Project 1 - Healthcare Insurance Analysis

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Project Task - Week 1

```
1. Collate the files so that all the information is in one place

2. Check for missing values in the dataset

3. Find the percentage of rows that have trivial value (for example, ?), and de
lete such rows if they do not contain       significant information

4. Use the necessary transformation methods to deal with the nominal and ordina
l categorical variables in the dataset
```

In [2]:

```
names = pd.read_excel('Names.xlsx')
```

In [3]:

```
medical = pd.read_csv('Medical Examinations.csv')
```

In [4]:

```
hospital = pd.read_csv('Hospitalisation details.csv')
```

In [5]:

```
names.head()
```

Out[5]:

	Customer ID	name
0	Id1	Hawks, Ms. Kelly
1	Id2	Lehner, Mr. Matthew D
2	Id3	Lu, Mr. Phil
3	Id4	Osborne, Ms. Kelsey
4	Id5	Kadala, Ms. Kristyn

In [6]:

```
names.shape
```

Out[6]:

```
(2335, 2)
```

In [7]:

```
medical.head()
```

Out[7]:

	Customer ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker
0	Id1	47.410	7.47	No	No	No	No major surgery	yes
1	Id2	30.360	5.77	No	No	No	No major surgery	yes
2	Id3	34.485	11.87	yes	No	No	2	yes
3	Id4	38.095	6.05	No	No	No	No major surgery	yes
4	Id5	35.530	5.45	No	No	No	No major surgery	yes

In [8]:

```
medical.shape
```

Out[8]:

```
(2335, 8)
```

In [9]:

```
hospital.head()
```

Out[9]:

	Customer ID	year	month	date	children	charges	Hospital tier	City tier	State ID
0	Id2335	1992	Jul	9	0	563.84	tier - 2	tier - 3	R1013
1	Id2334	1992	Nov	30	0	570.62	tier - 2	tier - 1	R1013
2	Id2333	1993	Jun	30	0	600.00	tier - 2	tier - 1	R1013
3	Id2332	1992	Sep	13	0	604.54	tier - 3	tier - 3	R1013
4	Id2331	1998	Jul	27	0	637.26	tier - 3	tier - 3	R1013

In [10]:

```
hospital.shape
```

Out[10]:

```
(2343, 9)
```

In [11]:

```
merge1 = pd.merge(names,medical,on='Customer ID',how='inner')
```

In [12]:

```
merge1
```

Out[12]:

	Customer ID	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSu
0	Id1	Hawks, Ms. Kelly	47.410	7.47	No	No	No	No major s
1	Id2	Lehner, Mr. Matthew D	30.360	5.77	No	No	No	No major s
2	Id3	Lu, Mr. Phil	34.485	11.87	yes	No	No	
3	Id4	Osborne, Ms. Kelsey	38.095	6.05	No	No	No	No major s
4	Id5	Kadala, Ms. Kristyn	35.530	5.45	No	No	No	No major s
...	...	...	...	...	...	...	...	
2330	Id2331	Brietzke, Mr. Jordan	22.340	5.57	No	No	No	
2331	Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	No	No	No	
2332	Id2333	Albano, Ms. Julie	16.470	6.35	No	No	Yes	
2333	Id2334	Rosendahl, Mr. Evan P	17.600	4.39	No	No	No	
2334	Id2335	German, Mr. Aaron K	17.580	4.51	No	No	No	

2335 rows × 9 columns

In [13]:

```
df = pd.merge(merge1,hospital,on='Customer ID',how='inner')
```

In [14]:

```
df
```

Out[14]:

	Customer ID	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSu
0	Id1	Hawks, Ms. Kelly	47.410	7.47	No	No	No	No major s
1	Id2	Lehner, Mr. Matthew D	30.360	5.77	No	No	No	No major s
2	Id3	Lu, Mr. Phil	34.485	11.87	yes	No	No	
3	Id4	Osborne, Ms. Kelsey	38.095	6.05	No	No	No	No major s
4	Id5	Kadala, Ms. Kristyn	35.530	5.45	No	No	No	No major s
...	...	...	...	...	...	...	...	
2330	Id2331	Brietzke, Mr. Jordan	22.340	5.57	No	No	No	
2331	Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	No	No	No	
2332	Id2333	Albano, Ms. Julie	16.470	6.35	No	No	Yes	
2333	Id2334	Rosendahl, Mr. Evan P	17.600	4.39	No	No	No	
2334	Id2335	German, Mr. Aaron K	17.580	4.51	No	No	No	

2335 rows × 17 columns

In [15]:

```
df.isnull().sum()
```

Out[15]:

```
Customer ID               0
name                      0
BMI                       0
HBA1C                     0
Heart Issues              0
Any Transplants           0
Cancer history            0
NumberOfMajorSurgeries    0
smoker                    0
year                      0
month                     0
date                      0
children                  0
charges                   0
Hospital tier             0
City tier                 0
State ID                  0
dtype: int64
```

In [16]:

```
df.shape
```

Out[16]:

```
(2335, 17)
```

In [17]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2335 entries, 0 to 2334
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Customer ID              2335 non-null   object 
 1   name                     2335 non-null   object 
 2   BMI                      2335 non-null   float64
 3   HBA1C                    2335 non-null   float64
 4   Heart Issues             2335 non-null   object 
 5   Any Transplants          2335 non-null   object 
 6   Cancer history           2335 non-null   object 
 7   NumberOfMajorSurgeries   2335 non-null   object 
 8   smoker                   2335 non-null   object 
 9   year                     2335 non-null   object 
 10  month                    2335 non-null   object 
 11  date                     2335 non-null   int64  
 12  children                 2335 non-null   int64  
 13  charges                  2335 non-null   float64
 14  Hospital tier            2335 non-null   object 
 15  City tier                2335 non-null   object 
 16  State ID                 2335 non-null   object 
dtypes: float64(3), int64(2), object(12)
memory usage: 328.4+ KB
```

In [18]:

```
df.dtypes
```

Out[18]:

```
Customer ID                object
name                       object
BMI                       float64
HBA1C                     float64
Heart Issues               object
Any Transplants            object
Cancer history             object
NumberOfMajorSurgeries     object
smoker                     object
year                       object
month                      object
date                        int64
children                    int64
charges                   float64
Hospital tier              object
City tier                  object
State ID                   object
dtype: object
```

In [19]:

```
df.duplicated().sum()
```

Out[19]:

```
0
```

In [20]:

```
df.describe()
```

Out[20]:

	BMI	HBA1C	date	children	charges
count	2335.000000	2335.000000	2335.000000	2335.000000	2335.000000
mean	30.972649	6.578998	15.563597	1.025696	13529.918034
std	8.742095	2.228731	8.720508	1.234754	11898.654299
min	15.010000	4.000000	1.000000	0.000000	563.840000
25%	24.600000	4.900000	8.000000	0.000000	5084.010000
50%	30.400000	5.810000	15.000000	0.000000	9630.910000
75%	36.300000	7.955000	23.000000	2.000000	16912.295000
max	55.050000	12.000000	30.000000	5.000000	63770.430000

In [21]:

```
df.columns
```

Out[21]:

```
Index(['Customer ID', 'name', 'BMI', 'HBA1C', 'Heart Issues',
       'Any Transplants', 'Cancer history', 'NumberOfMajorSurgeries', 'smo
ker',
       'year', 'month', 'date', 'children', 'charges', 'Hospital tier',
       'City tier', 'State ID'],
      dtype='object')
```

In [22]:

```
df.set_index('Customer ID',inplace=True)
```

In [23]:

```
df
```

Out[23]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Customer ID							
Id1	Hawks, Ms. Kelly	47.410	7.47	No	No	No	No major surgery
Id2	Lehner, Mr. Matthew D	30.360	5.77	No	No	No	No major surgery
Id3	Lu, Mr. Phil	34.485	11.87	yes	No	No	2
Id4	Osborne, Ms. Kelsey	38.095	6.05	No	No	No	No major surgery
Id5	Kadala, Ms. Kristyn	35.530	5.45	No	No	No	No major surgery
...	...	...	...	...	...	...	...
Id2331	Brietzke, Mr. Jordan	22.340	5.57	No	No	No	1
Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	No	No	No	1
Id2333	Albano, Ms. Julie	16.470	6.35	No	No	Yes	1
Id2334	Rosendahl, Mr. Evan P	17.600	4.39	No	No	No	1
Id2335	German, Mr. Aaron K	17.580	4.51	No	No	No	1

2335 rows × 16 columns

In [24]:

```
df[df == '?'].any()
```

Out[24]:

```
name                      False
BMI                       False
HBA1C                     False
Heart Issues              False
Any Transplants           False
Cancer history            False
NumberOfMajorSurgeries    False
smoker                     True
year                       True
month                      True
date                      False
children                  False
charges                   False
Hospital tier              True
City tier                  True
State ID                   True
dtype: bool
```

In [25]:

```
df.loc[df['smoker']=='?']
```

Out[25]:

Customer ID	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Id560	Pearlman, Mr. Oz	23.980	4.90	No	No	No	No major surgery
Id635	Bruns, Mr. Zachary T	25.175	4.96	No	yes	No	1

In [26]:

```
df.loc[df['year']=='?']
```

Out[26]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	sm
Customer ID								
Id1286	Ainsley, Ms. Katie M.	29.37	8.01	yes	No	No	1	
Id1289	Levine, Ms. Annie J.	24.32	11.56	yes	No	No	1	

In [27]:

```
df.loc[df['month']=='?']
```

Out[27]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	s
Customer ID								
Id3	Lu, Mr. Phil	34.485	11.87	yes	No	No	2	
Id2318	Gagnon, Ms. Candice M	18.820	5.51	yes	No	No	No major surgery	
Id2322	Street, Ms. Holly	21.380	8.01	No	No	No	No major surgery	

In [28]:

```
df.loc[df['Hospital tier']=='?']
```

Out[28]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	sn
Customer ID								
Id2324	Duffy, Ms. Meghan K	22.24	5.04	No	No	No	No major surgery	

In [29]:

```
df.loc[df['City tier']=='?']
```

Out[29]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	sı
Customer ID								
Id2318	Gagnon, Ms. Candice M	18.82	5.51	yes	No	No	No major surgery	

In [30]:

```
df.loc[df['State ID']=='?']
```

Out[30]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	
Customer ID								
Id170	Torphy, Mr. Bobby	37.620	6.32	yes	yes	No	2	
Id1793	Capriolo, Mr. Michael	18.905	4.91	yes	No	No	1	

These are the Customers ID values that contains trivial value ['Id560','Id635','Id1286','Id1289','Id3','Id2318','Id2322','Id2324','Id170','Id1793']

In [31]:

```
df1 = df.copy()
```

In [32]:

```
df.drop(['Id560','Id635','Id1286','Id1289','Id3','Id2318','Id2322','Id2324','Id170','Id1
```

In [33]:

```
df.shape
```

Out[33]:

(2325, 16)

In [34]:

```
df[df == '?'].any()
```

Out[34]:

```
name                     False
BMI                      False
HBA1C                    False
Heart Issues             False
Any Transplants          False
Cancer history           False
NumberOfMajorSurgeries   False
smoker                   False
year                     False
month                    False
date                     False
children                 False
charges                  False
Hospital tier            False
City tier                False
State ID                 False
dtype: bool
```

In [35]:

```
new_data = df.copy()
```

In [36]:

```
new_data.to_csv('Project 1 Sql.csv')
```

In [37]:

```
df.dtypes
```

Out[37]:

```
name                      object
BMI                      float64
HBA1C                    float64
Heart Issues              object
Any Transplants           object
Cancer history            object
NumberOfMajorSurgeries    object
smoker                    object
year                      object
month                     object
date                       int64
children                   int64
charges                  float64
Hospital tier             object
City tier                 object
State ID                  object
dtype: object
```

In [38]:

```
df['Heart Issues'].unique()
```

Out[38]:

```
array(['No', 'yes'], dtype=object)
```

In [39]:

```
df['Heart Issues'] = df['Heart Issues'].replace('No','0')
df['Heart Issues'] = df['Heart Issues'].replace('yes','1')


df['Heart Issues'] = df['Heart Issues'].astype(int)


df.dtypes
```

Out[39]:

```
name                        object
BMI                        float64
HBA1C                      float64
Heart Issues                 int32
Any Transplants             object
Cancer history              object
NumberOfMajorSurgeries      object
smoker                      object
year                        object
month                       object
date                         int64
children                     int64
charges                    float64
Hospital tier               object
City tier                   object
State ID                    object
dtype: object
```

In [40]:

```
df['Any Transplants'].unique()
```

Out[40]:

```
array(['No', 'yes'], dtype=object)
```

In [41]:

```
df['Any Transplants'] = df['Any Transplants'].replace('No','0')
df['Any Transplants'] = df['Any Transplants'].replace('yes','1')


df['Any Transplants'] = df['Any Transplants'].astype(int)


df.dtypes
```

Out[41]:

```
name                       object
BMI                       float64
HBA1C                     float64
Heart Issues                int32
Any Transplants             int32
Cancer history             object
NumberOfMajorSurgeries     object
smoker                     object
year                       object
month                      object
date                        int64
children                    int64
charges                   float64
Hospital tier              object
City tier                  object
State ID                   object
dtype: object
```

In [42]:

```
df['Cancer history'].unique()
```

Out[42]:

```
array(['No', 'Yes'], dtype=object)
```

In [43]:

```
df['Cancer history'] = df['Cancer history'].replace('No','0')
df['Cancer history'] = df['Cancer history'].replace('Yes','1')


df['Cancer history'] = df['Cancer history'].astype(int)


df.dtypes
```

Out[43]:

```
name                        object
BMI                        float64
HBA1C                      float64
Heart Issues                 int32
Any Transplants              int32
Cancer history               int32
NumberOfMajorSurgeries      object
smoker                      object
year                        object
month                       object
date                         int64
children                     int64
charges                    float64
Hospital tier               object
City tier                   object
State ID                    object
dtype: object
```

In [44]:

```
df['smoker'].unique()
```

Out[44]:

```
array(['yes', 'No'], dtype=object)
```

In [45]:

```
df['smoker'] = df['smoker'].replace('No','0')
df['smoker'] = df['smoker'].replace('yes','1')


df['smoker'] = df['smoker'].astype(int)


df.dtypes
```

Out[45]:

```
name                        object
BMI                        float64
HBA1C                      float64
Heart Issues                 int32
Any Transplants              int32
Cancer history               int32
NumberOfMajorSurgeries      object
smoker                       int32
year                        object
month                       object
date                         int64
children                     int64
charges                    float64
Hospital tier               object
City tier                   object
State ID                    object
dtype: object
```

In [46]:

```
df['NumberOfMajorSurgeries'].unique()
```

Out[46]:

```
array(['No major surgery', '3', '1', '2'], dtype=object)
```

In [47]:

```
df['NumberOfMajorSurgeries'] = df['NumberOfMajorSurgeries'].replace('No major surgery','

df['NumberOfMajorSurgeries'] = df['NumberOfMajorSurgeries'].astype(int)

df.dtypes
```

Out[47]:

```
name                        object
BMI                        float64
HBA1C                      float64
Heart Issues                 int32
Any Transplants              int32
Cancer history               int32
NumberOfMajorSurgeries       int32
smoker                       int32
year                        object
month                       object
date                         int64
children                     int64
charges                    float64
Hospital tier               object
City tier                   object
State ID                    object
dtype: object
```

In [48]:

```
df['State ID'].unique()
```

Out[48]:

```
array(['R1013', 'R1024', 'R1012', 'R1011', 'R1016', 'R1015', 'R1017',
       'R1014', 'R1023', 'R1019', 'R1018', 'R1026', 'R1022', 'R1021',
       'R1025', 'R1020'], dtype=object)
```

In [49]:

```
dum = pd.get_dummies(df['State ID'])
```

In [50]:

```
dum
```

Out[50]:

Customer ID	R1011	R1012	R1013	R1014	R1015	R1016	R1017	R1018	R1019	R1020	R102
Id1	0	0	1	0	0	0	0	0	0	0	
Id2	0	0	1	0	0	0	0	0	0	0	
Id4	0	0	0	0	0	0	0	0	0	0	
Id5	0	1	0	0	0	0	0	0	0	0	
Id6	1	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	.
Id2331	0	0	1	0	0	0	0	0	0	0	
Id2332	0	0	1	0	0	0	0	0	0	0	
Id2333	0	0	1	0	0	0	0	0	0	0	
Id2334	0	0	1	0	0	0	0	0	0	0	
Id2335	0	0	1	0	0	0	0	0	0	0	

2325 rows × 16 columns

In [51]:

```
dum.drop(dum.columns[3:16],axis=1,inplace=True)
```

In [52]:

```
dum
```

Out[52]:

	R1011	R1012	R1013
Customer ID			
Id1	0	0	1
Id2	0	0	1
Id4	0	0	0
Id5	0	1	0
Id6	1	0	0
...	...	...	...
Id2331	0	0	1
Id2332	0	0	1
Id2333	0	0	1
Id2334	0	0	1
Id2335	0	0	1

2325 rows × 3 columns

In [53]:

```
master_data = df.copy()
```

In [54]:

```
df = pd.concat([df,dum],axis=1)
df
```

Out[54]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Customer ID							
Id1	Hawks, Ms. Kelly	47.410	7.47	0	0	0	0
Id2	Lehner, Mr. Matthew D	30.360	5.77	0	0	0	0
Id4	Osborne, Ms. Kelsey	38.095	6.05	0	0	0	0
Id5	Kadala, Ms. Kristyn	35.530	5.45	0	0	0	0
Id6	Baker, Mr. Russell B.	32.800	6.59	0	0	0	0
...	...	...	...	...	...	...	...
Id2331	Brietzke, Mr. Jordan	22.340	5.57	0	0	0	1
Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	0	0	0	1
Id2333	Albano, Ms. Julie	16.470	6.35	0	0	1	1
Id2334	Rosendahl, Mr. Evan P	17.600	4.39	0	0	0	1
Id2335	German, Mr. Aaron K	17.580	4.51	0	0	0	1

2325 rows × 19 columns

In [55]:

```
df['year'] = df['year'].astype(int)
```

In [56]:

```
df['Age'] = 2023 - df['year']
df
```

Out[56]:

Customer ID	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Id1	Hawks, Ms. Kelly	47.410	7.47	0	0	0	0
Id2	Lehner, Mr. Matthew D	30.360	5.77	0	0	0	0
Id4	Osborne, Ms. Kelsey	38.095	6.05	0	0	0	0
Id5	Kadala, Ms. Kristyn	35.530	5.45	0	0	0	0
Id6	Baker, Mr. Russell B.	32.800	6.59	0	0	0	0
...	...	...	...	...	...	...	...
Id2331	Brietzke, Mr. Jordan	22.340	5.57	0	0	0	1
Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	0	0	0	1
Id2333	Albano, Ms. Julie	16.470	6.35	0	0	1	1
Id2334	Rosendahl, Mr. Evan P	17.600	4.39	0	0	0	1
Id2335	German, Mr. Aaron K	17.580	4.51	0	0	0	1

2325 rows × 20 columns

In [57]:

```
df['name'].str.contains('Mr.').sum()
```

Out[57]:

1302

In [58]:

```
df['name'].str.contains('Mrs.').sum()
```

Out[58]:

142

In [59]:

```
df['name'].str.contains('Ms.').sum()
```

Out[59]:

```
1023
```

In [60]:

```
df['gender'] = df['name'].str.extract('(Mr.|Mrs.|Ms.)')
```

In [61]:

```
df['gender'].value_counts()
```

Out[61]:

```
Mr.     1159
Ms.     1023
Mrs      142
Mro        1
Name: gender, dtype: int64
```

In [62]:

```
df['gender'] = df['gender'].replace('Mr.','Male')
df['gender'] = df['gender'].replace('Mro','Male')
df['gender'] = df['gender'].replace('Ms.','Female')
df['gender'] = df['gender'].replace('Mrs','Female')
```

In [63]:

```
df['gender'].value_counts()
```

Out[63]:

```
Female    1165
Male      1160
Name: gender, dtype: int64
```

In [64]:

```
df.head()
```

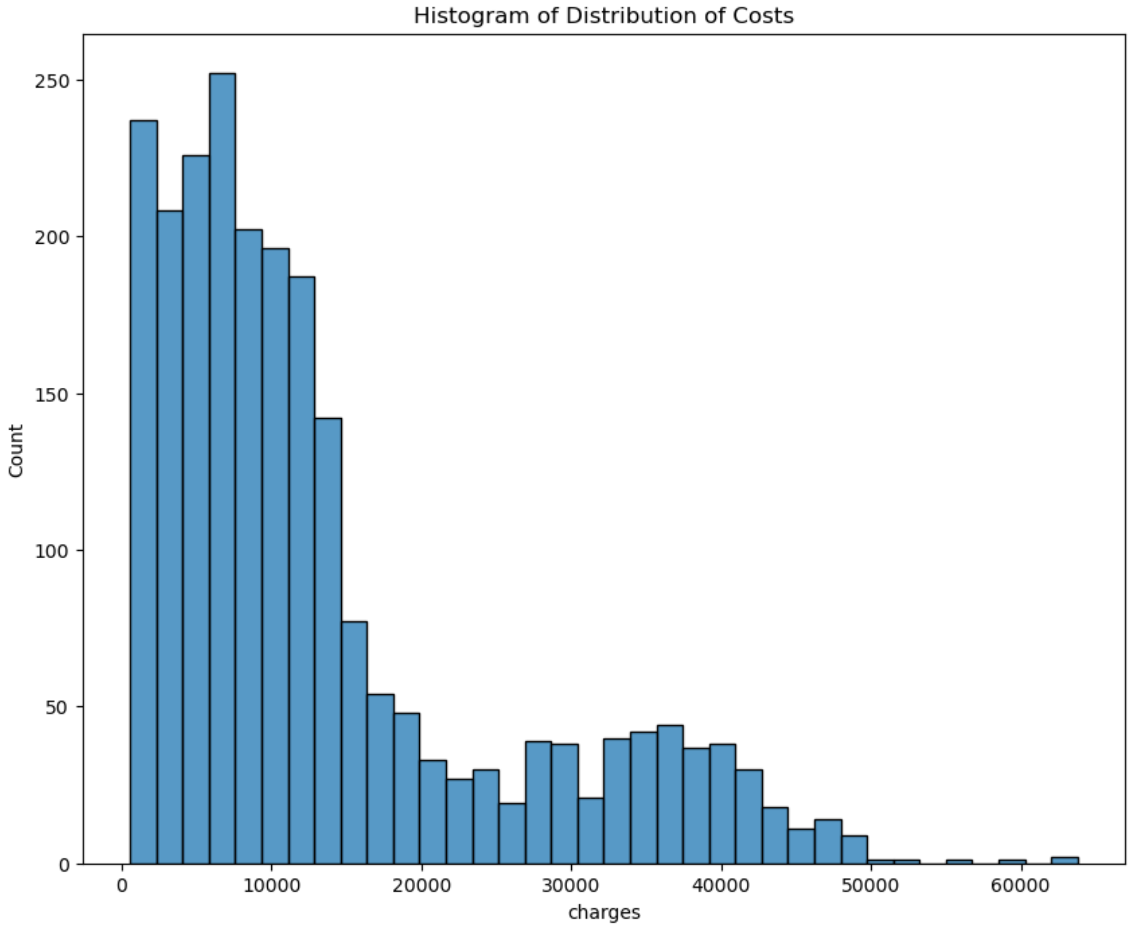
Out[64]:

Customer ID	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Id1	Hawks, Ms. Kelly	47.410	7.47	0	0	0	0
Id2	Lehner, Mr. Matthew D	30.360	5.77	0	0	0	0
Id4	Osborne, Ms. Kelsey	38.095	6.05	0	0	0	0
Id5	Kadala, Ms. Kristyn	35.530	5.45	0	0	0	0
Id6	Baker, Mr. Russell B.	32.800	6.59	0	0	0	0

5 rows × 21 columns

In [65]:

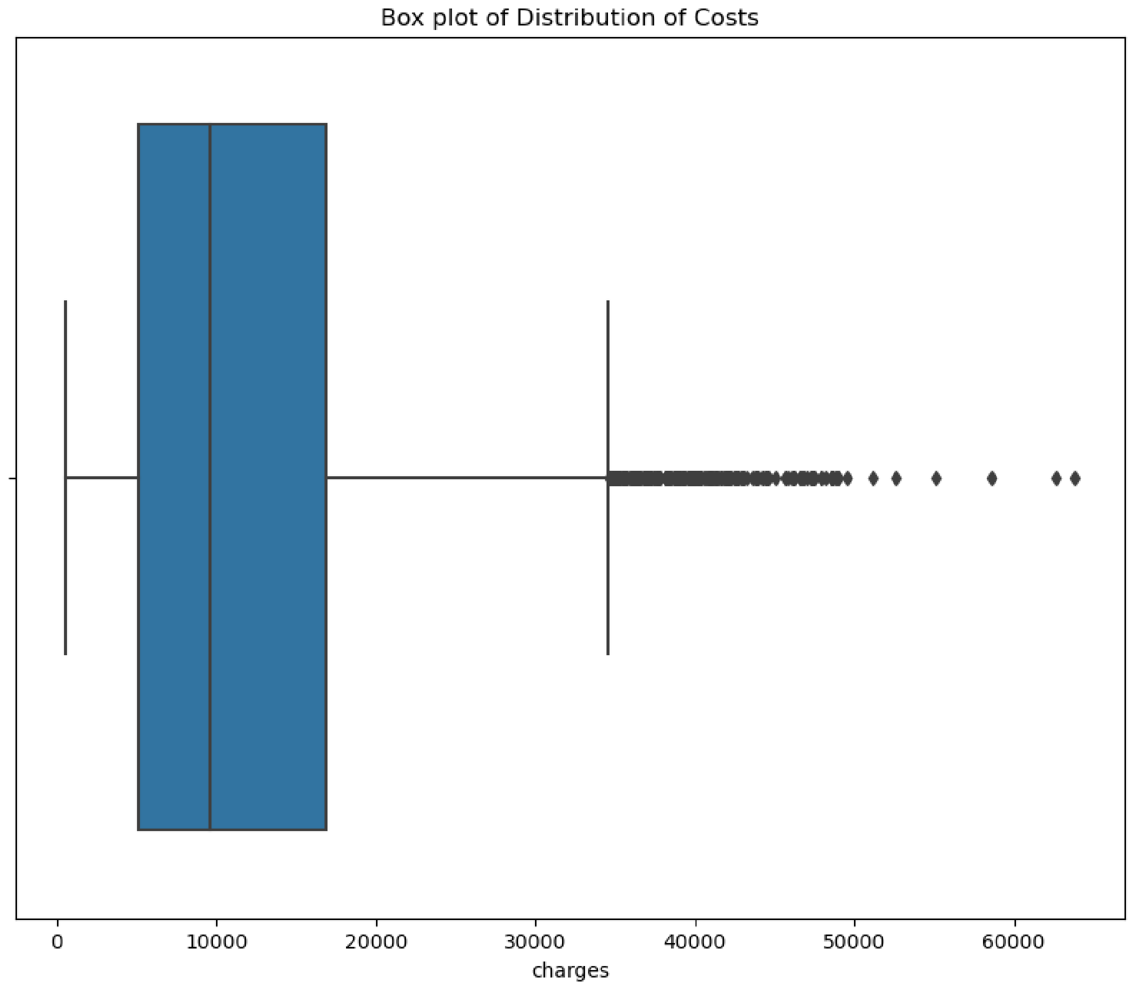
```
plt.figure(figsize=(10,8))
sns.histplot(df['charges'])
plt.title('Histogram of Distribution of Costs')
plt.show()
```



In [66]:

```
plt.figure(figsize=(10,8))
sns.boxplot(df['charges'])
plt.title('Box plot of Distribution of Costs')
plt.show()
```

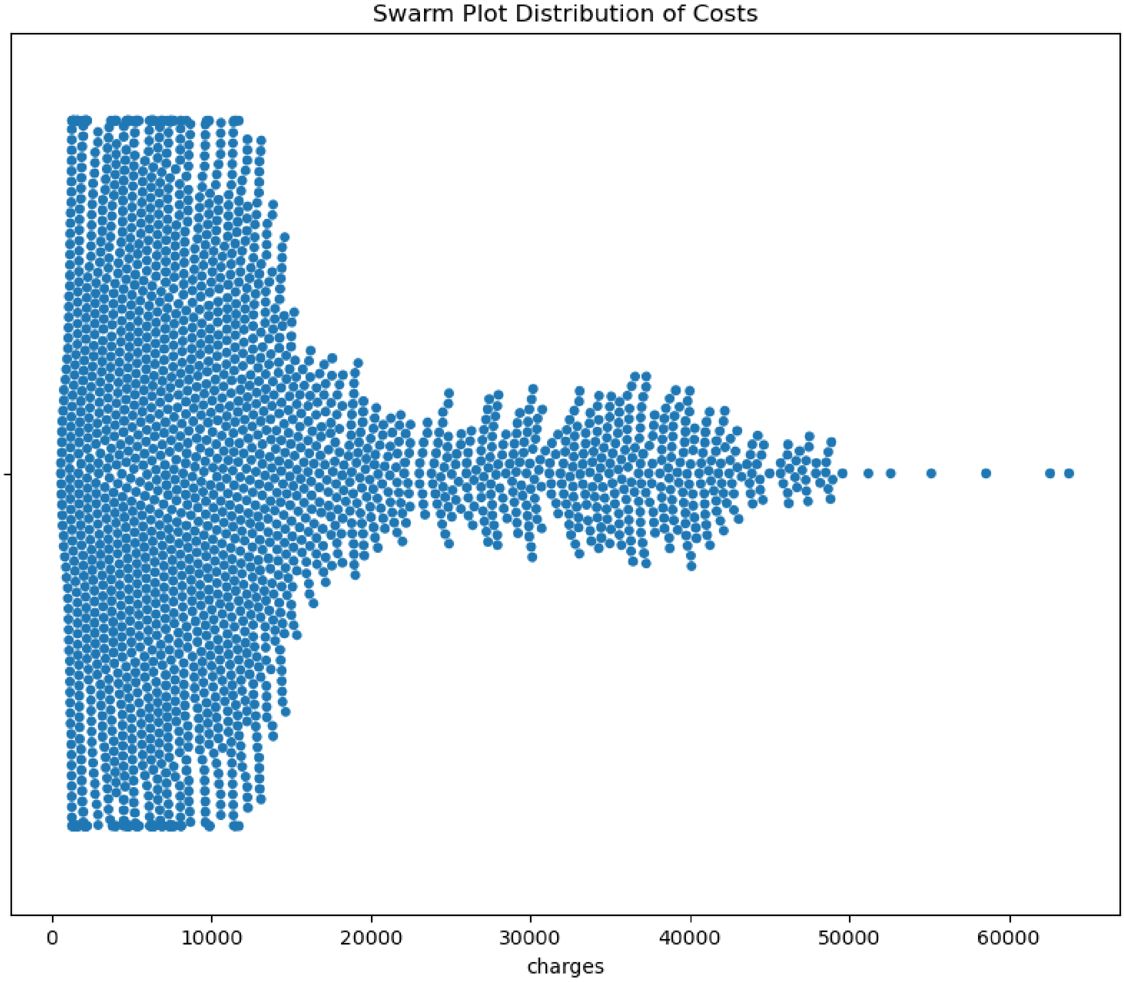
```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



In [67]:

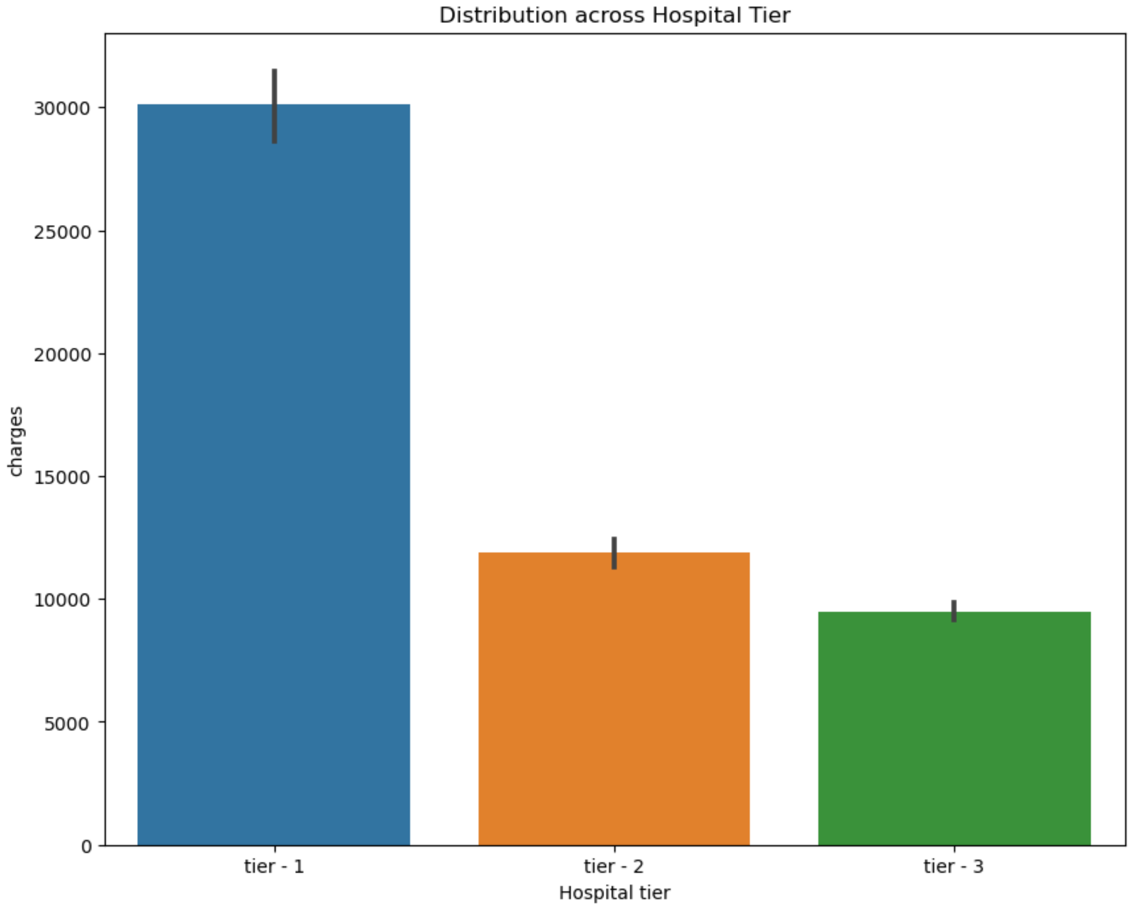
```
plt.figure(figsize=(10,8))
sns.swarmplot(df['charges'])
plt.title('Swarm Plot Distribution of Costs')
plt.show()
```

```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\categorical.py:1296: U
serWarning: 7.1% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



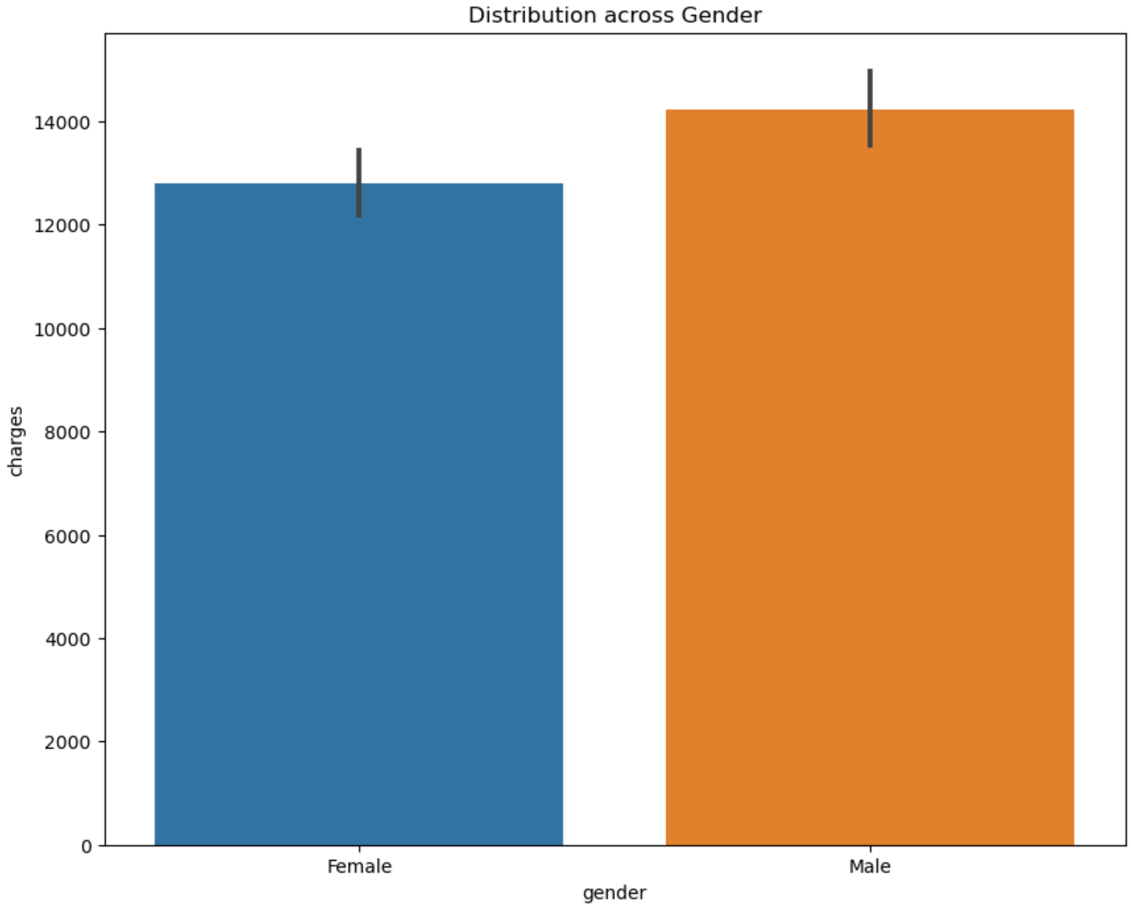
In [68]:

```
plt.figure(figsize=(10,8))
sns.barplot(x=df['Hospital tier'],y=df['charges'])
plt.title('Distribution across Hospital Tier')
plt.show()
```



In [69]:

```
plt.figure(figsize=(10,8))
sns.barplot(x=df['gender'],y=df['charges'])
plt.title('Distribution across Gender')
plt.show()
```



In [70]:

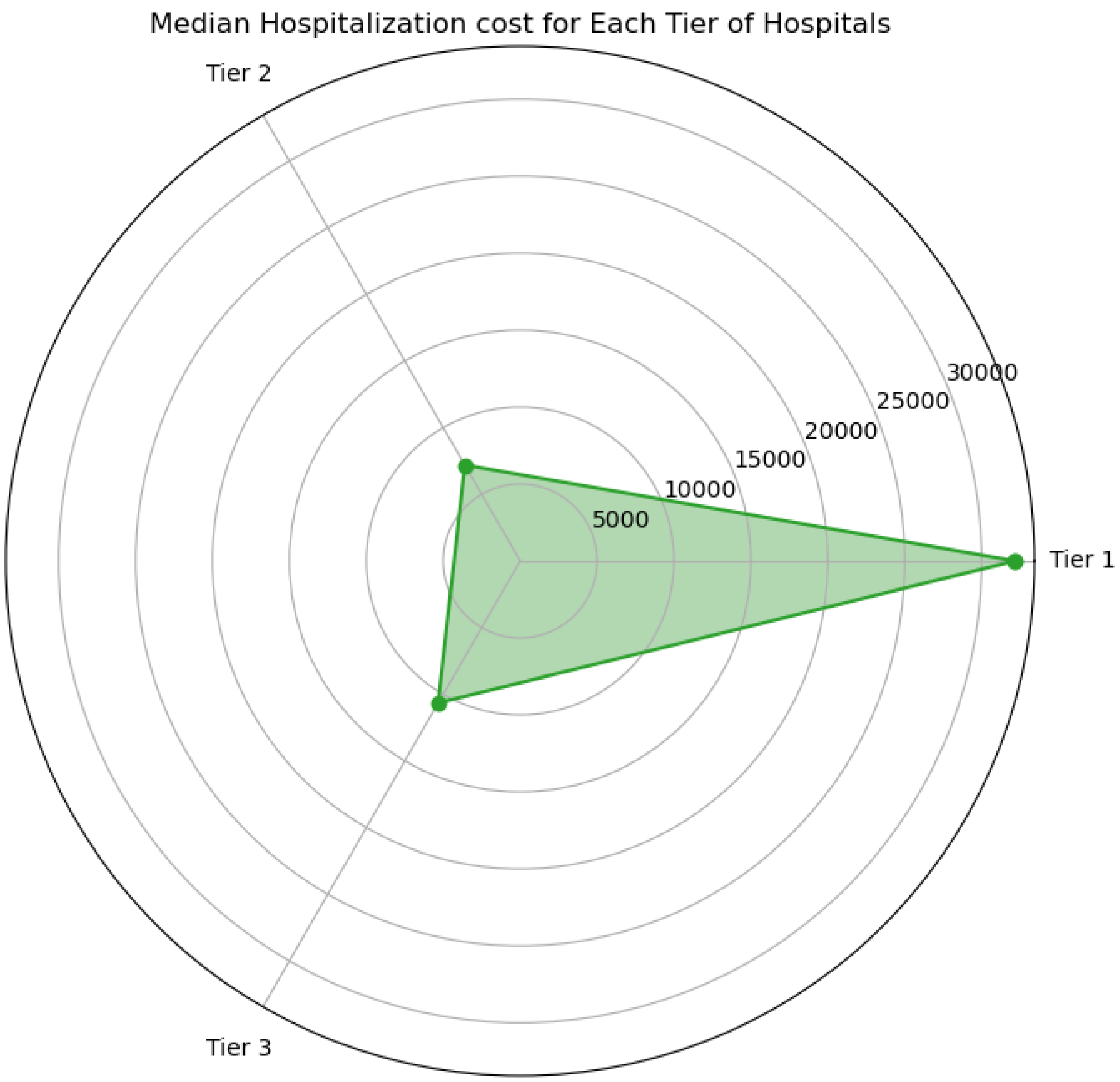
```
pd.DataFrame(df.groupby('Hospital tier')['charges'].median())
```

Out[70]:

Hospital tier	charges
tier - 1	32097.435
tier - 2	7168.760
tier - 3	10676.830

In [71]:

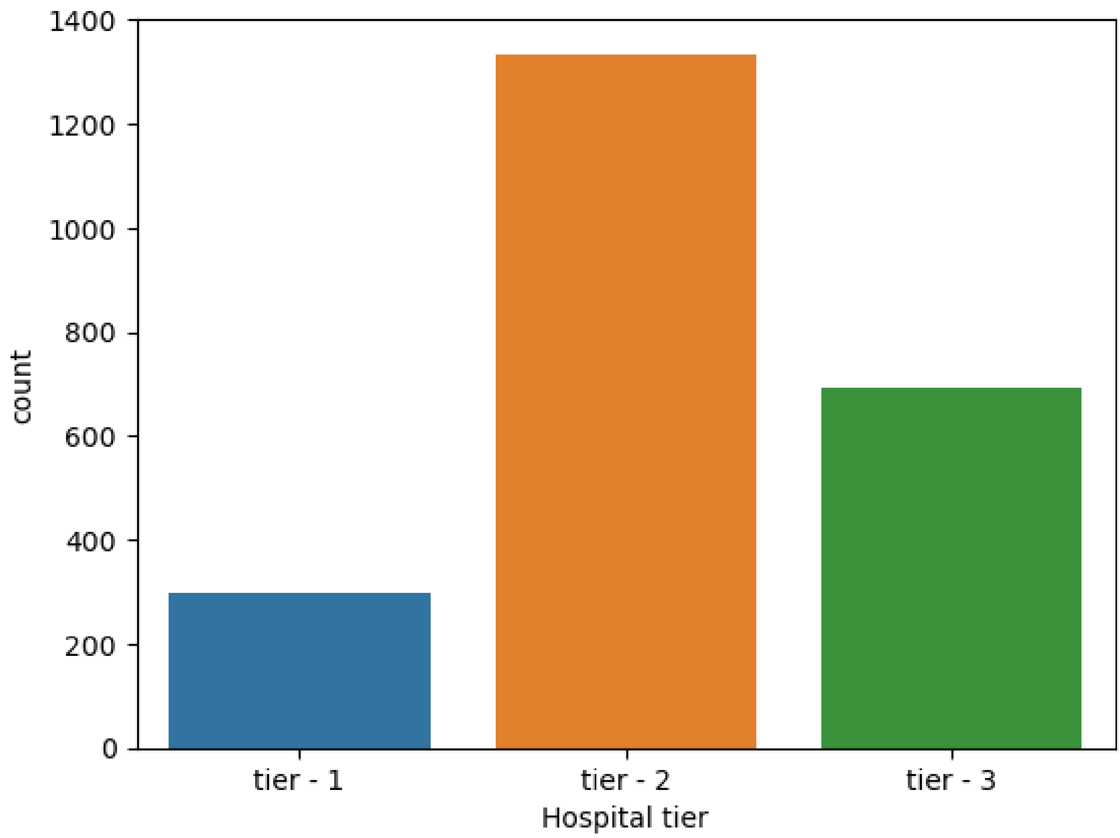
```
radar = pd.DataFrame({'Tier': ['Tier 1', 'Tier 2', 'Tier 3'],
                      'Median': [32192.760, 7201.700, 10646.625]})
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, polar=True)


theta = np.arange(len(radar) + 1) / float(len(radar)) * 2 * np.pi


values = radar['Median'].values
values = np.append(values, values[0])


l1, = ax.plot(theta, values, color='C2', marker='o')
plt.xticks(theta[:-1], radar['Tier'])
ax.tick_params(pad=10)
ax.fill(theta, values, 'green', alpha=0.3)


plt.title('Median Hospitalization cost for Each Tier of Hospitals')
plt.show()
```



In [72]:

```
sns.countplot(df['Hospital tier'])
plt.show()
```

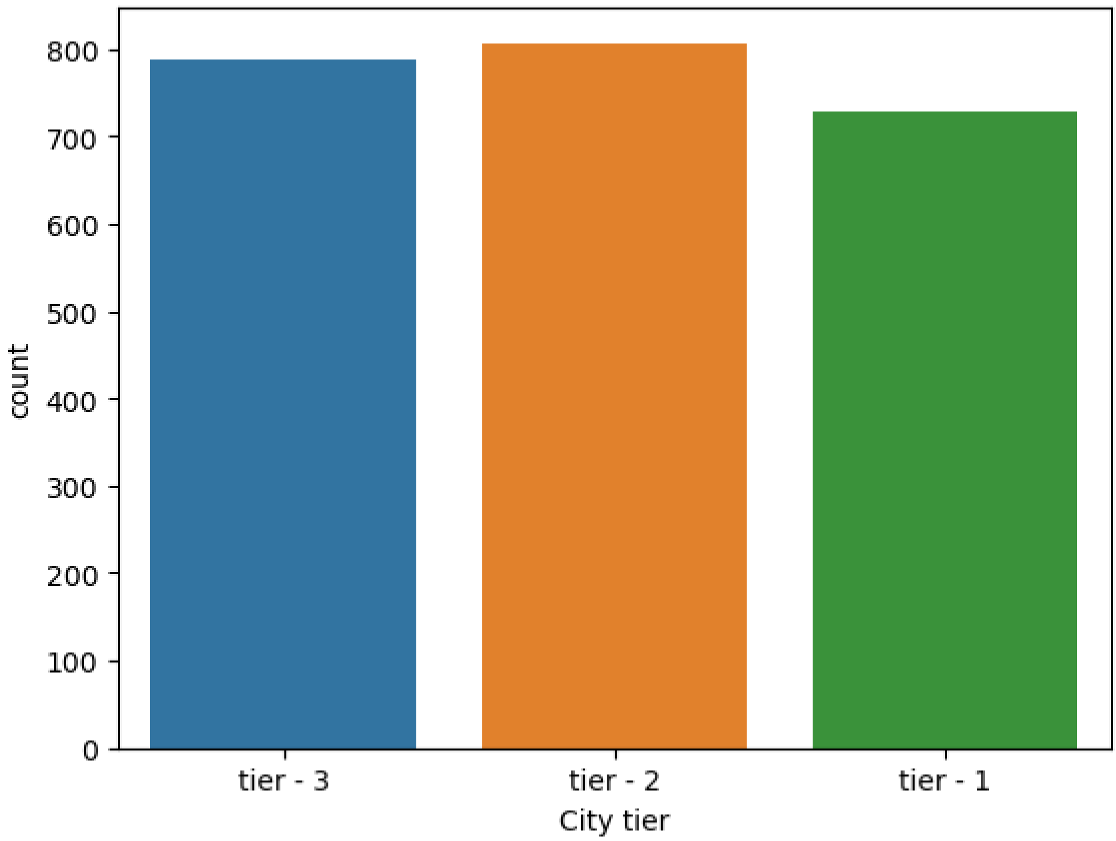
```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



In [73]:

```
sns.countplot(df['City tier'])
plt.show()
```

```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



In [74]:

```
df['Hospital tier'].value_counts()
```

Out[74]:

```
tier - 2    1334
tier - 3     691
tier - 1     300
Name: Hospital tier, dtype: int64
```

In [75]:

```
freq_table = pd.crosstab(index=df['Hospital tier'],columns=df['City tier'])
```

In [76]:

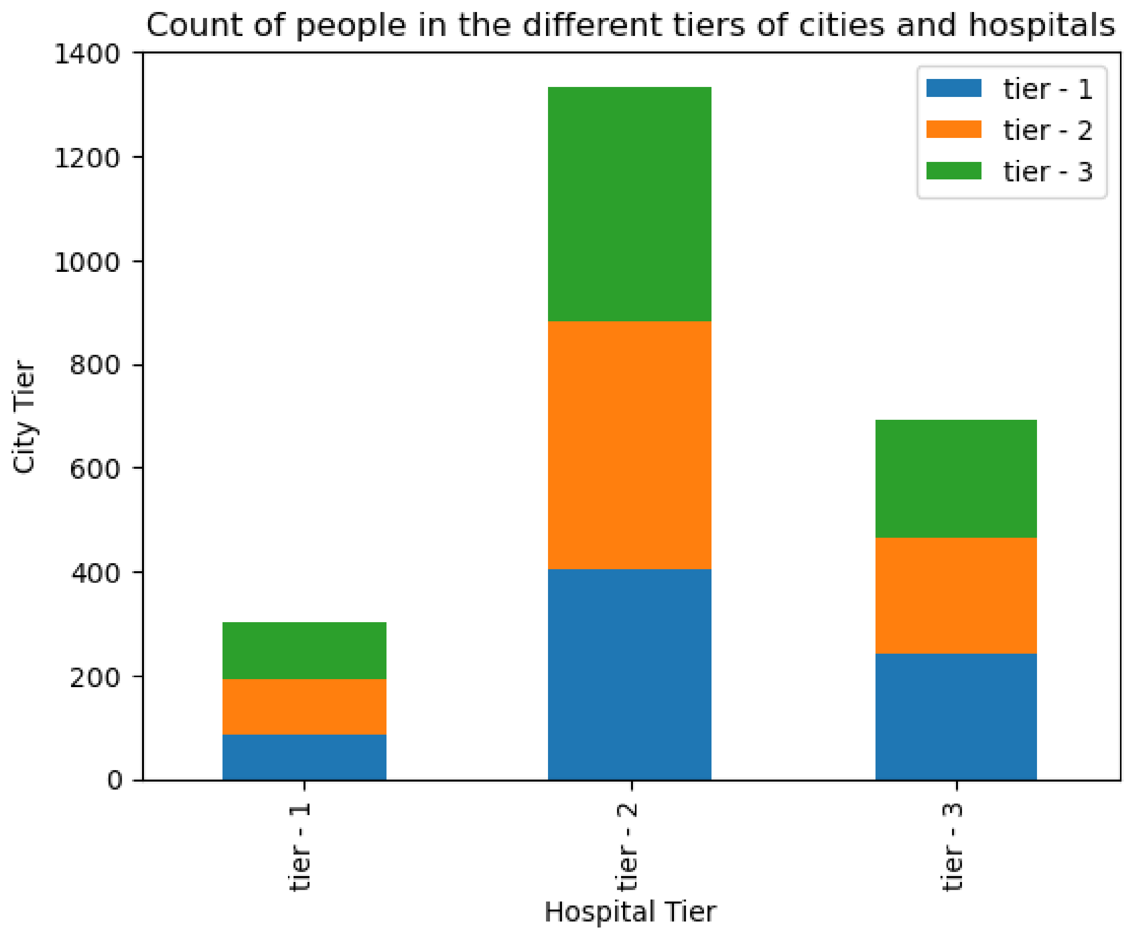
```
freq_table
```

Out[76]:

City tier	tier - 1	tier - 2	tier - 3
Hospital tier			
tier - 1	85	106	109
tier - 2	403	479	452
tier - 3	241	222	228

In [77]:

```
freq_table.plot(kind='bar',stacked=True)
plt.xlabel('Hospital Tier')
plt.ylabel('City Tier')
plt.title('Count of people in the different tiers of cities and hospitals')
plt.legend()
plt.show()
```



In [78]:

```
df
```

Out[78]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Customer ID							
Id1	Hawks, Ms. Kelly	47.410	7.47	0	0	0	0
Id2	Lehner, Mr. Matthew D	30.360	5.77	0	0	0	0
Id4	Osborne, Ms. Kelsey	38.095	6.05	0	0	0	0
Id5	Kadala, Ms. Kristyn	35.530	5.45	0	0	0	0
Id6	Baker, Mr. Russell B.	32.800	6.59	0	0	0	0
...	...	...	...	...	...	...	...
Id2331	Brietzke, Mr. Jordan	22.340	5.57	0	0	0	1
Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	0	0	0	1
Id2333	Albano, Ms. Julie	16.470	6.35	0	0	1	1
Id2334	Rosendahl, Mr. Evan P	17.600	4.39	0	0	0	1
Id2335	German, Mr. Aaron K	17.580	4.51	0	0	0	1

2325 rows × 21 columns

In [79]:

```
df.to_csv('project 1.csv')
```

In [80]:

```
pd.DataFrame(df.groupby('Hospital tier')['charges'].mean())
```

Out[80]:

	charges
Hospital tier	
tier - 1	30131.995900
tier - 2	11875.883861
tier - 3	9487.456223

The average hospitalization costs for the three types of hospitals are different

In [81]:

```
pd.DataFrame(df.groupby(['Hospital tier','City tier'])['charges'].mean())
```

Out[81]:

		charges
Hospital tier	**City tier**	
	tier - 1	29160.756118
tier - 1	**tier - 2**	29014.500472
	tier - 3	31976.123394
	tier - 1	11515.412928
tier - 2	**tier - 2**	11973.655344
	tier - 3	12093.665376
	tier - 1	9812.839544
tier - 3	**tier - 2**	9283.427477
	tier - 3	9342.179912

The average hospitalization costs for the three types of cities are different

In [82]:

```
pd.DataFrame(df.groupby('smoker')['charges'].mean())
```

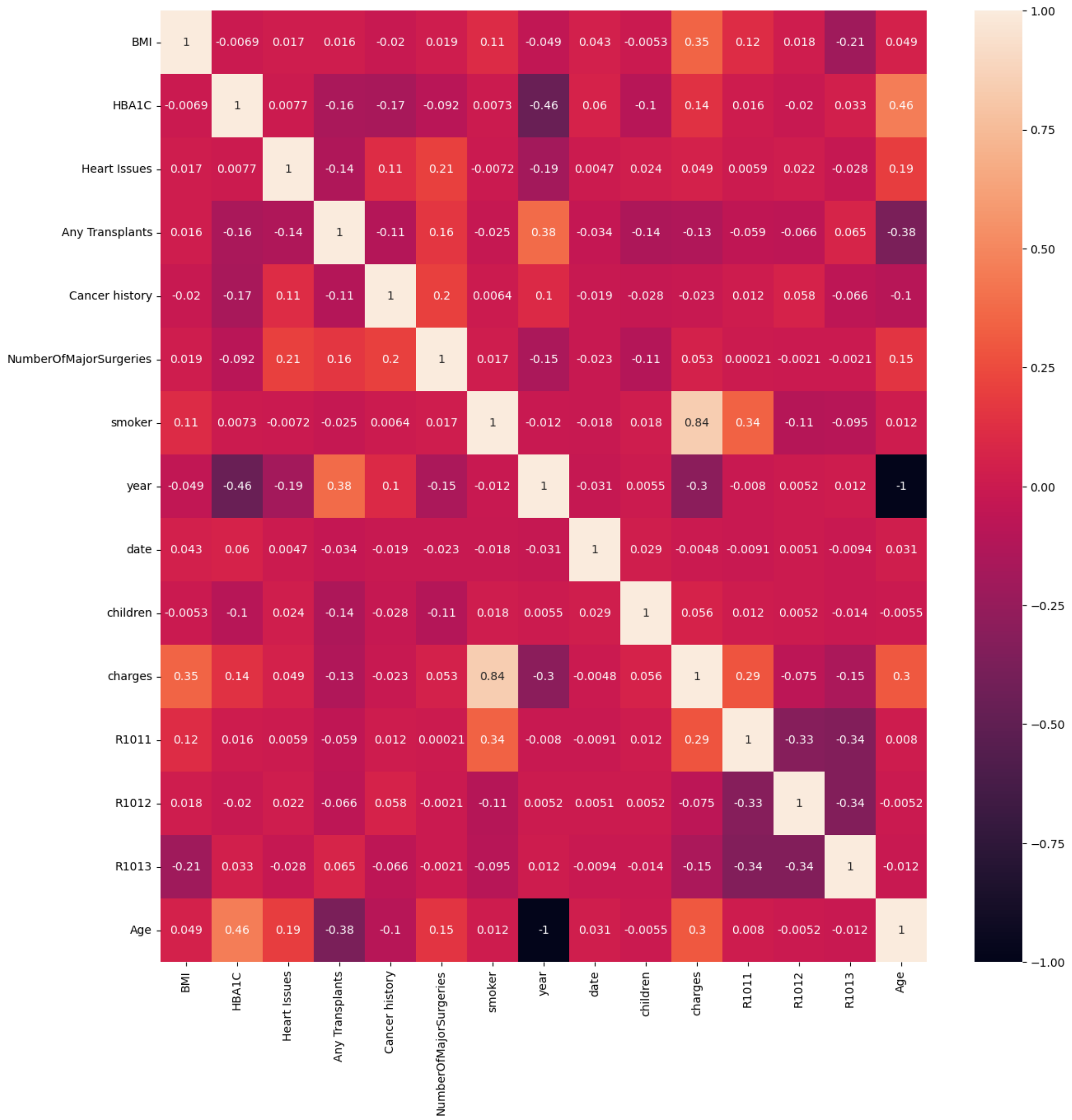
Out[82]:

	charges
smoker	
0	8409.199250
1	32866.960226

The average hospitalization cost for smokers is different from the average cost for nonsmokers

In [83]:

```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



From the heatmap the correlation between Smoker and Heart Issues is -0.00072 that is 0. So there is no correlation between these two variables and it is independent to each other

In [84]:

```
hypo_test = df[['Hospital tier','City tier','smoker','charges']]
```

In [85]:

```
hypo_test
```

Out[85]:

	Hospital tier	City tier	smoker	charges
Customer ID				
Id1	tier - 1	tier - 3	1	63770.43
Id2	tier - 2	tier - 3	1	62592.87
Id4	tier - 1	tier - 3	1	58571.07
Id5	tier - 1	tier - 2	1	55135.40
Id6	tier - 1	tier - 3	1	52590.83
...	...	...	...	...
Id2331	tier - 3	tier - 3	0	637.26
Id2332	tier - 3	tier - 3	0	604.54
Id2333	tier - 2	tier - 1	0	600.00
Id2334	tier - 2	tier - 1	0	570.62
Id2335	tier - 2	tier - 3	0	563.84

2325 rows × 4 columns

In [86]:

```
hypo_test = pd.get_dummies(hypo_test)
```

In [87]:

```
hypo_test
```

Out[87]:

Customer ID	smoker	charges	Hospital tier_tier - 1	Hospital tier_tier - 2	Hospital tier_tier - 3	City tier_tier - 1	City tier_tier - 2	City tier_tier - 3
Id1	1	63770.43	1	0	0	0	0	1
Id2	1	62592.87	0	1	0	0	0	1
Id4	1	58571.07	1	0	0	0	0	1
Id5	1	55135.40	1	0	0	0	1	0
Id6	1	52590.83	1	0	0	0	0	1
...	...	...	...	...	...	...	...	...
Id2331	0	637.26	0	0	1	0	0	1
Id2332	0	604.54	0	0	1	0	0	1
Id2333	0	600.00	0	1	0	1	0	0
Id2334	0	570.62	0	1	0	1	0	0
Id2335	0	563.84	0	1	0	0	0	1

2325 rows × 8 columns

In [88]:

```
X = hypo_test.drop('charges',axis=1)
```

In [89]:

```
X
```

Out[89]:

	smoker	Hospital tier_tier - 1	Hospital tier_tier - 2	Hospital tier_tier - 3	City tier_tier - 1	City tier_tier - 2	City tier_tier - 3
Customer ID							
Id1	1	1	0	0	0	0	1
Id2	1	0	1	0	0	0	1
Id4	1	1	0	0	0	0	1
Id5	1	1	0	0	0	1	0
Id6	1	1	0	0	0	0	1
...	...	...	...	...	...	...	...
Id2331	0	0	0	1	0	0	1
Id2332	0	0	0	1	0	0	1
Id2333	0	0	1	0	1	0	0
Id2334	0	0	1	0	1	0	0
Id2335	0	0	1	0	0	0	1

2325 rows × 7 columns

In [90]:

```
y = hypo_test['charges']
```

In [91]:

```
from sklearn.model_selection import train_test_split
```

In [92]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=42)
```

In [93]:

```
from sklearn.feature_selection import f_regression
```

In [94]:

```
f_regression(X_train,y_train)
```

Out[94]:

```
(array([4.52138923e+03, 7.42106378e+02, 4.39873200e+01, 1.00127099e+02,
        1.09672926e+01, 8.56831826e-02, 8.82332758e+00]),
 array([0.00000000e+000, 9.03140947e-138, 4.32045215e-011, 5.39588166e-02
3,
        9.45117930e-004, 7.69771237e-001, 3.01218016e-003]))
```

The p-value for Smoker - 0, Hospital Tier 1 - 9.03140947e-138, Hospital Tier 2 - 4.32045215e-011, Hospital Tier 3 - 5.39588166e-023, City Tier 1 - 9.45117930e-004, City Tier 2 - 7.69771237e-001, City Tier 3 - 3.01218016e-003

which is less than 0.05(p-value<0.05). So we reject the Null Hypothesis for Hospital Tier

Project Task - Week 2

1. Examine the correlation between predictors to identify highly correlated predictors. Use a heatmap to visualize this.
2. Develop and evaluate the final model using regression with a stochastic gradient descent optimizer. Also, ensure that you apply all the following suggestions:

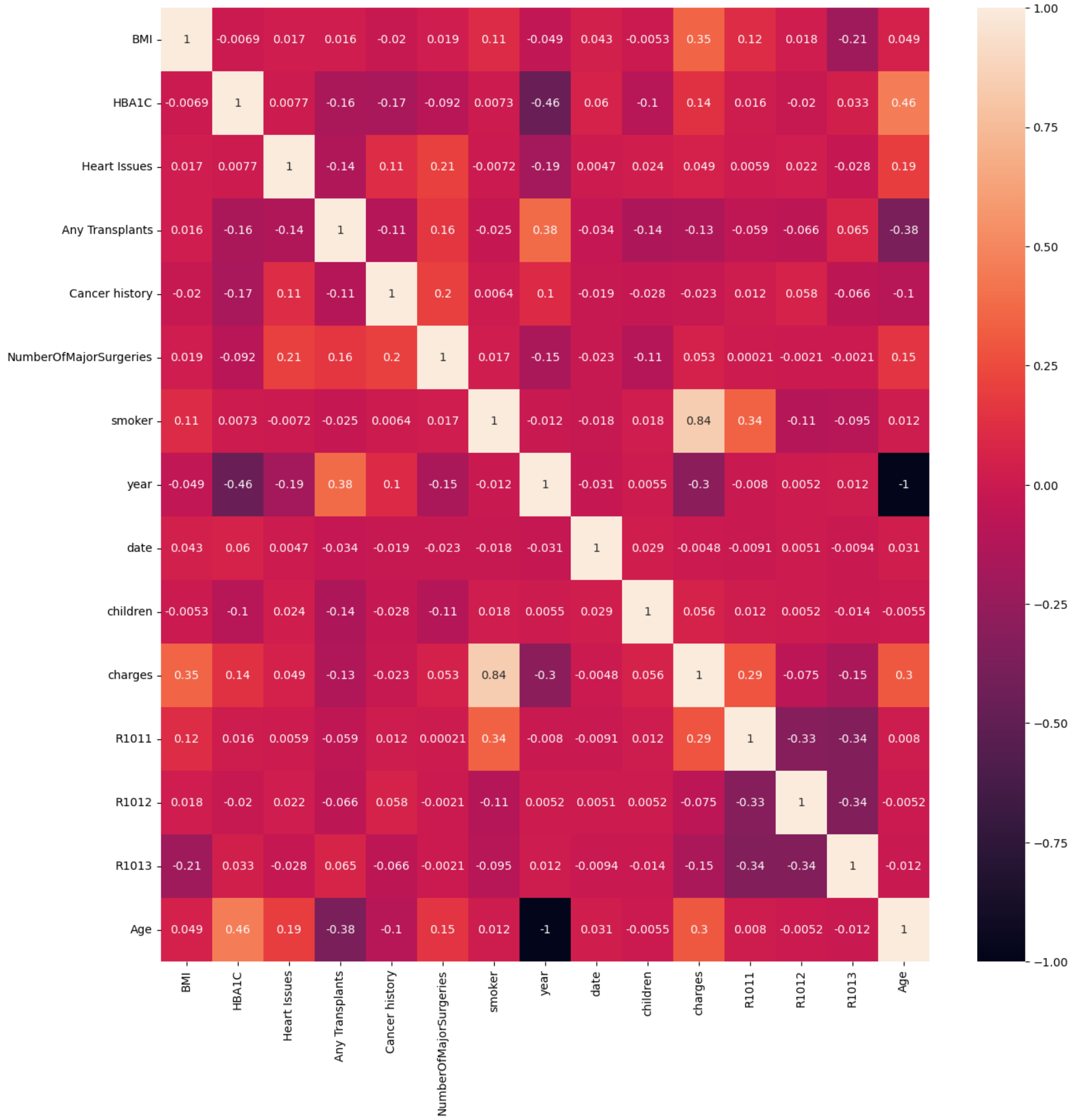
   Note:

   ```
    • Perform the stratified 5-fold cross-validation technique for model buildi
   ng and validation
    • Use standardization and hyperparameter tuning effectively
    • Use sklearn-pipelines
    • Use appropriate regularization techniques to address the bias-variance tr
   ade-off
        a. Create five folds in the data, and introduce a variable to identify
   the folds
        b. For each fold, run a for loop and ensure that 80 percent of the data
   is used to train the model and the            remaining 20 percent is used t
   o validate it in each iteration
        c. Develop five distinct models and five distinct validation scores (ro
   ot mean squared error values)
        d. Determine the variable importance scores, and identify the redundant
   variables
   ```

3. Use random forest and extreme gradient boosting for cost prediction, share your cross-validation results, and calculate the variable importance scores
4. Case scenario: Estimate the cost of hospitalization for Christopher, Ms. Jayna (her date of birth is 12/28/1988, height is 170 cm, and weight is 85 kgs). She lives in a tier-1 city and her state's State ID is R1011. She lives with her partner and two children. She was found to be nondiabetic (HbA1c = 5.8). She smokes but is otherwise healthy. She has had no transplants or major surgeries. Her father died of lung cancer. Hospitalization costs will be estimated using tier-1 hospitals.
5. Find the predicted hospitalization cost using all five models. The predicted value should be the mean of the five models' predicted values

In [95]:

```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



From the Heatmap Smoker and Charges are highly correlated

In [96]:

```
df
```

Out[96]:

	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Customer ID							
Id1	Hawks, Ms. Kelly	47.410	7.47	0	0	0	0
Id2	Lehner, Mr. Matthew D	30.360	5.77	0	0	0	0
Id4	Osborne, Ms. Kelsey	38.095	6.05	0	0	0	0
Id5	Kadala, Ms. Kristyn	35.530	5.45	0	0	0	0
Id6	Baker, Mr. Russell B.	32.800	6.59	0	0	0	0
...	...	...	...	...	...	...	...
Id2331	Brietzke, Mr. Jordan	22.340	5.57	0	0	0	1
Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	0	0	0	1
Id2333	Albano, Ms. Julie	16.470	6.35	0	0	1	1
Id2334	Rosendahl, Mr. Evan P	17.600	4.39	0	0	0	1
Id2335	German, Mr. Aaron K	17.580	4.51	0	0	0	1

2325 rows × 21 columns

In [97]:

```
data = df.copy()
```

In [98]:

```
data
```

Out[98]:

Customer ID	name	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries
Id1	Hawks, Ms. Kelly	47.410	7.47	0	0	0	0
Id2	Lehner, Mr. Matthew D	30.360	5.77	0	0	0	0
Id4	Osborne, Ms. Kelsey	38.095	6.05	0	0	0	0
Id5	Kadala, Ms. Kristyn	35.530	5.45	0	0	0	0
Id6	Baker, Mr. Russell B.	32.800	6.59	0	0	0	0
...	...	...	...	...	...	...	...
Id2331	Brietzke, Mr. Jordan	22.340	5.57	0	0	0	1
Id2332	Riveros Gonzalez, Mr. Juan D. Sr.	17.700	6.28	0	0	0	1
Id2333	Albano, Ms. Julie	16.470	6.35	0	0	1	1
Id2334	Rosendahl, Mr. Evan P	17.600	4.39	0	0	0	1
Id2335	German, Mr. Aaron K	17.580	4.51	0	0	0	1

2325 rows × 21 columns

In [99]:

```
data.reset_index(inplace=True)
```

In [100]:

```
data.columns
```

Out[100]:

```
Index(['Customer ID', 'name', 'BMI', 'HBA1C', 'Heart Issues',
       'Any Transplants', 'Cancer history', 'NumberOfMajorSurgeries', 'smo
ker',
       'year', 'month', 'date', 'children', 'charges', 'Hospital tier',
       'City tier', 'State ID', 'R1011', 'R1012', 'R1013', 'Age', 'gende
r'],
      dtype='object')
```

In [101]:

```
data = data.drop(['name','year', 'month', 'date', 'State ID','Customer ID'],axis=1)
```

In [102]:

```
data
```

Out[102]:

	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	childre
0	47.410	7.47	0	0	0	0	1	
1	30.360	5.77	0	0	0	0	1	
2	38.095	6.05	0	0	0	0	1	
3	35.530	5.45	0	0	0	0	1	
4	32.800	6.59	0	0	0	0	1	
...	...	...	...	...	...	...	...	
2320	22.340	5.57	0	0	0	1	0	
2321	17.700	6.28	0	0	0	1	0	
2322	16.470	6.35	0	0	1	1	0	
2323	17.600	4.39	0	0	0	1	0	
2324	17.580	4.51	0	0	0	1	0	

2325 rows × 16 columns

In [145]:

```
from sklearn.preprocessing import LabelEncoder
```

In [104]:

```
le = LabelEncoder()
```

In [105]:

```
data['Hospital tier'] = le.fit_transform(data['Hospital tier'])
data['City tier'] = le.fit_transform(data['City tier'])
data['gender'] = le.fit_transform(data['gender'])
```

In [106]:

```
data
```

Out[106]:

	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	childre
0	47.410	7.47	0	0	0	0	1	
1	30.360	5.77	0	0	0	0	1	
2	38.095	6.05	0	0	0	0	1	
3	35.530	5.45	0	0	0	0	1	
4	32.800	6.59	0	0	0	0	1	
...	...	...	...	...	...	...	...	
2320	22.340	5.57	0	0	0	1	0	
2321	17.700	6.28	0	0	0	1	0	
2322	16.470	6.35	0	0	1	1	0	
2323	17.600	4.39	0	0	0	1	0	
2324	17.580	4.51	0	0	0	1	0	

2325 rows × 16 columns

In [107]:

```
data.dtypes
```

Out[107]:

```
BMI                       float64
HBA1C                     float64
Heart Issues                int32
Any Transplants             int32
Cancer history              int32
NumberOfMajorSurgeries      int32
smoker                      int32
children                    int64
charges                   float64
Hospital tier               int32
City tier                   int32
R1011                       uint8
R1012                       uint8
R1013                       uint8
Age                         int32
gender                      int32
dtype: object
```

In [108]:

```
from sklearn.model_selection import KFold
```

In [109]:

```
kf = KFold(n_splits=5,shuffle=True,random_state=42)
```

In [124]:

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
fold_num = 0
for train_index, test_index in kf.split(data):
    data.loc[test_index, 'fold'] = fold_num
    fold_num += 1
```

In [125]:

```
data
```

Out[125]:

	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	childre
0	47.410	7.47	0	0	0	0	1	
1	30.360	5.77	0	0	0	0	1	
2	38.095	6.05	0	0	0	0	1	
3	35.530	5.45	0	0	0	0	1	
4	32.800	6.59	0	0	0	0	1	
...	...	...	...	...	...	...	...	
2320	22.340	5.57	0	0	0	1	0	
2321	17.700	6.28	0	0	0	1	0	
2322	16.470	6.35	0	0	1	1	0	
2323	17.600	4.39	0	0	0	1	0	
2324	17.580	4.51	0	0	0	1	0	

2325 rows × 17 columns

In [126]:

```
data.fold.value_counts()
```

Out[126]:

```
3.0    465
4.0    465
1.0    465
2.0    465
0.0    465
Name: fold, dtype: int64
```

In [127]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [128]:

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
fold_var = np.zeros(len(data))


fold_num = 0
for train_index, test_index in kf.split(data):
    fold_var[test_index] = fold_num
    fold_num += 1


for fold in range(5):

    train_data = data[fold_var != fold]
    test_data = data[fold_var == fold]
    x_train, y_train = train_data.drop('charges', axis=1), train_data['charges']
    x_test, y_test = test_data.drop('charges', axis=1), test_data['charges']


    model = LinearRegression()
    model.fit(x_train, y_train)


    y_pred = model.predict(x_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print(f"Fold {fold+1} RMSE: {rmse}")
    print('-----------------------------------------------------------------------------


    coef = model.coef_
    var_imp = abs(coef) / np.sum(abs(coef))
    print(f"Fold {fold+1} Variable Importance Scores: {var_imp}")
    print('-----------------------------------------------------------------------------


    redundant_vars = x_train.columns[var_imp < 0.1]
    print(f"Fold {fold+1} Redundant Variables: {redundant_vars}")
    print('-----------------------------------------------------------')
```

```
Fold 1 RMSE: 4611.611687922367
--------------------------------------------------------------------------------------------------------------
Fold 1 Variable Importance Scores: [1.13012323e-02 2.03828066e-03 9.06695088e-03 9.92211763e-03
 8.18822190e-03 4.89258925e-04 7.81234629e-01 1.42483226e-02
 6.16519002e-02 8.37164858e-03 2.93791134e-02 1.28012866e-02
 3.67628691e-02 9.03450378e-03 4.48929525e-03 1.02036952e-03]
--------------------------------------------------------------------------------------------------------------
Fold 1 Redundant Variables: Index(['BMI', 'HBA1C', 'Heart Issues', 'Any Transplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'children', 'Hospital tier', 'City tier',
       'R1011', 'R1012', 'R1013', 'Age', 'gender', 'fold'],
      dtype='object')
------------------------------------------------------------
Fold 2 RMSE: 4754.444655312291
--------------------------------------------------------------------------------------------------------------
Fold 2 Variable Importance Scores: [1.05844486e-02 2.97740937e-03 3.26747808e-03 2.55349751e-02
 1.44554168e-02 3.66755452e-03 7.57719757e-01 1.24300647e-02
 6.36157300e-02 3.08359143e-03 3.24069637e-02 1.78814557e-02
 4.00015052e-02 8.99278998e-03 2.88565457e-03 4.95205476e-04]
--------------------------------------------------------------------------------------------------------------
Fold 2 Redundant Variables: Index(['BMI', 'HBA1C', 'Heart Issues', 'Any Transplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'children', 'Hospital tier', 'City tier',
       'R1011', 'R1012', 'R1013', 'Age', 'gender', 'fold'],
      dtype='object')
------------------------------------------------------------
Fold 3 RMSE: 4153.571434587202
--------------------------------------------------------------------------------------------------------------
Fold 3 Variable Importance Scores: [1.08562546e-02 2.04893904e-03 1.87372916e-03 2.31219477e-02
 7.15592909e-03 1.83739462e-03 7.64600877e-01 1.61701992e-02
 6.56549026e-02 1.70515171e-03 3.04186866e-02 2.02814218e-02
 4.30037081e-02 9.04224332e-03 1.46627715e-03 7.62338041e-04]
--------------------------------------------------------------------------------------------------------------
Fold 3 Redundant Variables: Index(['BMI', 'HBA1C', 'Heart Issues', 'Any Transplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'children', 'Hospital tier', 'City tier',
       'R1011', 'R1012', 'R1013', 'Age', 'gender', 'fold'],
      dtype='object')
------------------------------------------------------------
Fold 4 RMSE: 4199.501100854111
--------------------------------------------------------------------------------------------------------------
Fold 4 Variable Importance Scores: [1.10326113e-02 2.37630590e-03 3.18312541e-04 1.94139071e-02
 6.58118279e-03 1.58576685e-03 7.80743626e-01 1.39354683e-02
 6.41171292e-02 1.49449441e-03 2.79856165e-02 1.84571387e-02
 4.14094067e-02 9.18661104e-03 6.29017444e-04 7.33405126e-04]
--------------------------------------------------------------------------------------------------------------
Fold 4 Redundant Variables: Index(['BMI', 'HBA1C', 'Heart Issues', 'Any Transplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'children', 'Hospital tier', 'City tier',
```

```
       'R1011', 'R1012', 'R1013', 'Age', 'gender', 'fold'],
      dtype='object')
-----------------------------------------------------------
Fold 5 RMSE: 4578.029692769042
--------------------------------------------------------------------------
-----------------------------------
Fold 5 Variable Importance Scores: [0.01101121 0.00322359 0.00153848 0.025
97263 0.01557972 0.00433719
 0.75891022 0.01391795 0.0668392  0.00084366 0.03302267 0.01549911
 0.03562084 0.00899331 0.00197108 0.00271912]
--------------------------------------------------------------------------
-----------------------------------
Fold 5 Redundant Variables: Index(['BMI', 'HBA1C', 'Heart Issues', 'Any Tr
ansplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'children', 'Hospital tier', 'City tier',
       'R1011', 'R1012', 'R1013', 'Age', 'gender', 'fold'],
      dtype='object')
-----------------------------------------------------------
```

Random Forest

In [129]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [133]:

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
fold_var = np.zeros(len(data))


fold_num = 0
for train_index, test_index in kf.split(data):
    fold_var[test_index] = fold_num
    fold_num += 1


for fold in range(5):
    train_data = data[fold_var != fold]
    test_data = data[fold_var == fold]


    x_train, y_train = train_data.drop('charges', axis=1), train_data['charges']
    x_test, y_test = test_data.drop('charges', axis=1), test_data['charges']


    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(x_train, y_train)


    y_pred = rf.predict(x_test)
    rmse_rf = np.sqrt(np.mean((y_pred - y_test)**2))


    print(f"Fold {fold+1} Random Forest RMSE: {rmse_rf}")
    print('-----------------------------------------------------------------------------


    var_imp_rf = rf.feature_importances_
    print(f"Fold {fold+1} Random Forest Variable Importance Scores: {var_imp_rf}")
    print('-----------------------------------------------------------------------------
```

```
Fold 1 Random Forest RMSE: 3710.857599729278
--------------------------------------------------------------------------
------------------------------------
Fold 1 Random Forest Variable Importance Scores: [1.25596296e-01 1.2351231
5e-02 8.90328552e-04 3.57652034e-04
 1.05388846e-03 1.29083986e-03 7.07771559e-01 1.53797276e-02
 2.15140322e-02 2.47179244e-03 6.36914158e-03 1.64928533e-03
 6.37143846e-03 9.08151588e-02 2.28266460e-03 3.83496340e-03]
--------------------------------------------------------------------------
------------------------------------
Fold 2 Random Forest RMSE: 3546.9525694604
--------------------------------------------------------------------------
------------------------------------
Fold 2 Random Forest Variable Importance Scores: [1.27684291e-01 1.4280033
7e-02 1.09194079e-03 1.89599554e-04
 1.09339840e-03 1.21969960e-03 7.01990090e-01 1.21525069e-02
 1.92159459e-02 2.99043660e-03 7.54504366e-03 1.25850636e-03
 6.08112832e-03 9.76534621e-02 1.71421369e-03 3.83970341e-03]
--------------------------------------------------------------------------
------------------------------------
Fold 3 Random Forest RMSE: 3158.739895853653
--------------------------------------------------------------------------
------------------------------------
Fold 3 Random Forest Variable Importance Scores: [1.26074325e-01 1.4384344
3e-02 1.34523434e-03 2.37986000e-04
 1.40714745e-03 1.43199756e-03 6.94782504e-01 1.64352348e-02
 2.31247220e-02 2.57807180e-03 7.90878985e-03 1.56671928e-03
 6.76306617e-03 9.60720974e-02 2.22503856e-03 3.66272151e-03]
--------------------------------------------------------------------------
------------------------------------
Fold 4 Random Forest RMSE: 3265.7888606713213
--------------------------------------------------------------------------
------------------------------------
Fold 4 Random Forest Variable Importance Scores: [1.23954097e-01 1.4272802
3e-02 1.23049098e-03 2.51208532e-04
 1.08999338e-03 1.32723376e-03 6.90235210e-01 1.57895564e-02
 2.29530938e-02 3.32540133e-03 8.55141586e-03 1.96056327e-03
 6.23549421e-03 1.03116538e-01 1.90179297e-03 3.80510780e-03]
--------------------------------------------------------------------------
------------------------------------
Fold 5 Random Forest RMSE: 3651.092123409294
--------------------------------------------------------------------------
------------------------------------
Fold 5 Random Forest Variable Importance Scores: [1.22628690e-01 1.3126409
8e-02 9.30176142e-04 1.68523586e-04
 1.18637462e-03 1.14373143e-03 7.09884099e-01 1.34686040e-02
 2.31603418e-02 2.43870744e-03 7.79952556e-03 1.24009183e-03
 5.23292376e-03 9.16442325e-02 2.42197846e-03 3.52559050e-03]
--------------------------------------------------------------------------
------------------------------------
```

Extreme Gradient Boosting

In [131]:

```
import xgboost as xgb
```

In [134]:

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
fold_var = np.zeros(len(data))


fold_num = 0
for train_index, test_index in kf.split(data):
    fold_var[test_index] = fold_num
    fold_num += 1


for fold in range(5):
    train_data = data[fold_var != fold]
    test_data = data[fold_var == fold]

    x_train, y_train = train_data.drop('charges', axis=1), train_data['charges']
    x_test, y_test = test_data.drop('charges', axis=1), test_data['charges'] # XGBoost M


    xgb_model = xgb.XGBRegressor(objective='reg:squarederror', random_state=42)
    xgb_model.fit(x_train, y_train)


    y_pred = xgb_model.predict(x_test)
    rmse_xgb = np.sqrt(np.mean((y_pred - y_test)**2))


    print(f"Fold {fold+1} XGBoost RMSE: {rmse_xgb}")
    print('---------------------------------------------------------------------------


    var_imp_xgb = xgb_model.feature_importances_
    print(f"Fold {fold+1} XGBoost Variable Importance Scores: {var_imp_xgb}")
    print('---------------------------------------------------------------------------
```

```
Fold 1 XGBoost RMSE: 3936.1873713204104
------------------------------------------------------------------------
-----------------------------------
Fold 1 XGBoost Variable Importance Scores: [8.3757257e-03 1.6242140e-03 7.
0062949e-04 9.6031197e-04 2.3240210e-03
 9.7629137e-04 9.3367231e-01 5.1872721e-03 1.1289973e-02 1.1303592e-03
 7.0247352e-03 1.3614267e-03 6.0801036e-03 1.5035150e-02 2.6463401e-03
 1.6110581e-03]
------------------------------------------------------------------------
-----------------------------------
Fold 2 XGBoost RMSE: 3776.653937245996
------------------------------------------------------------------------
-----------------------------------
Fold 2 XGBoost Variable Importance Scores: [9.9673830e-03 2.0464694e-03 7.
4983941e-04 1.3561645e-03 1.5901846e-03
 1.9411473e-03 9.2143255e-01 4.6968861e-03 1.1310951e-02 1.3181042e-03
 7.9833400e-03 2.6200840e-03 7.9160547e-03 1.9858170e-02 3.1583058e-03
 2.0543735e-03]
------------------------------------------------------------------------
-----------------------------------
Fold 3 XGBoost RMSE: 3408.9339380971896
------------------------------------------------------------------------
-----------------------------------
Fold 3 XGBoost Variable Importance Scores: [9.17739514e-03 1.81060424e-03
2.71299831e-03 5.63423033e-04
 1.25360838e-03 1.47793046e-03 9.29428101e-01 5.72216697e-03
 1.10008540e-02 9.10245348e-04 6.78947195e-03 1.28322211e-03
 8.63673165e-03 1.54248485e-02 2.57212995e-03 1.23630441e-03]
------------------------------------------------------------------------
-----------------------------------
Fold 4 XGBoost RMSE: 3403.3873288354566
------------------------------------------------------------------------
-----------------------------------
Fold 4 XGBoost Variable Importance Scores: [0.00934123 0.00194182 0.001298
6  0.00172756 0.00227696 0.002191
 0.9197736  0.00630279 0.01303377 0.00179135 0.0066326  0.00306469
 0.0079066  0.01820624 0.0026424  0.00186881]
------------------------------------------------------------------------
-----------------------------------
Fold 5 XGBoost RMSE: 3855.122190627832
------------------------------------------------------------------------
-----------------------------------
Fold 5 XGBoost Variable Importance Scores: [9.03274398e-03 1.78122695e-03
1.61500345e-03 8.86582711e-04
 1.45067030e-03 1.25703460e-03 9.27026749e-01 5.05067268e-03
 1.29192285e-02 1.31072989e-03 6.07745675e-03 2.19040853e-03
 6.69756858e-03 1.78991910e-02 3.31570255e-03 1.48905499e-03]
------------------------------------------------------------------------
-----------------------------------
```

Predicted Hospitalization cost using all five models

In [135]:

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
fold_var = np.zeros(len(data))


fold_num = 0
for train_index, test_index in kf.split(data):
    fold_var[test_index] = fold_num
    fold_num += 1



all_preds = []
for fold in range(5):
    train_data = data[fold_var != fold]
    test_data = data[fold_var == fold]


    x_train, y_train = train_data.drop('charges', axis=1), train_data['charges']
    x_test, y_test = test_data.drop('charges', axis=1), test_data['charges']


    x_train = std.fit_transform(x_train)
    x_test = std.fit_transform(x_test)


    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(x_train, y_train)


    y_pred_rf = rf.predict(x_test)
    all_preds.append(y_pred_rf)
```

In [136]:

```
mean_preds = np.mean(all_preds, axis=0)
print(f"Mean Predicted Hospitalization Cost: {mean_preds}")
```

```
Mean Predicted Hospitalization Cost: [42279.0914  43072.07086 47312.59366
47029.29886 43961.17878 43825.87966
 42224.92238 43808.8899  42203.52442 44005.33734 43135.93402 42742.66768
 42660.81464 40582.4306  40582.82944 41391.94914 40336.35936 41692.45258
 42654.97768 39896.6113  38319.92132 39705.9306  36941.00736 37458.19606
 39803.13638 39930.57928 39369.3009  38229.98776 38628.37128 40169.23908
 36394.25294 33396.4205  37077.57214 34734.48024 38261.62568 38316.56046
 37795.67894 38274.23114 36440.54162 35720.95128 33142.66842 36324.64142
 32658.354   28800.76904 37297.40818 33413.05794 33657.78874 35818.17122
 28363.1465  31492.56742 34420.86946 32650.48864 36036.6024  31144.72286
 28784.08058 31739.13988 30840.98742 27270.87552 29572.2076  29893.28414
 28554.63606 24067.55134 25136.66286 30589.27458 25570.55632 27154.29584
 23640.05694 24463.49582 27532.74362 26068.02166 24145.94378 24961.8192
 28465.89062 24256.63234 20037.86586 26891.08388 29548.74314 21356.14948
 19891.15966 22054.0329  22522.86026 22801.84032 26532.81638 20369.30892
 20494.87898 21409.101   12123.40904 20009.72746 14333.99358 24556.4392
 18509.2408  22240.63814 20576.22876 21322.26836 22412.03824 19582.25686
 23057.66402 15934.47242 12622.37082 18200.69124 18198.5996  15854.53
 20804.50398 12535.49812 21321.0924  16484.04654 16928.95646 16530.12978
 18362.1633  13981.1636  17852.02224 19167.63324 15785.87524 20272.84998
 17156.12886 18452.1032  15520.57312 16941.87224 16309.73574 14128.8721
 16472.67832 14328.1345  17286.83844 17311.68762 13955.89054 17650.12088
 17438.08044 15786.36146 13082.9212  14444.87692 15295.16716 15916.41432
 11092.63172 14934.17096 16097.52146 15375.6704  14631.61446 11521.89876
 17895.14934 15433.6765  15715.07024 16535.5574  15062.87614 16032.56636
 13175.24116 13794.8672  12884.85702 13714.03714 13226.5045  14962.82028
 13717.05796 11746.76704 13305.77872 13365.6543  13502.2162  12992.706
 13154.44862 12009.34814 12440.33438 11352.64766 12631.5336   9978.0876
 12533.87418 12472.5368  10317.63782 12229.78764  9580.0344  12795.30206
 12346.70916 12191.46538 11703.7814  11661.66248 12204.1261  10474.91236
 12096.0875  12690.16078 12289.69218 11862.57882 12105.54466 11660.9765
 11655.62496 11882.99028 11475.77032 11214.48702 11731.43802 10652.10836
 10878.65424 11344.37462 12069.45408 11358.43048 12013.57828  9384.35742
 12203.61244 10921.507   11486.1652  11944.62636 10308.69554 11094.76316
 10851.3653  11267.67162  9064.60408 10925.87386 11531.14268 10876.05716
 11217.49634 11777.0814  11332.55218 11271.76304 11107.9611  10926.40912
  9250.55176 10467.72068 10863.87888 10743.09314 10974.08272 10831.4899
 10833.83306 11121.28142 10546.46984 12179.89958 10625.93366 10699.5269
  9595.71044 10738.1901  10016.40798 12792.64556 10597.3797  10888.36894
 10024.23442 11573.76114 10123.24004  9924.57186 10293.1769  10435.56842
 11186.04634 10603.29246 10213.5639  10280.4695   9668.79704 11839.84502
 11359.44178 10741.74964 12002.7874   8641.8351  11035.89848 10332.83054
  9437.07846 10535.01272  8595.92236 10286.85698 10128.4141   9742.77356
  9010.2679   9005.06758  8632.90942  9695.64138  8136.04818 10243.80232
  9025.0124   9244.84942  9445.64032  9956.38138  9356.87452  8910.12206
  8806.31186  9565.40112  8573.82464 11412.77386  8696.588    7714.19762
  8783.33234 11926.62898 11097.258    9214.8918   8947.18454  7607.79292
  8810.34214  9210.45196  8599.70776  8248.7354   7805.83384  8179.11072
  7658.9713   6573.56674  8494.33612  7630.7484   7730.25278  7691.84762
  7355.74924  7887.18666  7369.92672  7657.67374  6948.77676  7907.79338
  7746.46296  6955.61246  7290.82896  9542.19222  7664.69304  6268.8909
  7426.42622  7858.22648  7657.58478  7245.14012  6833.29656  7497.23378
  6530.30518  6174.87078  7947.51094  7797.11464  6934.10888  6412.88816
  6148.97776  7244.4186   7989.71624  7259.40116  6844.61936  7522.28636
  7746.57616  6281.0726   7085.32762  7515.43392  8155.20094  6880.52586
  6941.74452  6505.45104  5917.40998  7051.09274  6181.35404  6243.53812
  7502.25696  6313.20918  5673.444    7249.14226  5285.95862  5879.30752
  5792.171    6571.8693   5442.83114  7479.44786  6209.38642  5902.0303
  6296.51036  5580.33912  5429.0115   5875.40046  5354.00374  5690.72412
  6372.17832  5296.27396  5603.746    5170.00814  5657.94858  5974.6816
  4754.09266  7131.20002  6052.91704  5241.09076  5348.96746  6207.72958
```

```
 5948.9436   5388.78594  6651.03156  5208.95932  5969.18326  6876.30922
 7186.63314  6482.25854  6496.64542  6025.97366  6038.63776  6382.52228
 5426.87818  4323.36558  5721.79056  5709.72806  5003.6778   5107.79808
 3981.83356  4394.8734   5332.482    7373.3238   5009.48342  4929.54314
 4673.44206  5442.89376  4395.07194  5465.73506  5116.42704  5776.16944
 5073.1058   7803.14762  4194.39858  5563.5206   4580.31376  3809.50982
 4333.47614  4266.62634  5411.7071   4717.49944  6619.8534   4050.08132
 3899.25744  3646.83514  3757.50144  3888.16458  4514.02928  3813.66534
 3580.22294  3694.3387   3494.41478  5143.40804  4550.70052  3011.9825
 3927.60534  3842.47728  3862.34244  3822.3858   3457.93488  4501.78628
 3004.16104  3183.2468   2571.32872  3030.96058  3295.57728  3369.34178
 3134.3495   3464.5363   2753.25754  3505.04598  2791.04406  3601.46054
 2628.81174  3208.30822  3277.00338  2319.23734  2902.72922  3361.25192
 2626.79214  2255.62124  1879.34514  2248.98964  2675.88506  2030.56118
 1848.43856  1911.91362  1518.5686   2348.16758  1793.16328  1996.47564
 1381.2386   3837.05074  1856.79702  2026.32814  2193.26524  1279.09968
 2701.8444   2053.68216  1192.72046  1200.91484  1574.18302  1582.86336
 1093.01644  1131.55714  1953.51784]
```

Case scenario:

In [137]:

```
import statsmodels.api as sm
```

In [139]:

```
data
```

Out[139]:

	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	childre
0	47.410	7.47	0	0	0	0	1	
1	30.360	5.77	0	0	0	0	1	
2	38.095	6.05	0	0	0	0	1	
3	35.530	5.45	0	0	0	0	1	
4	32.800	6.59	0	0	0	0	1	
...	...	...	...	...	...	...	...	
2320	22.340	5.57	0	0	0	1	0	
2321	17.700	6.28	0	0	0	1	0	
2322	16.470	6.35	0	0	1	1	0	
2323	17.600	4.39	0	0	0	1	0	
2324	17.580	4.51	0	0	0	1	0	

2325 rows × 17 columns

In [141]:

```
X = data.drop(['charges','fold'],axis=1)
```

In [142]:

```
X
```

Out[142]:

	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	childre
0	47.410	7.47	0	0	0	0	1	
1	30.360	5.77	0	0	0	0	1	
2	38.095	6.05	0	0	0	0	1	
3	35.530	5.45	0	0	0	0	1	
4	32.800	6.59	0	0	0	0	1	
...	...	...	...	...	...	...	...	
2320	22.340	5.57	0	0	0	1	0	
2321	17.700	6.28	0	0	0	1	0	
2322	16.470	6.35	0	0	1	1	0	
2323	17.600	4.39	0	0	0	1	0	
2324	17.580	4.51	0	0	0	1	0	

2325 rows × 15 columns

In [143]:

```
y = data['charges']
```

In [144]:

```
y
```

Out[144]:

```
0       63770.43
1       62592.87
2       58571.07
3       55135.40
4       52590.83
          ...   
2320      637.26
2321      604.54
2322      600.00
2323      570.62
2324      563.84
Name: charges, Length: 2325, dtype: float64
```

In [147]:

```
X = sm.add_constant(X)
```

In [148]:

```
X
```

Out[148]:

	const	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker
0	1.0	47.410	7.47	0	0	0	0	1
1	1.0	30.360	5.77	0	0	0	0	1
2	1.0	38.095	6.05	0	0	0	0	1
3	1.0	35.530	5.45	0	0	0	0	1
4	1.0	32.800	6.59	0	0	0	0	1
...	...	...	...	...	...	...	...	...
2320	1.0	22.340	5.57	0	0	0	1	0
2321	1.0	17.700	6.28	0	0	0	1	0
2322	1.0	16.470	6.35	0	0	1	1	0
2323	1.0	17.600	4.39	0	0	0	1	0
2324	1.0	17.580	4.51	0	0	0	1	0

2325 rows × 16 columns

In [149]:

```
model = sm.OLS(y,X)
```

In [150]:

```
results = model.fit()
```

In [151]:

```
print(results.summary())
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                charges   R-squared:
0.860
Model:                            OLS   Adj. R-squared:
0.859
Method:                 Least Squares   F-statistic:                     9
47.1
Date:                Sun, 02 Apr 2023   Prob (F-statistic):
0.00
Time:                        20:57:58   Log-Likelihood:                -22
823.
No. Observations:                2325   AIC:                         4.568
e+04
Df Residuals:                    2309   BIC:                         4.577
e+04
Df Model:                          15
Covariance Type:            nonrobust
==========================================================================
================
                            coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------
----------------
const                 -9770.9356    605.978    -16.124      0.000    -1.1
e+04   -8582.618
BMI                     318.9033     10.934     29.167      0.000     29
7.462     340.344
HBA1C                    74.4277     48.375      1.539      0.124     -2
0.434     169.290
Heart Issues            -34.3040    198.121     -0.173      0.863    -42
2.818     354.210
Any Transplants         608.8476    451.544      1.348      0.178    -27
6.627    1494.322
Cancer history          307.8973    264.267      1.165      0.244    -21
0.327     826.122
NumberOfMajorSurgeries   -60.5132   144.749     -0.418      0.676    -34
4.364     223.338
smoker                 2.237e+04    274.925     81.377      0.000    2.18
e+04    2.29e+04
children                412.9601     76.802      5.377      0.000     26
2.352     563.568
Hospital tier         -1873.2337    170.292    -11.000      0.000   -220
7.175   -1539.292
City tier                33.9287    114.849      0.295      0.768    -19
1.289     259.146
R1011                  -889.7118    276.423     -3.219      0.001   -143
1.776    -347.648
R1012                  -492.4038    264.940     -1.859      0.063   -101
1.949      27.141
R1013                 -1144.0982    264.817     -4.320      0.000   -166
3.403    -624.794
Age                     263.4259      8.995     29.286      0.000     24
5.787     281.065
gender                  -46.4049    186.038     -0.249      0.803    -41
1.223     318.413
==========================================================================
====
Omnibus:                  786.998   Durbin-Watson:
1.589
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               478
3.358
Skew:                           1.463   Prob(JB):
0.00
Kurtosis:                       9.389   Cond. No.
370.
==========================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

From the above stats results the HBA1C, Heart Issues, Any Transplants, Cancer History, Number of Major Surgeries, City tier, R1012, gender, the p-value is greater than 0.05 (p-value > 0.05). So the columns are been dropped and the results are taken again

In [152]:

```
X = X.drop(['HBA1C','Heart Issues','Any Transplants','Cancer history','NumberOfMajorSurg
```

In [153]:

```
X
```

Out[153]:

	const	BMI	smoker	children	Hospital tier	R1011	R1013	Age
0	1.0	47.410	1	0	0	0	1	55
1	1.0	30.360	1	0	1	0	1	46
2	1.0	38.095	1	1	0	0	0	32
3	1.0	35.530	1	0	0	0	0	34
4	1.0	32.800	1	0	0	1	0	61
...	...	...	...	...	...	...	...	...
2320	1.0	22.340	0	0	2	0	1	25
2321	1.0	17.700	0	0	2	0	1	31
2322	1.0	16.470	0	0	1	0	1	30
2323	1.0	17.600	0	0	1	0	1	31
2324	1.0	17.580	0	0	1	0	1	31

2325 rows × 8 columns

In [154]:

```
model = sm.OLS(y,X)
```

In [155]:

```
results = model.fit()
```

In [156]:

```
print(results.summary())
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                charges   R-squared:
0.860
Model:                            OLS   Adj. R-squared:
0.859
Method:                 Least Squares   F-statistic:                     2
027.
Date:                Sun, 02 Apr 2023   Prob (F-statistic):
0.00
Time:                        21:00:26   Log-Likelihood:                -22
827.
No. Observations:                2325   AIC:                         4.567
e+04
Df Residuals:                    2317   BIC:                         4.572
e+04
Df Model:                           7
Covariance Type:            nonrobust
==========================================================================
=======
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
-------
const         -9504.1311    504.941    -18.822      0.000   -1.05e+04   -8
513.947
BMI             319.8013     10.892     29.360      0.000     298.442
341.161
smoker         2.237e+04    273.448     81.818      0.000    2.18e+04
2.29e+04
children        382.9369     74.917      5.111      0.000     236.025
529.849
Hospital tier -1853.7165    169.448    -10.940      0.000   -2186.001   -1
521.432
R1011          -642.5927    241.057     -2.666      0.008   -1115.302    -
169.883
R1013          -877.8975    227.525     -3.858      0.000   -1324.070    -
431.725
Age             263.3516      7.006     37.587      0.000     249.612
277.091
==========================================================================
====
Omnibus:                      787.600   Durbin-Watson:
1.578
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             474
9.954
Skew:                           1.467   Prob(JB):
0.00
Kurtosis:                       9.358   Cond. No.
295.
==========================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

From the above stats model no variable p-value has greater than 0.05. so we can say that our model

is significant

In [157]:

```
case = pd.DataFrame({
    'BMI':[29.41],
    'HBA1C':[5.8],
    'Heart Issues':['no'],
    'Any Transplants':[0],
    'Cancer history':['yes'],
    'NumberOfMajorSurgeries':['No major surgery'],
    'smoker':['yes'],
    'children':[2],
    'Hospital tier':['tier - 1'],
    'City tier':['tier - 1'],
    'State ID':['R1011'],
    'Age':[35],
    'gender':['Female']
})
```

This is data given from the given case scenario

In [158]:

```
case
```

Out[158]:

	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	children	H
0	29.41	5.8	no	0	yes	No major surgery	yes	2	

from the given details from the questions we are taken the following required informations as follows

BMI - 29.41

Smoker - 1 (yes)

Children - 2

Hospital tier - 1 (tier 1)

R1011 - 1 (yes)

R1013 - 0 (no)

Age - 35

By using the formula we are predicting the charges for the given condions:

Formula = (-9504.1311)+(319.80*BMI)+((2.237e+04)*Smoker)+(382.9369*Children)+(-1853.72*Hospital tier)+(-642.5927*R1011)+(-877.8975*R1013)+(263.3516*Age)

Formula = (-9504.1311)+(319.80_29.41)+((2.237e4)_1)+(382.9369_2)+(-1853.72_1)+(-642.5927_1)+ (-877.8975_0)+(263.3516*35)

The 14232.1001 is the predicted charges for the case scenario. The Predicted charges is not accurate it may vary by giving some more relavent or dependent features.

In [ ]:

```

```