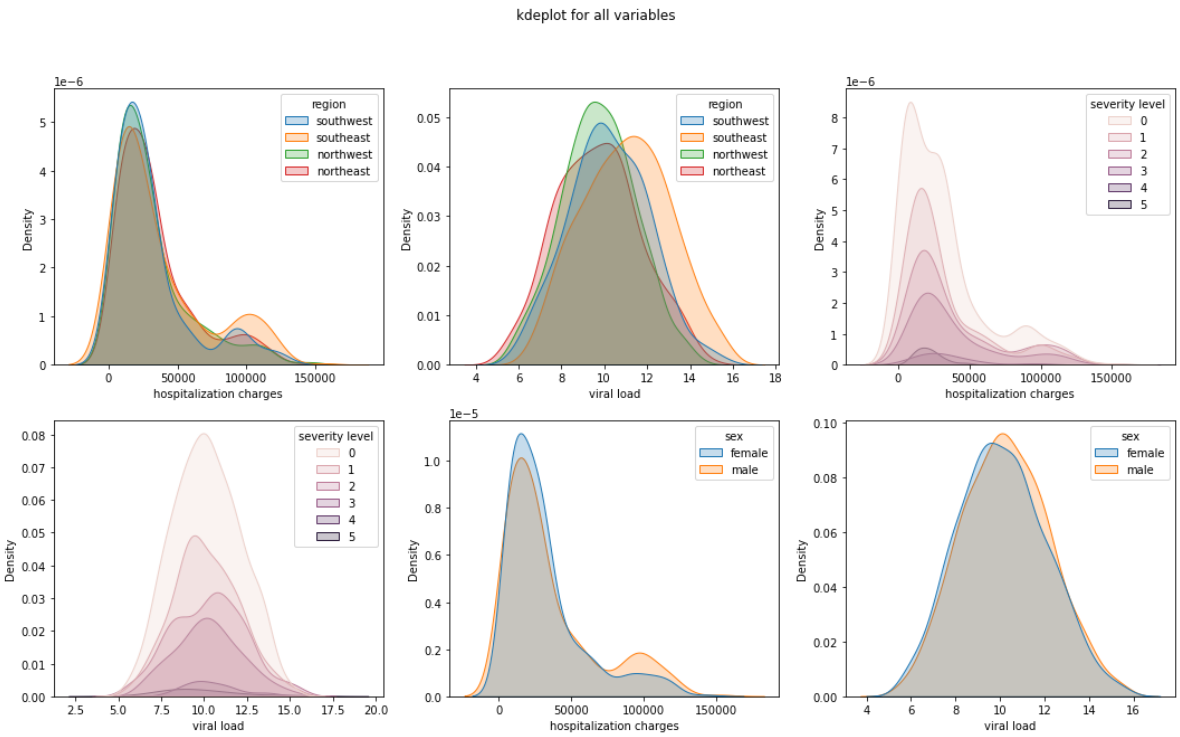
Out[46]:

```
<AxesSubplot:xlabel='viral load_bins', ylabel='count'>
```



In [47]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('kdeplot for all variables')
sns.kdeplot(ax=axes[0, 0], data=df,x='hospitalization charges',hue='region',shade=True)
sns.kdeplot(ax=axes[0, 1], data=df, x='viral load',hue='region',shade=True)
sns.kdeplot(ax=axes[0, 2], data=df, x='hospitalization charges',hue='severity level',shade=
sns.kdeplot(ax=axes[1, 0], data=df, x='viral load',hue='severity level',shade=True)
sns.kdeplot(ax=axes[1, 1], data=df, x='hospitalization charges',hue='sex',shade=True)
sns.kdeplot(ax=axes[1, 2], data=df, x='viral load',hue='sex',shade=True)
plt.show()

# Kde plot for all numerical variables
```



In [48]:

```
df.groupby('region')[['hospitalization charges']].aggregate({'hospitalization charges':['mi
# Min ,Max , mean , median and count for each region for hospital charges
# Median for southeast is the highest
```

Out[48]:

	hospitalization charges				
	min	max	mean	median	count
region					
northeast	4237	146428	33546.721362	25180.0	323
northwest	4053	150053	31043.941538	22414.0	325
southeast	2805	156482	36572.859944	23209.0	357
southwest	3104	131477	30606.787037	21976.0	324

In [49]:

```
df.groupby('region')[['viral load']].aggregate({'viral load':['min','max','mean','median','
# Min ,Max , mean , median and count for each region for viral load
# Median for southeast is the highest an lowest for northeast and northwest
```

Out[49]:

	viral load				
	min	max	mean	median	count
region					
northeast	5.32	14.92	9.705232	9.630	323
northwest	5.80	14.31	9.733508	9.630	325
southeast	6.60	15.58	11.011793	11.110	357
southwest	5.80	15.57	10.181481	10.085	324

In [50]:

```
df.groupby(['region','sex'])[['hospitalization charges','viral load']].aggregate({'hospital
# Min ,Max , mean , median  for each region and sex for viral load
# Difference between Median for southeast between male and female is higher
```

Out[50]:

		hospitalization charges		viral load	
		mean	median	mean	median
region	sex				
northeast	female	32438.025000	25710.5	9.736187	9.675
	male	34635.012270	24894.0	9.674847	9.630
northwest	female	31199.689024	24035.5	9.759634	9.645
	male	30885.291925	21034.0	9.706894	9.630
southeast	female	33026.913793	21440.5	10.862126	10.725
	male	39944.415301	23761.0	11.154098	11.290
southwest	female	27644.993789	21305.0	9.983789	9.900
	male	33532.239264	23478.0	10.376748	10.270

In [51]:

```
df.groupby(['region','smoker'])[['hospitalization charges']].aggregate({'hospitalization ch
# Min ,Max , mean , median  for each region and sex for hospital charges
# All smokers have greater bills than non smokers
```

Out[51]:

		hospitalization charges				
		min	max	mean	median	count
region	smoker					
northeast	no	4237	80272	22911.226562	20846.5	256
	yes	32074	146428	74183.835821	70253.0	67
northwest	no	4053	83680	21391.176030	18142.0	267
	yes	36779	150053	75479.948276	68722.5	58
southeast	no	2805	91451	20149.074627	16557.5	268
	yes	41444	156482	86028.752809	92913.0	89
southwest	no	3104	92277	20048.194757	18370.0	267
	yes	34611	131477	80065.456140	87097.0	57

In [52]:

```
df.groupby(['region','smoker'])[['viral load']].aggregate({'viral load':['min','max','mean'
# Viral load median for regions remains the almost the same for smoker and a non smoker
```

Out[52]:

		viral load				
		min	max	mean	median	count
region	smoker					
northeast	no	5.32	14.92	9.753320	9.63	256
	yes	5.73	14.25	9.521493	9.50	67
northwest	no	5.80	14.31	9.737828	9.78	267
	yes	5.92	13.52	9.713621	9.44	58
southeast	no	6.78	15.58	11.046940	11.11	268
	yes	6.60	15.40	10.905955	11.04	89
southwest	no	5.80	15.57	10.169438	10.07	267
	yes	6.10	13.30	10.237895	10.43	57

In [53]:

```
df.groupby(['region','smoker'])[['age']].aggregate({'age':['min','max','mean','median','cou
# Difference in Median of age in southwest region between smoker and a non smoker is greate
```

Out[53]:

		age				
		min	max	mean	median	count
region	smoker					
northeast	no	18	64	39.511719	40.0	256
	yes	18	63	38.238806	39.0	67
northwest	no	19	64	39.168539	39.0	267
	yes	19	64	39.327586	42.0	58
southeast	no	18	64	38.656716	38.5	268
	yes	18	64	39.775281	42.0	89
southwest	no	19	64	40.183521	41.0	267
	yes	19	64	36.087719	33.0	57

In [54]:

```
# Normality Tests
def qqplot(dff,a):
    A=pd.DataFrame()
    print('This test is for visual only')
    fig=sm.qqplot(dff[a],line='45')
    plt.grid()
    plt.show()
def kstest(dff,a):
    A=pd.DataFrame()
    stat,p_value=stats.kstest(dff[a],'norm')
    print('Ho: The sample {} follows normal distribution'.format(a))
    print('Ha: The sample {} does not follows normal distribution'.format(a))
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    print()
    if p_value>0.05:
        print('The sample {} follows normal distribution'.format(a))
    else:
        print('The sample {} does not follows normal distribution'.format(a))
def shapiro(dff,a):
    A=pd.DataFrame()
    stat,p_value=stats.shapiro(dff[a])
    print('Ho: The sample {} follows normal distribution'.format(a))
    print('Ha: The sample {} does not follows normal distribution'.format(a))
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample {} follows normal distribution'.format(a))
    else:
        print('The sample {} does not follows normal distribution'.format(a))
```

In [55]:

```
# Tranformations and the tests from 'Normality Tests'
def logtrans(dff,a):
    A=pd.DataFrame()
    A[a]=np.log(dff[a])
    print('After applying log transforms')
    qqplot(A,a)
    kstest(A,a)
    print()
    shapiro(A,a)

def box(dff,a):
    A=pd.DataFrame()
    fitted_data, fitted_lambda = stats.boxcox(dff[a])
    A[a]=fitted_data
    print('After applying boxcox transforms')
    qqplot(A,a)
    kstest(A,a)
    print()
    shapiro(A,a)

def rec(dff,a):
    A=pd.DataFrame()
    A[a]=1/dff[a]
    print('After applying reciprocal transforms')
    qqplot(A,a)
    kstest(A,a)
    print()
    shapiro(A,a)

def sq(dff,a):
    A=pd.DataFrame()
    A[a]=np.sqrt(dff[a])
    print('After applying log transforms')
    qqplot(A,a)
    kstest(A,a)
    print()
    shapiro(A,a)
```

In [56]:

```
# Correlation Tests
def pear(df1,df2,a,b,c):
    print('Ho: The sample {} and {} are independent'.format(b,c))
    print('Ha: The sample {} and {} are dependent'.format(b,c))
    stat, p_value = stats.pearsonr(df1[a], df2[a])
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample {} and {} are independent'.format(b,c))
    else:
        print('The sample {} and {} are dependent'.format(b,c))

def spearmanr(df1,df2,a,b,c):
    print('Ho: The sample {} and {} are independent'.format(b,c))
    print('Ha: The sample {} and {} are dependent'.format(b,c))
    stat, p_value = stats.spearmanr(df1[a], df2[b])
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample {} and {} are independent'.format(b,c))
    else:
        print('The sample {} and {} are dependent'.format(b,c))

def kend(df1,df2,a,b,c):
    print('Ho: The sample {} and {} are independent'.format(b,c))
    print('Ha: The sample {} and {} are dependent'.format(b,c))
    stat, p_value = stats.kendalltau(df1[a], df2[b])
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample {} and {} are independent'.format(b,c))
    else:
        print('The sample {} and {} are dependent'.format(b,c))
```

In [57]:

```
# Variance tests
def bar(df1,df2,a,b,c):
    stat, p_value = stats.bartlett(df1[a], df2[a])
    print('Ho: The sample {} and {} have equal variance'.format(b,c))
    print('Ha: The sample {} and {} are unequal variance'.format(b,c))
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample {} and {} have equal variance'.format(b,c))
    else:
        print('The sample {} and {} unequal variance'.format(b,c))
def lev(df1,df2,a,b,c):
    stat, p_value = stats.levene(df1[a], df2[a],center='median')
    print('Ho: The sample {} and {} have equal variance'.format(b,c))
    print('Ha: The sample {} and {} are unequal variance'.format(b,c))
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample {} and {} have equal variance'.format(b,c))
    else:
        print('The sample {} and {} unequal variance'.format(b,c))
```

In [58]:

```
# tests when data is not normal
def mann(df1,df2,a,b,c):
    print('Ho: The sum of ranking of {} and {} are equal'.format(b,c))
    print('Ha: The sum of ranking of {} and {} are not equal'.format(b,c))
    print('Assumption of a Mann-Whitney U test are:')
    print('1.Should have a ordinal variable\n''2. Only 2 independent random samples with a
    stat,p_value=stats.mannwhitneyu(df1[a], df2[a])
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sum of ranking of {} and {} are equal'.format(b,c))
    else:
        print('The sum of ranking of {} and {} are not equal'.format(b,c))
def will(df1,df2,a,b,c):
    print('Ho: The central tendencies of {} and {} are equal'.format(b,c))
    print('Ha: The central tendencies of {} and {} are not equal'.format(b,c))
    print('Assumption of a Wilcoxon Signed-Rank Test are:')
    print('1.Should have a ordinal variable\n''2. Only 2 independent random samples with a
    stat,p_value=stats.mannwhitneyu(df1[a], df2[a])
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The central tendencies of {} and {} are equal'.format(b,c))
    else:
        print('The central tendencies of {} and {} are not equal'.format(b,c))
```

In [59]:

```
# Ttest for equal variance
def ttest(df1,df2,a,b,c,d):
    print('Ho: The sample means of {} and {} are equal'.format(b,c))
    print('Ha: The sample means of {} and {} are not equal'.format(b,c))
    print('Assumption of a t_test are:')
    print('1.Observations of 2 samples are independent\n''2. Both samples are approx normal
    stat,p_value=stats.ttest_ind(df1[a], df2[a], equal_var=d)
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample means of {} and {} are equal'.format(b,c))
    else:
        print('The sample means of {} and {} are not equal'.format(b,c))
```

In [60]:

```
def ttest1(df1,df2,a,b,c,d):
    print('Ho: The sample means of {} and {} are equal'.format(b,c))
    print('Ha: The sample means of {} is > than {} '.format(b,c))
    print('Assumption of a t_test are:')
    print('1.Observations of 2 samples are independent\n''2. Both samples are approx normal
    stat,p_value=stats.ttest_ind(df1[a], df2[a], equal_var=d,alternative='greater')
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample means of {} and {} are equal'.format(b,c))
    else:
        print('The sample means of {} is > than {} '.format(b,c))
```

In [61]:

```
# T test with unequal variance
def ttestv(df1,df2,a,b,c,d):
    print('Ho: The sample means of {} and {} are equal'.format(b,c))
    print('Ha: The sample means of {} and {} are not equal'.format(b,c))
    print('Assumption of a t_test are:')
    print('1.Observations of 2 samples are independent\n''2. Both samples are approx normal
    stat,p_value=stats.ttest_ind(df1[a], df2[a], equal_var=d)
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample means of {} and {} are equal'.format(b,c))
    else:
        print('The sample means of {} and {} are not equal'.format(b,c))
```

In [62]:

```
#chi2
def chis(df1,df2,a):
    stat, p_value, dof, ex=stats.chi2_contingency(df1[a],df2[a])
    print('Ho: The samples are independent')
    print('Ha: The samples dependent')
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The samples are independent')
    else:
        print('The samples are dependent')
```

In [63]:

```
def chis1(df1,df2,df3,a):
    stat, p_value, dof, ex=stats.chi2_contingency(df1[a],df2[a],df3[a])
    print('Ho: The samples are independent')
    print('Ha: The samples dependent')
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The samples are independent')
    else:
        print('The samples are dependent')
```

In [64]:

```
def chis2(df1,df2,df3,df4,a):
    stat, p_value, dof, ex=stats.chi2_contingency(df1[a],df2[a],df3[a],df4[a])
    print('Ho: The samples are independent')
    print('Ha: The samples dependent')
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The samples are independent')
    else:
        print('The samples are dependent')
```

In [65]:

```
def chis22(df1):
    stat, p_value, dof, ex=stats.chi2_contingency(df1)
    print('Ho: The samples are independent')
    print('Ha: The samples dependent')
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The samples are independent')
    else:
        print('The samples are dependent')
```

In [66]:

```
# ANOVA
def anova(df1,df2,df3,a):
    print('Assumptions of ANOVA are:')
    print('1. Each group assumptions is gaussian\n 2.Each group variance is roughly the sam
    print('Ho: The sample means equal')
    print('Ha: There exists atleast one sample that is not equal to other mean')
    minn=min(len(df1),len(df2),len(df3))
    stat,p_value=stats.f_oneway(np.random.choice(df1[a],minn),np.random.choice(df2[a],minn)
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample means equal')
    else:
        print('The sample means are not equal')
def kwtest(df1,df2,df3,df4,a):
    print('Assumptions of Kruskal-Wallis one-way analysis of variance are:')
    print('Ho: The sample median equal')
    print('Ha: There exists atleast one sample that is not equal to other median')
    stat,p_value=stats.kruskal(df1[a],df2[a],df3[a],df4[a])
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample medians equal')
    else:
        print('The sample medians are not equal')

def kwtest1(df1,df2,df3,a):
    print('Assumptions of Kruskal-Wallis one-way analysis of variance are:')
    print('Ho: The sample median equal')
    print('Ha: There exists atleast one sample that is not equal to other median')
    stat,p_value=stats.kruskal(df1[a],df2[a],df3[a])
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample medians equal')
    else:
        print('The sample medians are not equal')
```

In [67]:

```
def conf(dff,para):
    per=[0.05,0.025,0.005]
    ls=[]
    for i in range(80000):
        ans=np.random.choice(dff[para],len(dff),replace=True)
        l1=np.mean(ans)
        ls.append(l1)
    print('Confidence interval using Bootstrap : ')
    print('Confidence interval for {} group is [{},{}] with {}% confidence'.format(para,np.
    print('Confidence interval for {} group is [{},{}] with {}% confidence'.format(para,np.
    print('Confidence interval for {} group is [{},{}] with {}% confidence'.format(para,np.
```

Prove (or disprove) that the hospitalization of people who do

smoking is greater than those who don't? (T-test Right tailed)

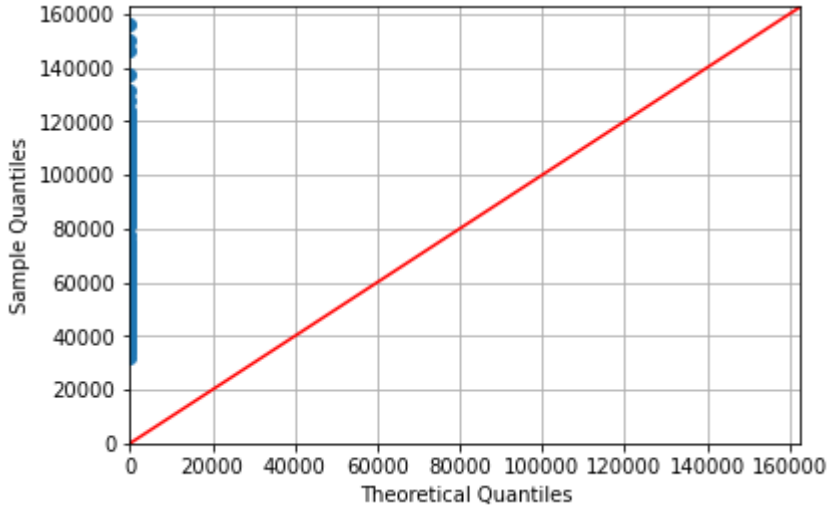
In [68]:

```
df1=df[df['smoker']=='yes']
df2=df[df['smoker']=='no']
```

In [69]:

```
qqplot(df1,'hospitalization charges')
# Plot shows that distribution does not follow normal distribution
```

This test is for visual only



In [70]:

```
kstest(df1,'hospitalization charges')
# KS test shows that The sample hospitalization charges does not follows normal distributio
```

```
Ho: The sample hospitalization charges follows normal distribution
Ha: The sample hospitalization charges does not follows normal distribution
stat=1.000, p_value=0.000

The sample hospitalization charges does not follows normal distribution
```

In [71]:

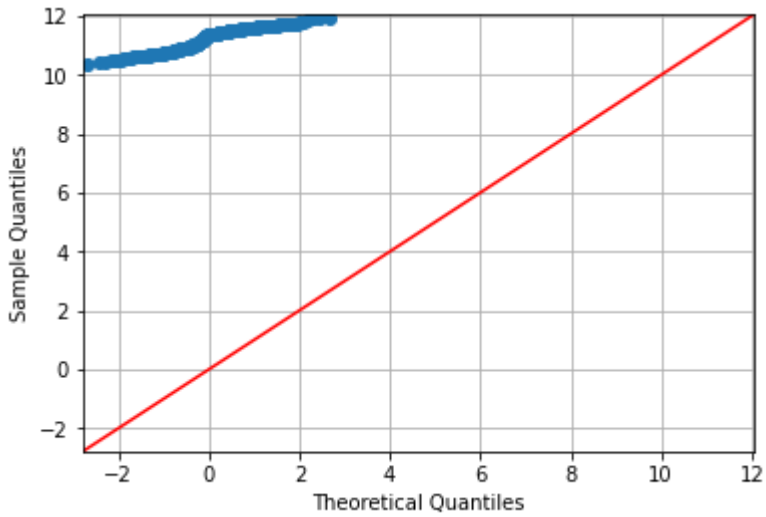
```
shapiro(df1,'hospitalization charges')
# Shapiro test shows that The sample hospitalization charges does not follows normal distri
```

```
Ho: The sample hospitalization charges follows normal distribution
Ha: The sample hospitalization charges does not follows normal distribution

stat=0.938, p_value=0.000
The sample hospitalization charges does not follows normal distribution
```

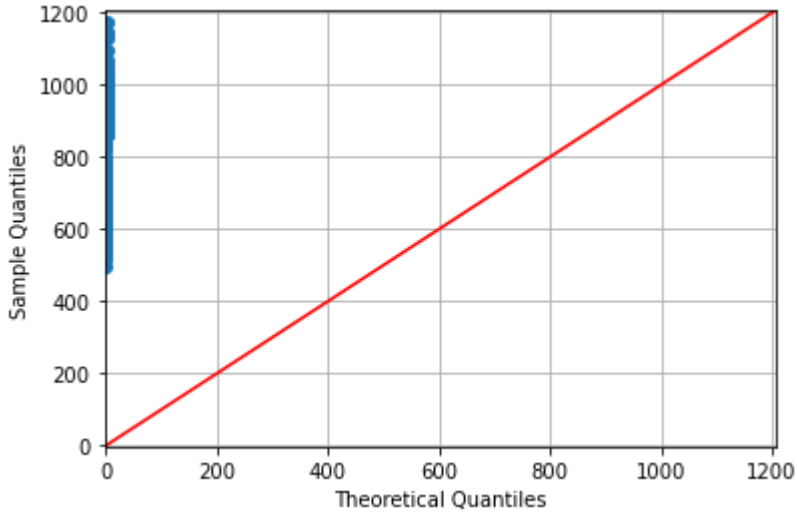
In [72]:

```
logtrans(df1,'hospitalization charges')
box(df1,'hospitalization charges')
# Even after applying log and box transforms the distribution is not normal
```

```
After applying log transforms
This test is for visual only
```




```
Ho: The sample hospitalization charges follows normal distribution
Ha: The sample hospitalization charges does not follows normal distribution
stat=1.000, p_value=0.000

The sample hospitalization charges does not follows normal distribution

Ho: The sample hospitalization charges follows normal distribution
Ha: The sample hospitalization charges does not follows normal distribution

stat=0.927, p_value=0.000
The sample hospitalization charges does not follows normal distribution
After applying boxcox transforms
This test is for visual only
```




```
Ho: The sample hospitalization charges follows normal distribution
Ha: The sample hospitalization charges does not follows normal distribution
stat=1.000, p_value=0.000

The sample hospitalization charges does not follows normal distribution

Ho: The sample hospitalization charges follows normal distribution
Ha: The sample hospitalization charges does not follows normal distribution

stat=0.936, p_value=0.000
The sample hospitalization charges does not follows normal distribution
```

In [73]:

```
bar(df1,df2,'hospitalization charges','Smoking','Non-Smoking')
# Bartlett's test shoes that The sample Smoking and Non-Smoking  have unequal variance
```

```
Ho: The sample Smoking and Non-Smoking have equal variance
Ha: The sample Smoking and Non-Smoking are unequal variance
stat=217.014, p_value=0.000
The sample Smoking and Non-Smoking unequal variance
```

In [74]:

```
lev(df1,df2,'hospitalization charges','Smoking','Non-Smoking')
# Levene's test shoes that The sample Smoking and Non-Smoking  have unequal variance
```

```
Ho: The sample Smoking and Non-Smoking have equal variance
Ha: The sample Smoking and Non-Smoking are unequal variance
stat=324.497, p_value=0.000
The sample Smoking and Non-Smoking unequal variance
```

In [75]:

```
ttest(df1,df2,'hospitalization charges','Smoking','Non-Smoking','Flase')
print('----------------')
ttest1(df1,df2,'hospitalization charges','Smoking','Non-Smoking','Flase')
print('----------------')
ttestv(df1,df2,'hospitalization charges','Smoking','Non-Smoking','Flase')
# T test says that The sample means of Smoking is > than Non-Smoking
```

```
Ho: The sample means of Smoking and Non-Smoking are equal
Ha: The sample means of Smoking and Non-Smoking are not equal
Assumption of a t_test are:
1.Observations of 2 samples are independent
2. Both samples are approx normal distribution
3. Population standard deviation is not available

stat=46.300, p_value=0.000
The sample means of Smoking and Non-Smoking are not equal
----------------
Ho: The sample means of Smoking and Non-Smoking are equal
Ha: The sample means of Smoking is > than Non-Smoking
Assumption of a t_test are:
1.Observations of 2 samples are independent
2. Both samples are approx normal distribution
3. Population standard deviation is not available

stat=46.300, p_value=0.000
The sample means of Smoking is > than Non-Smoking
----------------
Ho: The sample means of Smoking and Non-Smoking are equal
Ha: The sample means of Smoking and Non-Smoking are not equal
Assumption of a t_test are:
1.Observations of 2 samples are independent
2. Both samples are approx normal distribution
3. Population standard deviation is not available

stat=46.300, p_value=0.000
The sample means of Smoking and Non-Smoking are not equal
```

In [76]:

```
mann(df1,df2,'hospitalization charges','Smoking','Non-Smoking')
print('----------------')
will(df1,df2,'hospitalization charges','Smoking','Non-Smoking')
# Mann-Whitney U test and Wilconon test says that The sum of ranking of Smoking and Non-Smo
# The central tendencies of Smoking and Non-Smoking are not equal respectively
```

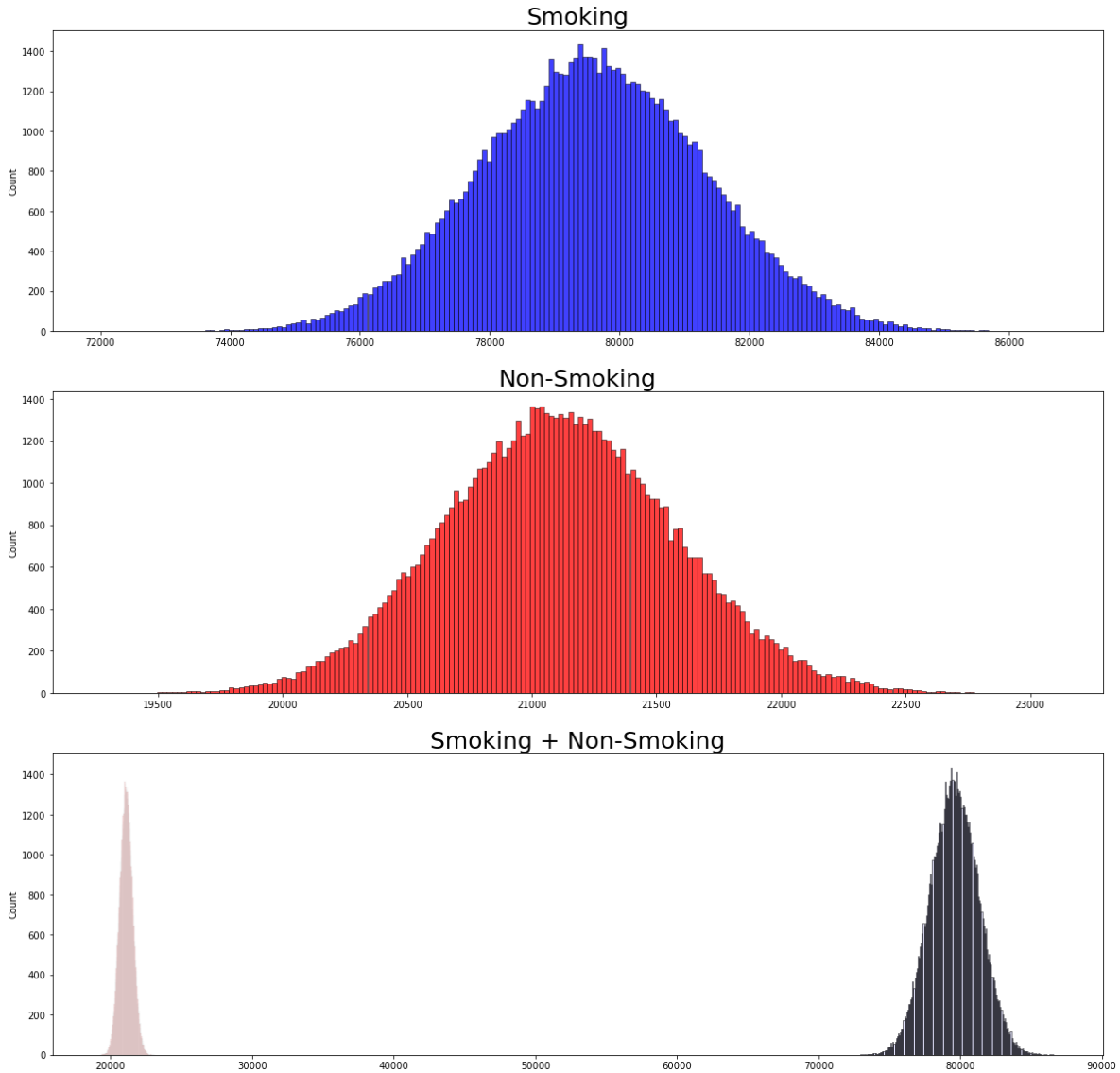
```
Ho: The sum of ranking of Smoking and Non-Smoking are equal
Ha: The sum of ranking of Smoking and Non-Smoking are not equal
Assumption of a Mann-Whitney U test are:
1.Should have a ordinal variable
2. Only 2 independent random samples with a least ordinally scales character
istics


stat=279314.500, p_value=0.000
The sum of ranking of Smoking and Non-Smoking are not equal
----------------
Ho: The central tendencies of Smoking and Non-Smoking are equal
Ha: The central tendencies of Smoking and Non-Smoking are not equal
Assumption of a Wilcoxon Signed-Rank Test are:
1.Should have a ordinal variable
2. Only 2 independent random samples with a least ordinally scales character
istics


stat=279314.500, p_value=0.000
The central tendencies of Smoking and Non-Smoking are not equal
```

In [77]:

```
ls=[]
for i in range(80000):
    ans=np.random.choice(df1['hospitalization charges'],len(df1),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(3, 1, figsize=(20, 20))
plt.subplot(3,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[2],alpha=0.1,bins=200)
ls=[]
for i in range(80000):
    ans=np.random.choice(df2['hospitalization charges'],len(df2),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(3,1,2)
sns.histplot(ls,color='r',ax=axes[1],bins=200)
sns.histplot(ls,color='r',ax=axes[2],alpha=0.1,bins=200)
axes[0].set_title('Smoking',fontsize=25)
axes[1].set_title('Non-Smoking',fontsize=25)
axes[2].set_title('Smoking + Non-Smoking',fontsize=25)
# Confidence intervals plots
```

Out[77]:

```
Text(0.5, 1.0, 'Smoking + Non-Smoking')
```



In [78]:

```
conf(df1,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [76730.59114391144,
82424.52527675277] with 90.0% confidence
Confidence interval for hospitalization charges group is [76180.97896678967,
82964.78090405904] with 95.0% confidence
Confidence interval for hospitalization charges group is [75158.1683394834,8
4042.84304428044] with 99.0% confidence
```

In [79]:

```
conf(df2,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [20353.17731568998,
21867.96361058601] with 90.0% confidence
Confidence interval for hospitalization charges group is [20211.63310491493
4,22010.69884215501] with 95.0% confidence
Confidence interval for hospitalization charges group is [19939.28224007561
6,22301.714106805295] with 99.0% confidence
```

In [ ]:

```

```

Prove (or disprove) with statistical evidence that the viral load of females is different from that of males (T-test Two tailed)

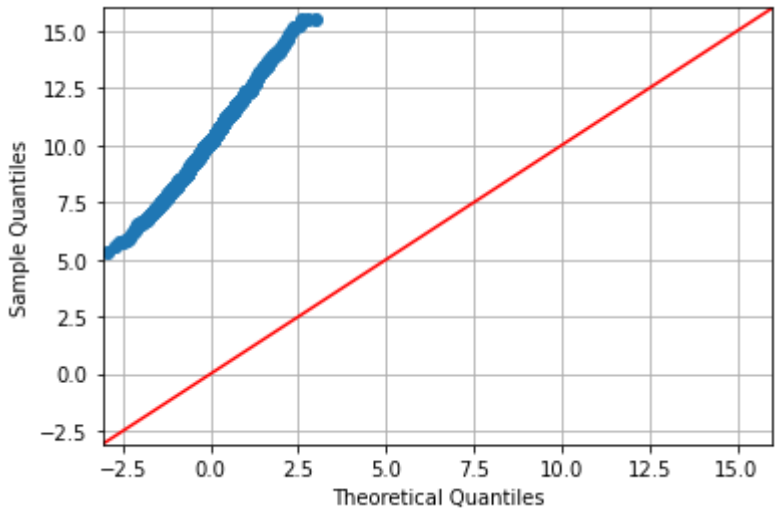
In [80]:

```
df1=df[df['sex']=='male']
df2=df[df['sex']=='female']
```

In [81]:

```
qqplot(df1,'viral load')
# Plot shows that distribution does not follow normal distribution
```

```
This test is for visual only
```



In [82]:

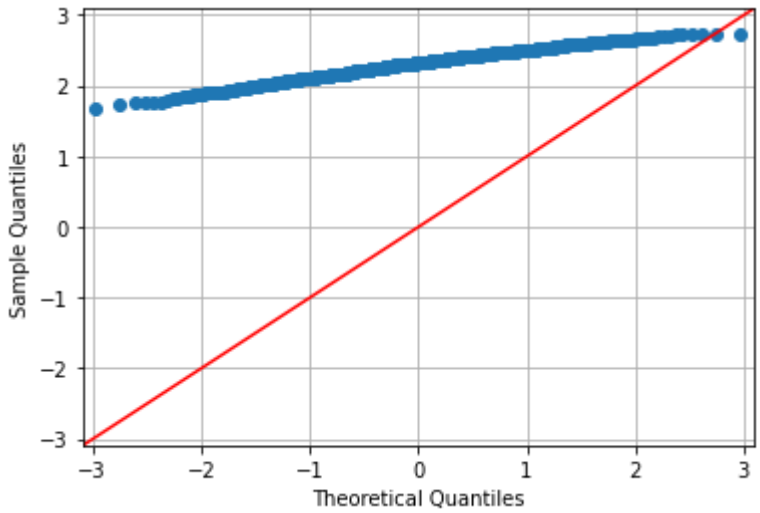
```
kstest(df1,'viral load')
# KS test shows that The sample viral load does not follows normal distribution
```

```
Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution
stat=1.000, p_value=0.000

The sample viral load does not follows normal distribution
```

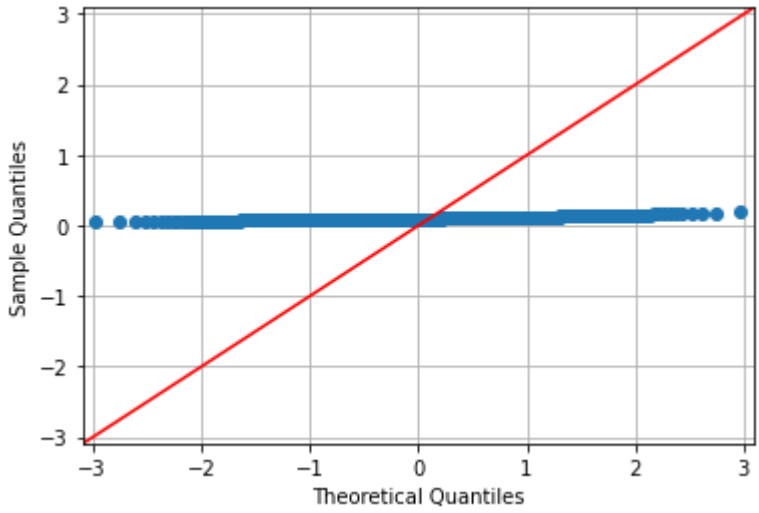
In [85]:

```
logtrans(df1,'viral load')
rec(df1,'viral load')
#Even after applying log and box transforms the distribution is not normal
```

```
After applying log transforms
This test is for visual only
```




```
Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution
stat=0.957, p_value=0.000

The sample viral load does not follows normal distribution

Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution

stat=0.990, p_value=0.000
The sample viral load does not follows normal distribution
After applying reciprocal transforms
This test is for visual only
```



```
Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution
stat=0.526, p_value=0.000

The sample viral load does not follows normal distribution

Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution

stat=0.952, p_value=0.000
The sample viral load does not follows normal distribution
```

In [86]:

```
ttestv(df1,df2,'viral load','male','female','False')
# T test says that The sample means of The sample means of male and female are equal for vi
```

```
Ho: The sample means of male and female are equal
Ha: The sample means of male and female are not equal
Assumption of a t_test are:
1.Observations of 2 samples are independent
2. Both samples are approx normal distribution
3. Population standard deviation is not available

stat=1.456, p_value=0.146
The sample means of male and female are equal
```

In [87]:

```
mann(df1,df2,'viral load','male','female')
will(df1,df2,'viral load','male','female')
# Mann-Whitney U test and Wilconon test says that The sum of ranking of male and female are
# The central tendencies of male and female are equal respectively
```

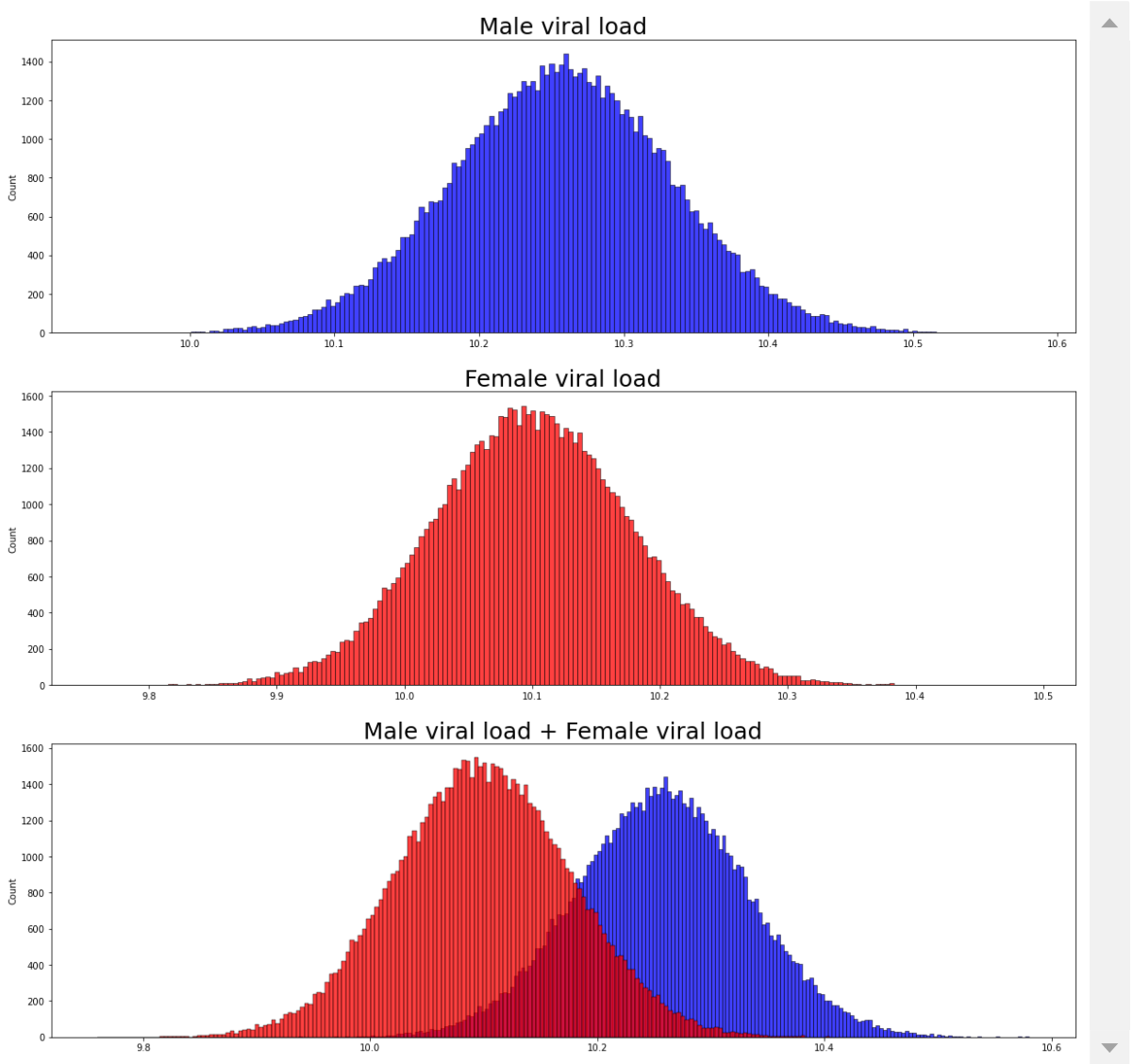
```
Ho: The sum of ranking of male and female are equal
Ha: The sum of ranking of male and female are not equal
Assumption of a Mann-Whitney U test are:
1.Should have a ordinal variable
2. Only 2 independent random samples with a least ordinally scales character
istics


stat=231350.000, p_value=0.130
The sum of ranking of male and female are equal
Ho: The central tendencies of male and female are equal
Ha: The central tendencies of male and female are not equal
Assumption of a Wilcoxon Signed-Rank Test are:
1.Should have a ordinal variable
2. Only 2 independent random samples with a least ordinally scales character
istics


stat=231350.000, p_value=0.130
The central tendencies of male and female are equal
```

In [88]:

```
ls=[]
for i in range(80000):
    ans=np.random.choice(df1['viral load'],len(df1),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(3, 1, figsize=(20, 20))
plt.subplot(3,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[2],bins=200)
ls=[]
for i in range(80000):
    ans=np.random.choice(df2['viral load'],len(df2),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(3,1,2)
sns.histplot(ls,color='r',ax=axes[1],bins=200)
sns.histplot(ls,color='r',ax=axes[2],bins=200)
axes[0].set_title('Male viral load',fontsize=25)
axes[1].set_title('Female viral load',fontsize=25)
axes[2].set_title('Male viral load + Female viral load',fontsize=25)
# Confidence intervals plots
```

Out[88]:

```
Text(0.5, 1.0, 'Male viral load + Female viral load')
```



In [89]:

```
conf(df1,'viral load')
```

```
Confidence interval using Bootstrap :
Confidence interval for viral load group is [10.132939552238804,10.382373134
32836] with 90.0% confidence
Confidence interval for viral load group is [10.109566417910447,10.405866791
044776] with 95.0% confidence
Confidence interval for viral load group is [10.061611343283582,10.454164701
492537] with 99.0% confidence
```

In [90]:

```
conf(df2,'viral load')
```

```
Confidence interval using Bootstrap :
Confidence interval for viral load group is [9.972412746585736,10.2276631259
48407] with 90.0% confidence
Confidence interval for viral load group is [9.949210166919576,10.2519578907
43551] with 95.0% confidence
Confidence interval for viral load group is [9.900439908952958,10.2994537177
54174] with 99.0% confidence
```

Is the proportion of smoking significantly different across

different regions? (Chi-square)

In [91]:

```
df1=df[['region','smoker']]
```

In [92]:

```
df1=pd.crosstab(index=df1['region'],columns=df1['smoker'])
```

In [93]:

```
chis22(df1)
```

```
Ho: The samples are independent
Ha: The samples dependent
stat=7.415, p_value=0.060
The samples are independent
```

Is the mean viral load of women with 0 Severity level , 1 Severity level, and 2 Severity level the same? Explain your answer with statistical evidence (One way Anova)

In [94]:

```
dff=df[df['sex']=='female']
```

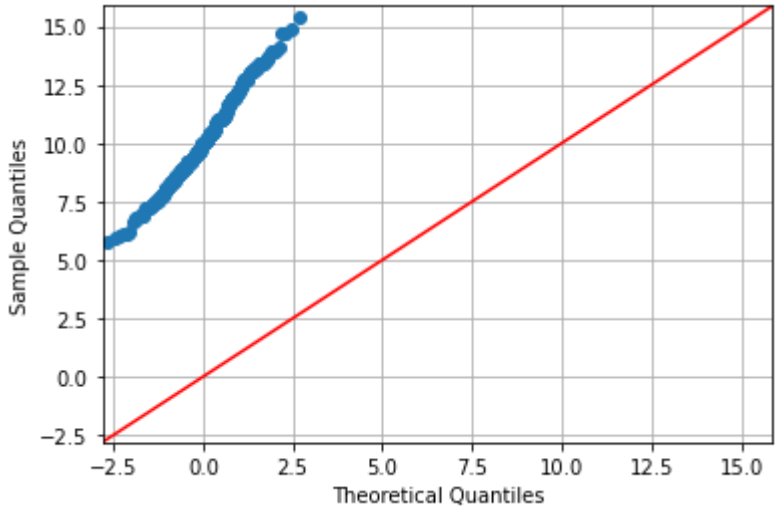
In [95]:

```
df1=dff[dff['severity level'] == 0]
df2=dff[dff['severity level'] == 1]
df3=dff[dff['severity level'] == 2]
```

In [96]:

```
qqplot(df1,'viral load')
# Plot shows that distribution does not follow normal distribution
```

```
This test is for visual only
```



In [97]:

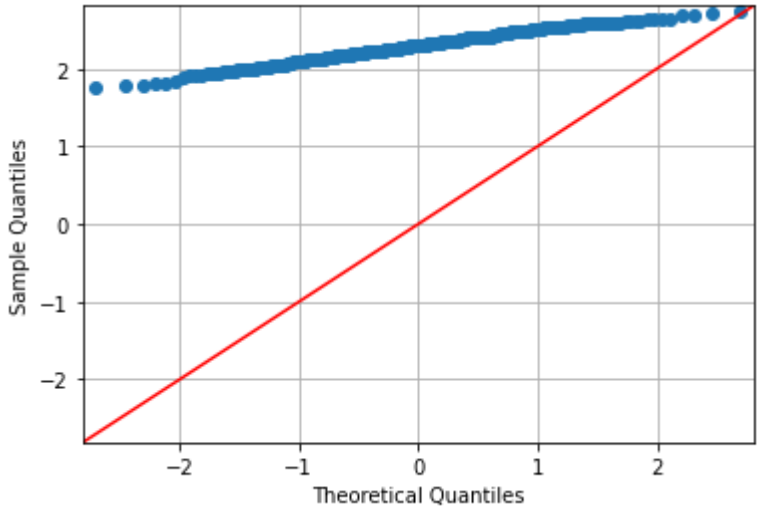
```
kstest(df1,'viral load')
# KS test shows that The sample viral load does not follows normal distribution
```

```
Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution
stat=1.000, p_value=0.000

The sample viral load does not follows normal distribution
```

In [98]:

```
logtrans(df1,'viral load')
# Even after applying log transforms the distribution is not normal
```

```
After applying log transforms
This test is for visual only
```




```
Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution
stat=0.960, p_value=0.000

The sample viral load does not follows normal distribution

Ho: The sample viral load follows normal distribution
Ha: The sample viral load does not follows normal distribution

stat=0.990, p_value=0.046
The sample viral load does not follows normal distribution
```

In [99]:

```
def bar1(df1,df2,df3,a,b,c,d):
    stat, p_value = stats.bartlett(df1[a], df2[a],df3[a])
    print('Ho: The sample {},{} and {} have equal variance'.format(b,c,d))
    print('Ha: The sample {},{} and {} are unequal variance'.format(b,c,d))
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample {}, {}  and {} have equal variance'.format(b,c,d))
    else:
        print('The sample {}, {}  and {} unequal variance'.format(b,c,d))
# # Bartlett's test shows that The sample severity level 0, severity level 1  and severity
```

In [100]:

```
bar1(df1,df2,df3,'viral load','severity level 0','severity level 1','severity level 2')
```

```
Ho: The sample severity level 0,severity level 1 and severity level 2 have e
qual variance
Ha: The sample severity level 0,severity level 1 and severity level 2 are un
equal variance
stat=1.018, p_value=0.601
The sample severity level 0, severity level 1  and severity level 2 have equ
al variance
```

Type *Markdown* and LaTeX: α^2

In [101]:

```
chis1(df1,df2,df3,'viral load')
# Chi 2 test shows that the samples are independent
```

```
Ho: The samples are independent
Ha: The samples dependent
stat=0.000, p_value=1.000
The samples are independent
```

In [102]:

```
anova(df1,df2,df3,'viral load')
# Anova test shoes that the samples means are not equal
```

```
Assumptions of ANOVA are:
1. Each group assumptions is gaussian
 2.Each group variance is roughly the same
 3.Each observation is independent
 4.Different groups must have equal sample size
Ho: The sample means equal
Ha: There exists atleast one sample that is not equal to other mean

stat=3.896, p_value=0.021
The sample means are not equal
```

In [103]:

```
kwtest1(df1,df2,df3,'viral load')
# kruskal wallis test shoes that the sample medians are equal
```

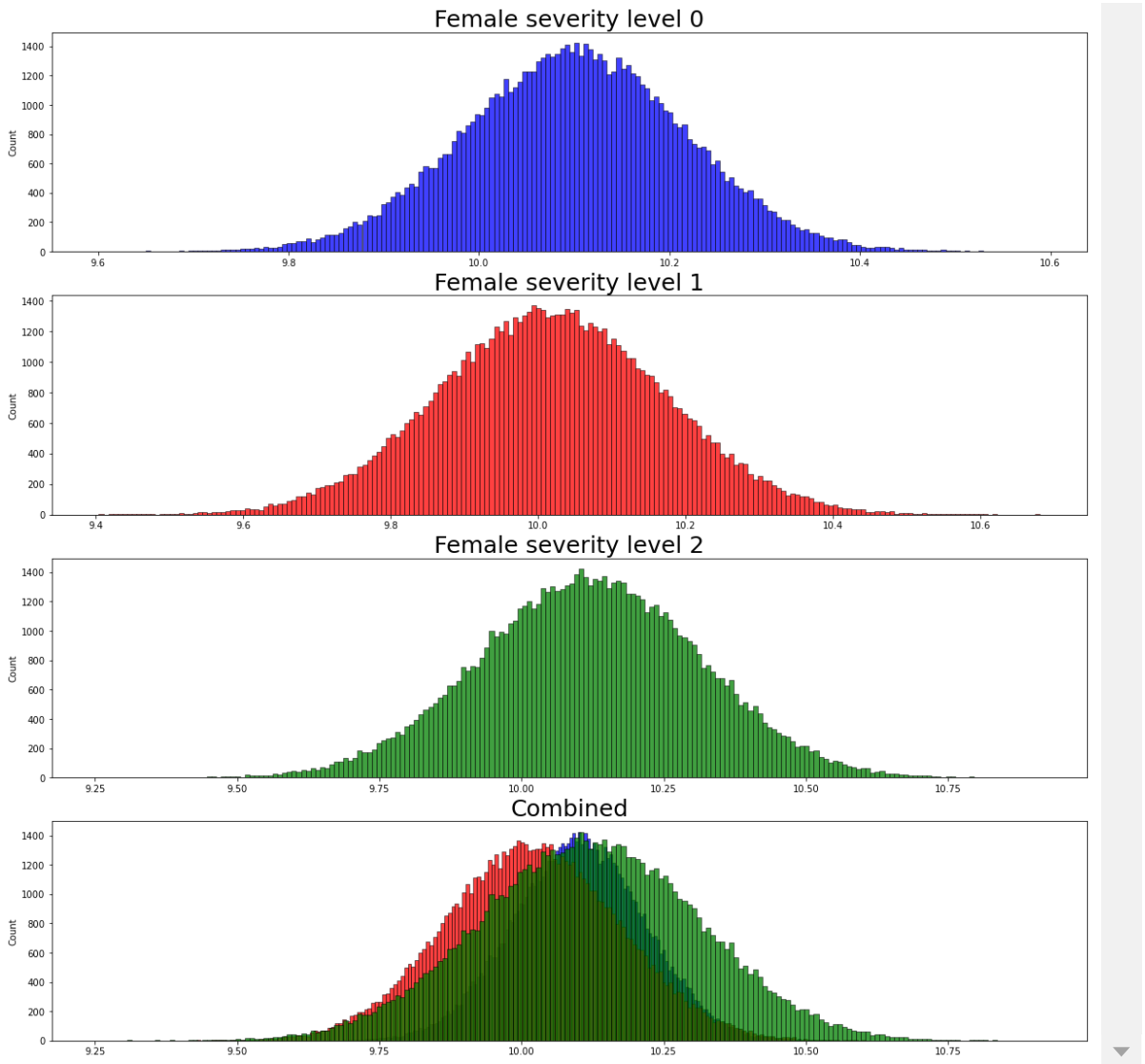
```
Assumptions of Kruskal-Wallis one-way analysis of variance are:
Ho: The sample median equal
Ha: There exists atleast one sample that is not equal to other median

stat=0.387, p_value=0.824
The sample medians equal
```

In [104]:

```
ls=[]
for i in range(80000):
    ans=np.random.choice(df1['viral load'],len(df1),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(4, 1, figsize=(20, 20))
plt.subplot(4,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[3],bins=200)
ls=[]
for i in range(80000):
    ans=np.random.choice(df2['viral load'],len(df2),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(4,1,2)
sns.histplot(ls,color='r',ax=axes[1],bins=200)
sns.histplot(ls,color='r',ax=axes[3],bins=200)
ls=[]
for i in range(80000):
    ans=np.random.choice(df3['viral load'],len(df3),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(4,1,3)
sns.histplot(ls,color='g',ax=axes[2],bins=200)
sns.histplot(ls,color='g',ax=axes[3],bins=200)



axes[0].set_title('Female severity level 0',fontsize=25)
axes[1].set_title('Female severity level 1',fontsize=25)
axes[2].set_title('Female severity level 2',fontsize=25)
axes[3].set_title('Combined',fontsize=25)
# Confidence intervals plots
```

Out[104]:

```
Text(0.5, 1.0, 'Combined')
```



In [105]:

```
df1=df[df['severity level'] == 0]
df2=df[df['severity level'] == 1]
df3=df[df['severity level'] == 2]
df4=df[df['severity level'] == 3]
df5=df[df['severity level'] == 4]
df6=df[df['severity level'] == 5]
```

In [106]:

```
def kwtest(df1,df2,df3,df4,df5,df6,a):
    print('Assumptions of Kruskal-Wallis one-way analysis of variance are:')
    print('Ho: The sample median equal')
    print('Ha: There exists atleast one sample that is not equal to other median')
    stat,p_value=stats.kruskal(df1[a],df2[a],df3[a],df4[a],df5[a],df6[a])
    print()
    print('stat=%.3f, p_value=%.3f' % (stat, p_value))
    if p_value>0.05:
        print('The sample medians equal')
    else:
        print('The sample medians are not equal')
```

In [107]:

```
kwtest(df1,df2,df3,df4,df5,df6,'hospitalization charges')
# kruskal wallis test shoes that the sample medians are equal
```

```
Assumptions of Kruskal–Wallis one-way analysis of variance are:
Ho: The sample median equal
Ha: There exists atleast one sample that is not equal to other median

stat=29.348, p_value=0.000
The sample medians are not equal
```

In [ ]:

```
# Below is the sample confidence intervals and plots
```

In [108]:

```
conf(df1,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [28734.93966725043
7,32799.2915061296] with 90.0% confidence
Confidence interval for hospitalization charges group is [28347.16379159369
5,33218.57854640981] with 95.0% confidence
Confidence interval for hospitalization charges group is [27644.94172504378
4,34000.46327495623] with 99.0% confidence
```

In [109]:

```
conf(df2,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [29040.28390625000
2,34473.12953125] with 90.0% confidence
Confidence interval for hospitalization charges group is [28573.158671875,35
008.716484375] with 95.0% confidence
Confidence interval for hospitalization charges group is [27650.48346875,360
50.32575000001] with 99.0% confidence
```

In [110]:

```
conf(df3,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [34057.65315126051,
40860.29453781513] with 90.0% confidence
Confidence interval for hospitalization charges group is [33449.88214285715,
41541.59957983193] with 95.0% confidence
Confidence interval for hospitalization charges group is [32290.14348739496,
42890.96834033614] with 99.0% confidence
```

In [111]:

```
conf(df4,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [34450.62643312102,
42505.41178343949] with 90.0% confidence
Confidence interval for hospitalization charges group is [33754.43312101911,
43341.33073248408] with 95.0% confidence
Confidence interval for hospitalization charges group is [32309.71828025478,
45029.4598089172] with 99.0% confidence
```

In [112]:

```
conf(df5,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [27666.85200000000
3,42355.952000000005] with 90.0% confidence
Confidence interval for hospitalization charges group is [26553.07100000000
4,44000.044] with 95.0% confidence
Confidence interval for hospitalization charges group is [24600.3154,47365.6
0360000004] with 99.0% confidence
```

In [113]:

```
conf(df6,'hospitalization charges')
```

```
Confidence interval using Bootstrap :
Confidence interval for hospitalization charges group is [18549.69166666666
6,25745.166666666668] with 90.0% confidence
Confidence interval for hospitalization charges group is [17985.87777777778,
26545.738888888885] with 95.0% confidence
Confidence interval for hospitalization charges group is [16911.88055555555
5,28086.336944444447] with 99.0% confidence
```

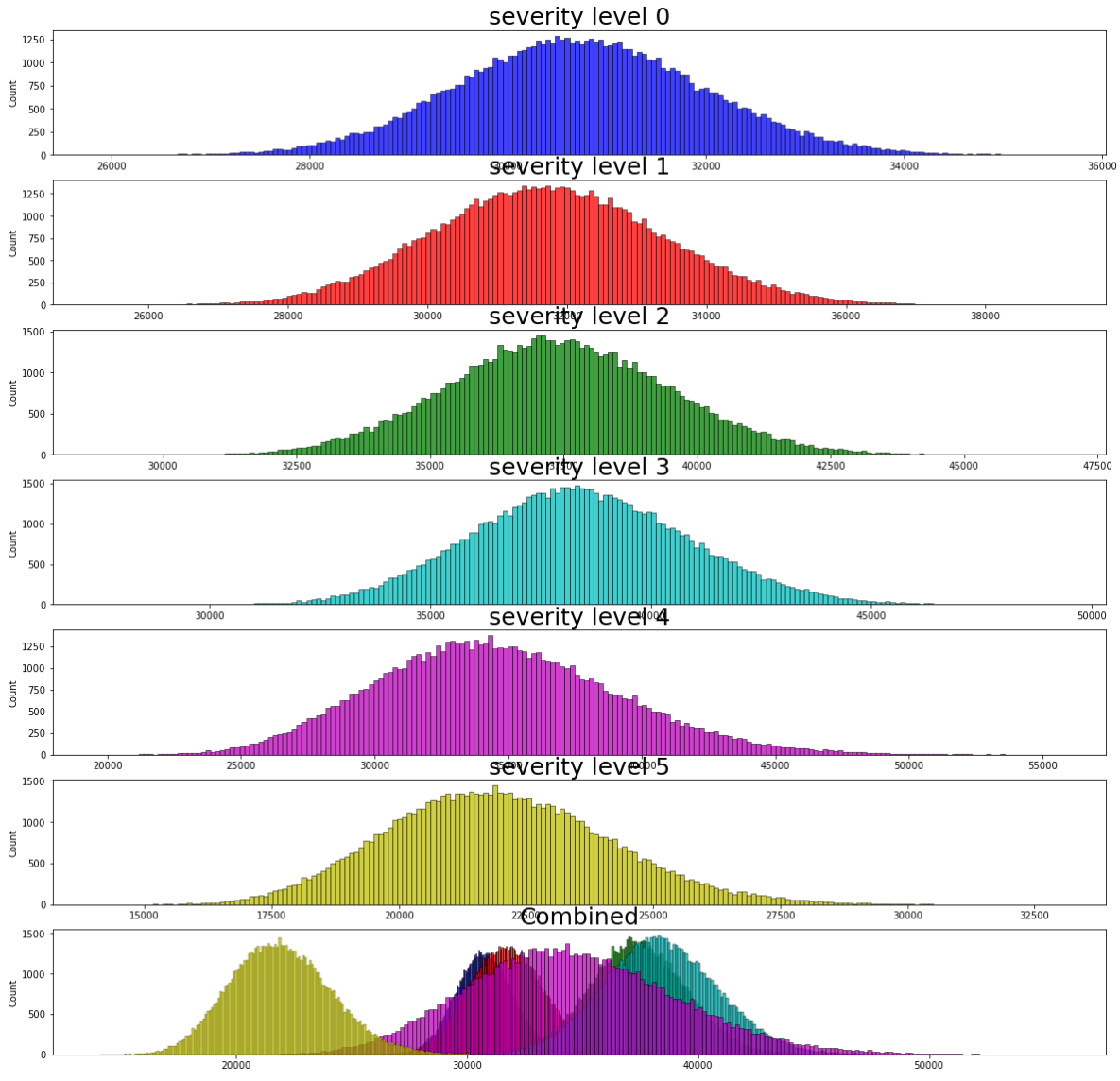
In [114]:

```
ls=[]
for i in range(80000):
    ans=np.random.choice(df1['hospitalization charges'],len(df1),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(7, 1, figsize=(20, 20))

plt.subplot(7,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[6],bins=200)


ls=[]
for i in range(80000):
    ans=np.random.choice(df2['hospitalization charges'],len(df2),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(7,1,2)
sns.histplot(ls,color='r',ax=axes[1],bins=200)
sns.histplot(ls,color='r',ax=axes[6],bins=200)


ls=[]
for i in range(80000):
    ans=np.random.choice(df3['hospitalization charges'],len(df3),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(7,1,3)
sns.histplot(ls,color='g',ax=axes[2],bins=200)
sns.histplot(ls,color='g',ax=axes[6],bins=200)


ls=[]
for i in range(80000):
    ans=np.random.choice(df4['hospitalization charges'],len(df4),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(7,1,4)
sns.histplot(ls,color='c',ax=axes[3],bins=200)
sns.histplot(ls,color='c',ax=axes[6],bins=200)



ls=[]
for i in range(80000):
    ans=np.random.choice(df5['hospitalization charges'],len(df5),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(7,1,5)
sns.histplot(ls,color='m',ax=axes[4],bins=200)
sns.histplot(ls,color='m',ax=axes[6],bins=200)

ls=[]
for i in range(80000):
    ans=np.random.choice(df6['hospitalization charges'],len(df6),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(7,1,6)
sns.histplot(ls,color='y',ax=axes[5],bins=200)
```

```
sns.histplot(ls,color='y',ax=axes[6],bins=200)


axes[0].set_title('severity level 0',fontsize=25)
axes[1].set_title('severity level 1',fontsize=25)
axes[2].set_title('severity level 2',fontsize=25)
axes[3].set_title('severity level 3',fontsize=25)
axes[4].set_title('severity level 4',fontsize=25)
axes[5].set_title('severity level 5',fontsize=25)
axes[6].set_title('Combined',fontsize=25)
```

Out[114]:

```
Text(0.5, 1.0, 'Combined')
```



In [ ]: