```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')

Hosp=pd.read_csv('Hospitalisation details.csv')
Medic=pd.read_csv('Medical Examinations.csv')
Names=pd.read_excel('Names.xlsx')

Hosp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2343 entries, 0 to 2342
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Customer ID    2343 non-null   object
 1   year           2343 non-null   object
 2   month          2343 non-null   object
 3   date           2343 non-null   int64
 4   children       2343 non-null   int64
 5   charges        2343 non-null   float64
 6   Hospital tier  2343 non-null   object
 7   City tier      2343 non-null   object
 8   State ID       2343 non-null   object
dtypes: float64(1), int64(2), object(6)
memory usage: 164.9+ KB
```

```python
Medic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2335 entries, 0 to 2334
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Customer ID             2335 non-null   object
 1   BMI                     2335 non-null   float64
 2   HBA1C                   2335 non-null   float64
 3   Heart Issues            2335 non-null   object
 4   Any Transplants         2335 non-null   object
 5   Cancer history          2335 non-null   object
 6   NumberOfMajorSurgeries  2335 non-null   object
 7   smoker                  2335 non-null   object
dtypes: float64(2), object(6)
memory usage: 146.1+ KB
```

```
Names.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2335 entries, 0 to 2334
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Customer ID  2335 non-null   object
 1   name         2335 non-null   object
dtypes: object(2)
memory usage: 36.6+ KB
```

Collate the files so that all the information is in one place

```
Customer_details=pd.merge(Hosp,Medic,how='inner',on='Customer ID')
Customer_details
```

```
     Customer ID  year month  date  children   charges Hospital
tier  \
0         Id2335  1992   Jul     9         0    563.84      tier - 2

1         Id2334  1992   Nov    30         0    570.62      tier - 2

2         Id2333  1993   Jun    30         0    600.00      tier - 2

3         Id2332  1992   Sep    13         0    604.54      tier - 3

4         Id2331  1998   Jul    27         0    637.26      tier - 3

...          ...   ...   ...   ...       ...       ...           ...

2330         Id5  1989   Jun    19         0  55135.40      tier - 1

2331         Id4  1991   Jun     6         1  58571.07      tier - 1

2332         Id3  1970     ?    11         3  60021.40      tier - 1

2333         Id2  1977   Jun     8         0  62592.87      tier - 2

2334         Id1  1968   Oct    12         0  63770.43      tier - 1


     City tier State ID     BMI  HBA1C Heart Issues Any Transplants  \
0     tier - 3    R1013  17.580   4.51           No              No
1     tier - 1    R1013  17.600   4.39           No              No
2     tier - 1    R1013  16.470   6.35           No              No
3     tier - 3    R1013  17.700   6.28           No              No
```

```
4     tier - 3    R1013  22.340   5.57           No              No
...        ...      ...     ...    ...          ...             ...
2330  tier - 2    R1012  35.530   5.45           No              No
2331  tier - 3    R1024  38.095   6.05           No              No
2332  tier - 1    R1012  34.485  11.87          yes              No
2333  tier - 3    R1013  30.360   5.77           No              No
2334  tier - 3    R1013  47.410   7.47           No              No

     Cancer history NumberOfMajorSurgeries smoker
0                No                      1     No
1                No                      1     No
2               Yes                      1     No
3                No                      1     No
4                No                      1     No
...             ...                    ...    ...
2330             No      No major surgery    yes
2331             No      No major surgery    yes
2332             No                      2    yes
2333             No      No major surgery    yes
2334             No      No major surgery    yes

[2335 rows x 16 columns]
```

```
Customer_details.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2335 entries, 0 to 2334
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Customer ID             2335 non-null   object
 1   year                    2335 non-null   object
 2   month                   2335 non-null   object
 3   date                    2335 non-null   int64
 4   children                2335 non-null   int64
 5   charges                 2335 non-null   float64
 6   Hospital tier           2335 non-null   object
 7   City tier               2335 non-null   object
 8   State ID                2335 non-null   object
 9   BMI                     2335 non-null   float64
 10  HBA1C                   2335 non-null   float64
 11  Heart Issues            2335 non-null   object
 12  Any Transplants         2335 non-null   object
 13  Cancer history          2335 non-null   object
 14  NumberOfMajorSurgeries  2335 non-null   object
 15  smoker                  2335 non-null   object
dtypes: float64(3), int64(2), object(11)
memory usage: 310.1+ KB
```

```
Customer_details=Customer_details.merge(Names,on='Customer ID')
Customer_details
```

	Customer ID	year	month	date	children	charges	Hospital tier
0	Id2335	1992	Jul	9	0	563.84	tier - 2
1	Id2334	1992	Nov	30	0	570.62	tier - 2
2	Id2333	1993	Jun	30	0	600.00	tier - 2
3	Id2332	1992	Sep	13	0	604.54	tier - 3
4	Id2331	1998	Jul	27	0	637.26	tier - 3
...	...	...	...	...	...	...	...
2330	Id5	1989	Jun	19	0	55135.40	tier - 1
2331	Id4	1991	Jun	6	1	58571.07	tier - 1
2332	Id3	1970	?	11	3	60021.40	tier - 1
2333	Id2	1977	Jun	8	0	62592.87	tier - 2
2334	Id1	1968	Oct	12	0	63770.43	tier - 1

	City tier	State ID	BMI	HBA1C	Heart Issues	Any Transplants
0	tier - 3	R1013	17.580	4.51	No	No
1	tier - 1	R1013	17.600	4.39	No	No
2	tier - 1	R1013	16.470	6.35	No	No
3	tier - 3	R1013	17.700	6.28	No	No
4	tier - 3	R1013	22.340	5.57	No	No
...	...	...	...	...	...	...
2330	tier - 2	R1012	35.530	5.45	No	No
2331	tier - 3	R1024	38.095	6.05	No	No
2332	tier - 1	R1012	34.485	11.87	yes	No
2333	tier - 3	R1013	30.360	5.77	No	No
2334	tier - 3	R1013	47.410	7.47	No	No

	Cancer history	NumberOfMajorSurgeries	smoker
0	No	1	No
1	No	1	No
2	Yes	1	No
3	No	1	No
4	No	1	No
...	...	...	...
2330	No	No major surgery	yes
2331	No	No major surgery	yes
2332	No	2	yes

```
2333             No       No major surgery    yes
2334             No       No major surgery    yes

                                    name
0                    German, Mr.  Aaron K
1                  Rosendahl, Mr.  Evan P
2                      Albano, Ms.  Julie
3     Riveros Gonzalez, Mr.  Juan D. Sr.
4                   Brietzke, Mr.  Jordan
...                                   ...
2330                 Kadala, Ms.  Kristyn
2331                 Osborne, Ms.  Kelsey
2332                        Lu, Mr.  Phil
2333               Lehner, Mr.  Matthew D
2334                    Hawks, Ms.  Kelly

[2335 rows x 17 columns]
```

```
Customer_details.columns=Customer_details.columns.str.lower()
Customer_details.columns=Customer_details.columns.str.replace(' ','_')
Customer_details.columns
```

```
Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name'],
      dtype='object')
```

```
Customer_details.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   year                    2335 non-null   object
 2   month                   2335 non-null   object
 3   date                    2335 non-null   int64
 4   children                2335 non-null   int64
 5   charges                 2335 non-null   float64
 6   hospital_tier           2335 non-null   object
 7   city_tier               2335 non-null   object
 8   state_id                2335 non-null   object
 9   bmi                     2335 non-null   float64
 10  hba1c                   2335 non-null   float64
 11  heart_issues            2335 non-null   object
 12  any_transplants         2335 non-null   object
 13  cancer_history          2335 non-null   object
 14  numberofmajorsurgeries  2335 non-null   object
```

```
 15   smoker                     2335 non-null   object
 16   name                       2335 non-null   object
dtypes: float64(3), int64(2), object(12)
memory usage: 328.4+ KB
```

2. Check for missing values in the dataset

```
Customer_details.isnull().sum()

customer_id                 0
year                        0
month                       0
date                        0
children                    0
charges                     0
hospital_tier               0
city_tier                   0
state_id                    0
bmi                         0
hba1c                       0
heart_issues                0
any_transplants             0
cancer_history              0
numberofmajorsurgeries      0
smoker                      0
name                        0
dtype: int64

# The data seems to have trivial values in a few variables. These are
“?” in all coulmns
(Customer_details== '?' ).sum()

customer_id                 0
year                        2
month                       3
date                        0
children                    0
charges                     0
hospital_tier               1
city_tier                   1
state_id                    2
bmi                         0
hba1c                       0
heart_issues                0
any_transplants             0
cancer_history              0
numberofmajorsurgeries      0
smoker                      2
```

```
name                                0
dtype: int64
```

```
(Customer_details== '?' ).sum(axis=1).head(20)
```

```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     1
12     0
13     1
14     0
15     0
16     0
17     2
18     0
19     0
dtype: int64
```

```
Customer_details.shape
```

```
(2335, 17)
```

```
Customer_details.shape[1]
```

```
17
```

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

percentage of trivial value in all columns

```
Miss_perc= (Customer_details==
'?' ).sum(axis=1)/Customer_details.shape[1] * 100
Miss_perc
Miss_perc[Miss_perc > 0]
```

```
11       5.882353
13       5.882353
17      11.764706
542      5.882353
1046     5.882353
1049     5.882353
1700     5.882353
1775     5.882353
2165     5.882353
2332     5.882353
dtype: float64
```

```
Miss_perc[Miss_perc > 0].index
```

```
Int64Index([11, 13, 17, 542, 1046, 1049, 1700, 1775, 2165, 2332],
dtype='int64')
```

percentage of trivial value in all rows

```
Miss_perc_rows= (Customer_details==
'?' ).sum(axis=0)/Customer_details.shape[0] * 100
Miss_perc_rows.sort_values(ascending=False)
```

```
month                        0.128480
state_id                     0.085653
year                         0.085653
smoker                       0.085653
city_tier                    0.042827
hospital_tier                0.042827
date                         0.000000
children                     0.000000
charges                      0.000000
name                         0.000000
bmi                          0.000000
hba1c                        0.000000
heart_issues                 0.000000
any_transplants              0.000000
cancer_history               0.000000
numberofmajorsurgeries       0.000000
customer_id                  0.000000
dtype: float64
```

```
Customer_details_noQ=Customer_details.drop(Miss_perc[Miss_perc >
0].index)
Customer_details_noQ
```

```
     customer_id  year month  date  children   charges
hospital_tier  \
0         Id2335  1992   Jul     9         0    563.84      tier - 2
```

```
1          Id2334  1992   Nov    30         0    570.62      tier - 2

2          Id2333  1993   Jun    30         0    600.00      tier - 2

3          Id2332  1992   Sep    13         0    604.54      tier - 3

4          Id2331  1998   Jul    27         0    637.26      tier - 3

...           ...   ...   ...   ...       ...       ...           ...

2329          Id6  1962   Aug     4         0  52590.83      tier - 1

2330          Id5  1989   Jun    19         0  55135.40      tier - 1

2331          Id4  1991   Jun     6         1  58571.07      tier - 1

2333          Id2  1977   Jun     8         0  62592.87      tier - 2

2334          Id1  1968   Oct    12         0  63770.43      tier - 1


     city_tier state_id     bmi  hba1c heart_issues any_transplants  \
0     tier - 3    R1013  17.580   4.51           No              No
1     tier - 1    R1013  17.600   4.39           No              No
2     tier - 1    R1013  16.470   6.35           No              No
3     tier - 3    R1013  17.700   6.28           No              No
4     tier - 3    R1013  22.340   5.57           No              No
...        ...      ...     ...    ...          ...             ...
2329  tier - 3    R1011  32.800   6.59           No              No
2330  tier - 2    R1012  35.530   5.45           No              No
2331  tier - 3    R1024  38.095   6.05           No              No
2333  tier - 3    R1013  30.360   5.77           No              No
2334  tier - 3    R1013  47.410   7.47           No              No

     cancer_history numberofmajorsurgeries smoker  \
0                No                      1     No
1                No                      1     No
2               Yes                      1     No
3                No                      1     No
4                No                      1     No
...             ...                    ...    ...
2329             No       No major surgery    yes
2330             No       No major surgery    yes
2331             No       No major surgery    yes
2333             No       No major surgery    yes
2334             No       No major surgery    yes

                                     name
0                    German, Mr.  Aaron K
1                  Rosendahl, Mr.  Evan P
```

```
2                        Albano, Ms.  Julie
3     Riveros Gonzalez, Mr.  Juan D. Sr.
4                     Brietzke, Mr.  Jordan
...                                      ...
2329                 Baker, Mr.  Russell B.
2330                   Kadala, Ms.  Kristyn
2331                   Osborne, Ms.  Kelsey
2333                 Lehner, Mr.  Matthew D
2334                        Hawks, Ms.  Kelly

[2325 rows x 17 columns]
```

```
Customer_details_noQ.columns
```

```
Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name'],
      dtype='object')
```

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

Label Encoding

Nominal Variable -State ID

Ordinal Variable - Ranking vaiables-numbers are assigned to categories based on rank - 'Hospital tier', 'City tier'

```
Customer_details_noQ['hospital_tier'].unique()
```

```
array(['tier - 2', 'tier - 3', 'tier - 1'], dtype=object)
```

```
Customer_details_noQ.groupby('hospital_tier').count()['customer_id']
```

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

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

```
label_encoder=LabelEncoder()
Customer_details_noQ['hospital_tier_ord']=label_encoder.fit_transform(
Customer_details_noQ['hospital_tier'])

Customer_details_noQ['hospital_tier_ord'].unique()

array([1, 2, 0])

Customer_details_noQ['hospital_tier_ord']

0       1
1       1
2       1
3       2
4       2
       ..
2329    0
2330    0
2331    0
2333    1
2334    0
Name: hospital_tier_ord, Length: 2325, dtype: int32

pd.crosstab(Customer_details_noQ['hospital_tier_ord'],Customer_details
_noQ['hospital_tier'])

hospital_tier      tier - 1  tier - 2  tier - 3
hospital_tier_ord
0                       300         0         0
1                         0      1334         0
2                         0         0       691

Customer_details_noQ['city_tier'].unique()

array(['tier - 3', 'tier - 1', 'tier - 2'], dtype=object)

Customer_details_noQ.groupby('city_tier').count()['customer_id']

city_tier
tier - 1    729
tier - 2    807
tier - 3    789
Name: customer_id, dtype: int64

Customer_details_noQ['city_tier_ord']=
label_encoder.fit_transform(Customer_details_noQ['city_tier'])
Customer_details_noQ['city_tier_ord']

0       2
1       0
2       0
3       2
```

```
4       2
       ..
2329    2
2330    1
2331    2
2333    2
2334    2
Name: city_tier_ord, Length: 2325, dtype: int32
```

```
Customer_details_noQ['city_tier_ord'].unique()
```

```
array([2, 0, 1])
```

```
pd.crosstab(Customer_details_noQ['city_tier_ord'],Customer_details_noQ
['city_tier'])
```

```
city_tier      tier - 1  tier - 2  tier - 3
city_tier_ord
0                   729         0         0
1                     0       807         0
2                     0         0       789
```

```
Customer_details_noQ.head(5)
```

```
  customer_id  year month  date  children  charges hospital_tier
city_tier  \
0      Id2335  1992   Jul     9         0   563.84      tier - 2  tier
- 3
1      Id2334  1992   Nov    30         0   570.62      tier - 2  tier
- 1
2      Id2333  1993   Jun    30         0   600.00      tier - 2  tier
- 1
3      Id2332  1992   Sep    13         0   604.54      tier - 3  tier
- 3
4      Id2331  1998   Jul    27         0   637.26      tier - 3  tier
- 3

  state_id    bmi  hba1c heart_issues any_transplants
cancer_history  \
0    R1013  17.58   4.51           No              No             No

1    R1013  17.60   4.39           No              No             No

2    R1013  16.47   6.35           No              No            Yes

3    R1013  17.70   6.28           No              No             No

4    R1013  22.34   5.57           No              No             No


  numberofmajorsurgeries smoker                                name  \
```

```
0                      1     No                German, Mr.  Aaron K
1                      1     No              Rosendahl, Mr.  Evan P
2                      1     No                  Albano, Ms.  Julie
3                      1     No  Riveros Gonzalez, Mr.  Juan D. Sr.
4                      1     No                Brietzke, Mr.  Jordan

   hospital_tier_ord  city_tier_ord
0                  1              2
1                  1              0
2                  1              0
3                  2              2
4                  2              2
```

5. The dataset has State ID, which has around 16 states. All states are not represented in equal proportions in the data. Creating dummy variables for all regions may also result in too many insignificant predictors. Nevertheless, only R1011, R1012, and R1013 are worth investigating further. Design a suitable strategy to create dummy variables with these restraints.

Creating dummy variable of all 16 states will may lead to insignificant predictors.

####Choosing the most frequent of each category

```
Customer_details_noQ['state_id'].unique()

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

SC=Customer_details_noQ.groupby('state_id').count()
['customer_id'].sort_values(ascending=False)
SC
```

```
state_id
R1013    609
R1011    574
R1012    572
R1024    159
R1026     84
R1021     70
R1016     64
R1025     40
R1023     38
R1017     36
R1019     26
R1022     14
R1014     13
R1015     11
R1018      9
R1020      6
Name: customer_id, dtype: int64
```

```
SC[:3].index   # first 3 state are most frequent
```

```
Index(['R1013', 'R1011', 'R1012'], dtype='object', name='state_id')
```

```
for i in SC [:3].index:
    var_name= 'State_ID_'+ i
    print(var_name)
    Customer_details_noQ[var_name]=0
    Customer_details_noQ.loc[Customer_details_noQ['state_id'] ==
i,var_name]=1
```

```
State_ID_R1013
State_ID_R1011
State_ID_R1012
```

```
#Customer_details_noQ['State ID']
Customer_details_noQ['State_ID_R1013'].value_counts()
```

```
0    1716
1     609
Name: State_ID_R1013, dtype: int64
```

```
Customer_details_noQ['State_ID_R1012'].value_counts()
```

```
0    1753
1     572
Name: State_ID_R1012, dtype: int64
```

```
Customer_details_noQ['State_ID_R1011'].value_counts()
```

```
0    1751
1     574
Name: State_ID_R1011, dtype: int64
```

6. The variable NumberOfMajorSurgeries also appears to have string values. Apply a suitable method to clean up this variable.

```
Customer_details_noQ['numberofmajorsurgeries'].unique()

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

Customer_details_noQ['numberofmajorsurgeries'].value_counts()

No major surgery    1070
1                    961
2                    272
3                     22
Name: numberofmajorsurgeries, dtype: int64

Customer_details_noQ['numberofmajorsurgeries']=pd.to_numeric(Customer_
details_noQ['numberofmajorsurgeries'],errors='coerce')
Customer_details_noQ['numberofmajorsurgeries']

0       1.0
1       1.0
2       1.0
3       1.0
4       1.0
       ...
2329    NaN
2330    NaN
2331    NaN
2333    NaN
2334    NaN
Name: numberofmajorsurgeries, Length: 2325, dtype: float64

Customer_details_noQ['numberofmajorsurgeries'].fillna(0,inplace=True)

Customer_details_noQ['numberofmajorsurgeries'].unique()

array([1., 0., 2., 3.])
```

7. Age appears to be a significant factor in this analysis. Calculate the patients' ages based on their

dates of birth.

```
Customer_details_noQ.year=Customer_details_noQ.year.astype(int)
Customer_details_noQ['age']=2024-Customer_details_noQ.year
Customer_details_noQ['age']
```

```
0       32
1       32
2       31
3       32
4       26
        ..
2329    62
2330    35
2331    33
2333    47
2334    56
Name: age, Length: 2325, dtype: int32
```

8. The gender of the patient may be an important factor in determining the cost of hospitalization. The salutations in a beneficiary's name can be used to determine their gender. Make a new field for the beneficiary's gender.

```
Customer_details_noQ['title']=Customer_details_noQ['name'].str.split('
[,.]').str[1].str.strip()
Customer_details_noQ['title'].value_counts()
```

```
Mr     1160
Ms     1023
Mrs     142
Name: title, dtype: int64
```

```
Customer_details_noQ.shape
```

```
(2325, 24)
```

```
Customer_details_noQ [ 'gender'] = 'Female'
Customer_details_noQ.loc[Customer_details_noQ.title == 'Mr' ,
'gender'] = 'Male'


Customer_details_noQ['gender']
```

```
0         Male
1         Male
2       Female
3         Male
4         Male
         ...
2329      Male
2330    Female
2331    Female
2333      Male
2334    Female
Name: gender, Length: 2325, dtype: object
```

```
Customer_details_noQ.columns
```

```
Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name',
'hospital_tier_ord',
       'city_tier_ord', 'State_ID_R1013', 'State_ID_R1011',
'State_ID_R1012',
       'age', 'title', 'gender'],
      dtype='object')
```

```
Customer_details_noQ
```

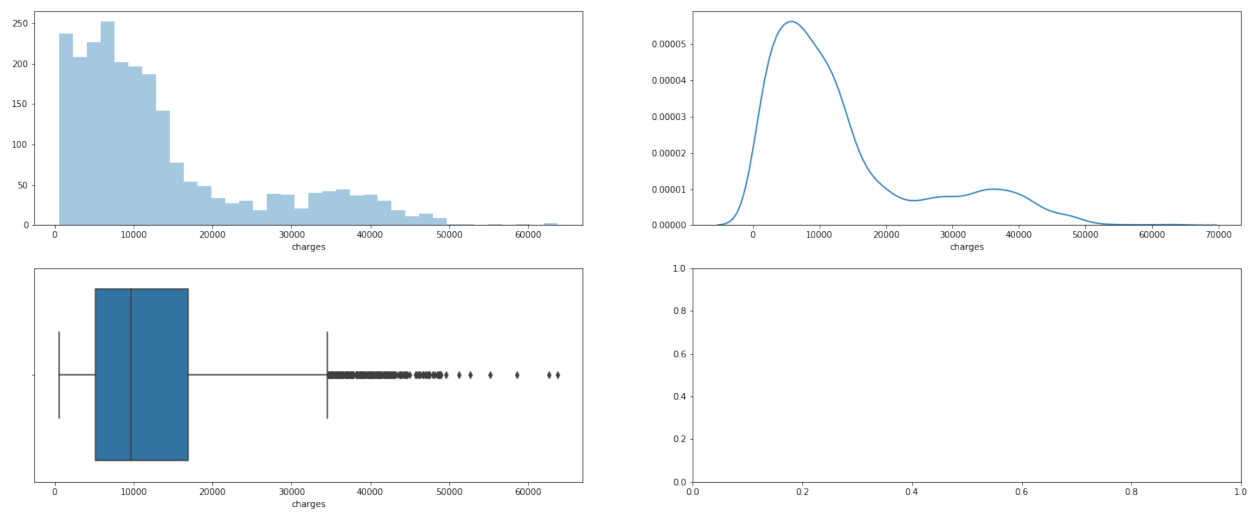
```
     customer_id  year month  date  children   charges
hospital_tier  \
0          Id2335  1992   Jul     9         0    563.84      tier - 2

1          Id2334  1992   Nov    30         0    570.62      tier - 2

2          Id2333  1993   Jun    30         0    600.00      tier - 2

3          Id2332  1992   Sep    13         0    604.54      tier - 3

4          Id2331  1998   Jul    27         0    637.26      tier - 3

...           ...   ...   ...   ...       ...       ...           ...

2329          Id6  1962   Aug     4         0  52590.83      tier - 1
```

```
2330         Id5  1989   Jun    19         0  55135.40      tier - 1

2331         Id4  1991   Jun     6         1  58571.07      tier - 1

2333         Id2  1977   Jun     8         0  62592.87      tier - 2

2334         Id1  1968   Oct    12         0  63770.43      tier - 1


     city_tier state_id     bmi  ...  smoker  \
0     tier - 3    R1013  17.580  ...      No
1     tier - 1    R1013  17.600  ...      No
2     tier - 1    R1013  16.470  ...      No
3     tier - 3    R1013  17.700  ...      No
4     tier - 3    R1013  22.340  ...      No
...        ...      ...     ...  ...     ...
2329  tier - 3    R1011  32.800  ...     yes
2330  tier - 2    R1012  35.530  ...     yes
2331  tier - 3    R1024  38.095  ...     yes
2333  tier - 3    R1013  30.360  ...     yes
2334  tier - 3    R1013  47.410  ...     yes

                                    name hospital_tier_ord
city_tier_ord  \
0                    German, Mr.  Aaron K                 1
2
1                  Rosendahl, Mr.  Evan P                 1
0
2                      Albano, Ms.  Julie                 1
0
3     Riveros Gonzalez, Mr.  Juan D. Sr.                 2
2
4                   Brietzke, Mr.  Jordan                 2
2
...                                  ...               ...            .
..
2329               Baker, Mr.  Russell B.                 0
2
2330                 Kadala, Ms.  Kristyn                 0
1
2331                 Osborne, Ms.  Kelsey                 0
2
2333               Lehner, Mr.  Matthew D                 1
2
2334                    Hawks, Ms.  Kelly                 0
2

      State_ID_R1013 State_ID_R1011 State_ID_R1012  age  title  gender

0                  1              0              0   32     Mr    Male
```

```
1                       1                  0                  0   32      Mr     Male
2                       1                  0                  0   31      Ms   Female
3                       1                  0                  0   32      Mr     Male
4                       1                  0                  0   26      Mr     Male
...                   ...                ...                ...  ...     ...      ...
2329                    0                  1                  0   62      Mr     Male
2330                    0                  0                  1   35      Ms   Female
2331                    0                  0                  0   33      Ms   Female
2333                    1                  0                  0   47      Mr     Male
2334                    1                  0                  0   56      Ms   Female

[2325 rows x 25 columns]
```

9. You should also visualize the distribution of costs using a histogram, box and whisker plot, and swarm plot.

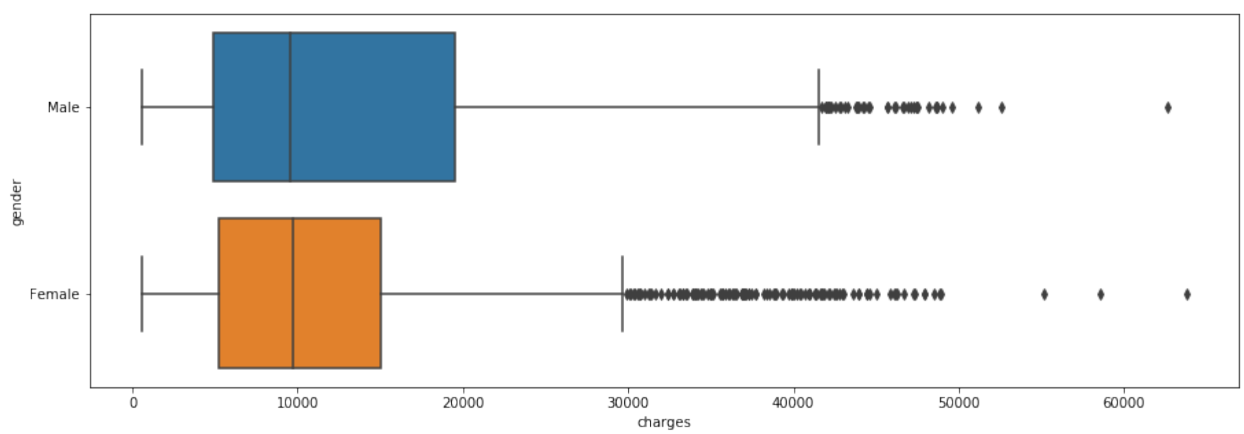
```
fig,ax=plt.subplots(2,2,figsize=[25,10])
sns.distplot(Customer_details_noQ['charges'],hist=True,kde=
False,ax=ax[0][0])
sns.distplot(Customer_details_noQ['charges'],hist=False,kde=
True,ax=ax[0][1])
sns.boxplot(Customer_details_noQ['charges'],ax=ax[1][0])
plt.show()
```



10. State how the distribution is different across gender and tiers of hospitals

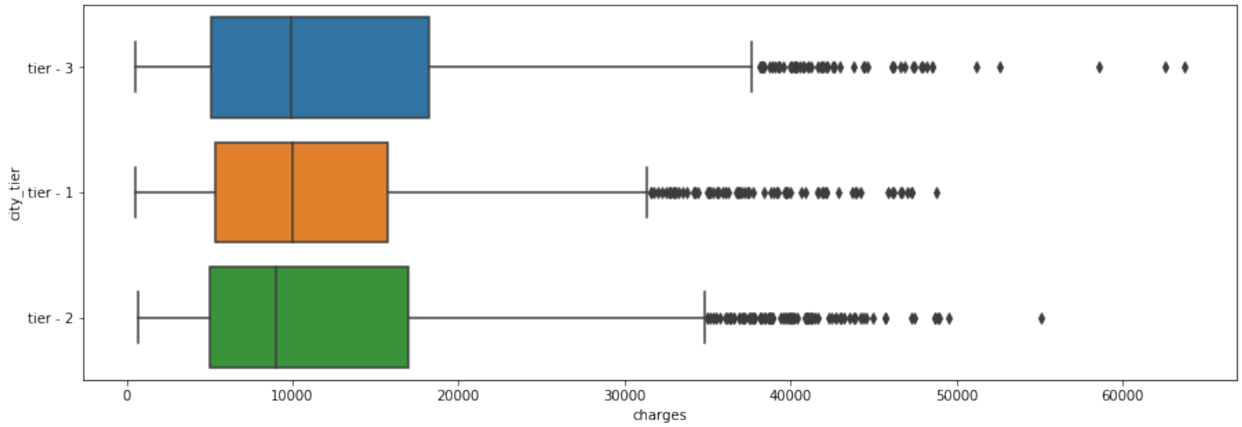
WRT Gender

```
plt.figure(figsize=(15,5))
sns.boxplot(x='charges',y='gender',data=Customer_details_noQ )
plt.show()
```



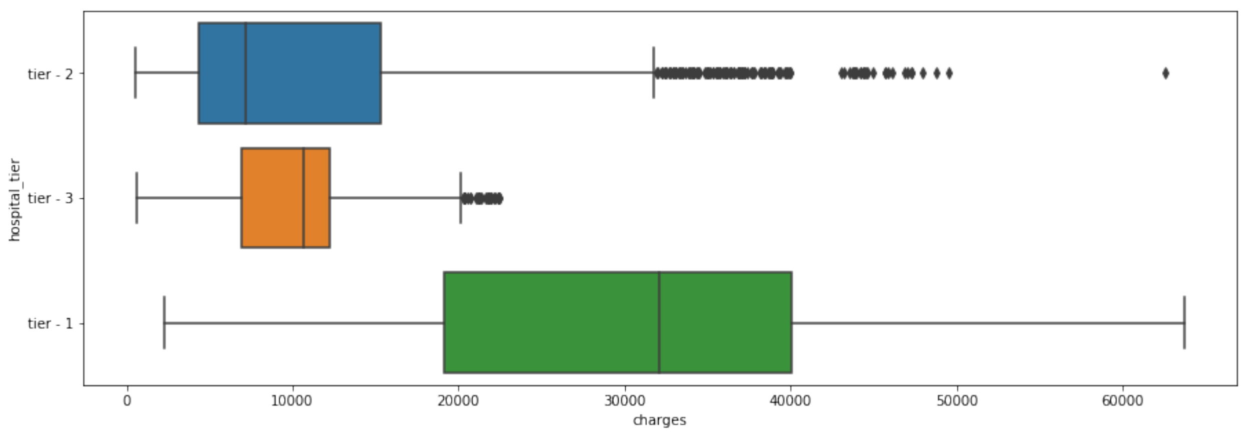
WRT city tier

```
plt.figure(figsize=(15,5))
sns.boxplot(x='charges',y='city_tier',data=Customer_details_noQ )
plt.show()
```



WRT Hospital tier

```
plt.figure(figsize=(15,5))
sns.boxplot(x='charges',y='hospital_tier',data=Customer_details_noQ )
plt.show()
```



11. Create a radar chart to showcase the median hospitalization cost for each tier of hospitals

```
median = Customer_details_noQ.groupby('hospital_tier')
[['charges']].median().reset_index()
median
```

	hospital_tier	charges
0	tier - 1	32097.435
1	tier - 2	7168.760
2	tier - 3	10676.830

```python
fig = px.line_polar(median, r='charges', theta='hospital_tier') #,
line_close=True
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"hovertemplate":"charges=%{r}<br>hospital_tier=%{theta}<extra></
extra>","legendgroup":"","line":
{"color":"#636efa","dash":"solid"},"marker":
{"symbol":"circle"},"mode":"lines","name":"","r":
[32097.434999999998,7168.76,10676.83],"showlegend":false,"subplot":"po
lar","theta":["tier - 1","tier - 2","tier -
3"],"type":"scatterpolar"}],"layout":{"legend":
{"tracegroupgap":0},"margin":{"t":60},"polar":{"angularaxis":
{"direction":"clockwise","rotation":90},"domain":{"x":[0,1],"y":
[0,1]}},"template":{"data":{"bar":[{"error_x":
{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
```

```
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
```

```
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}}}}
```

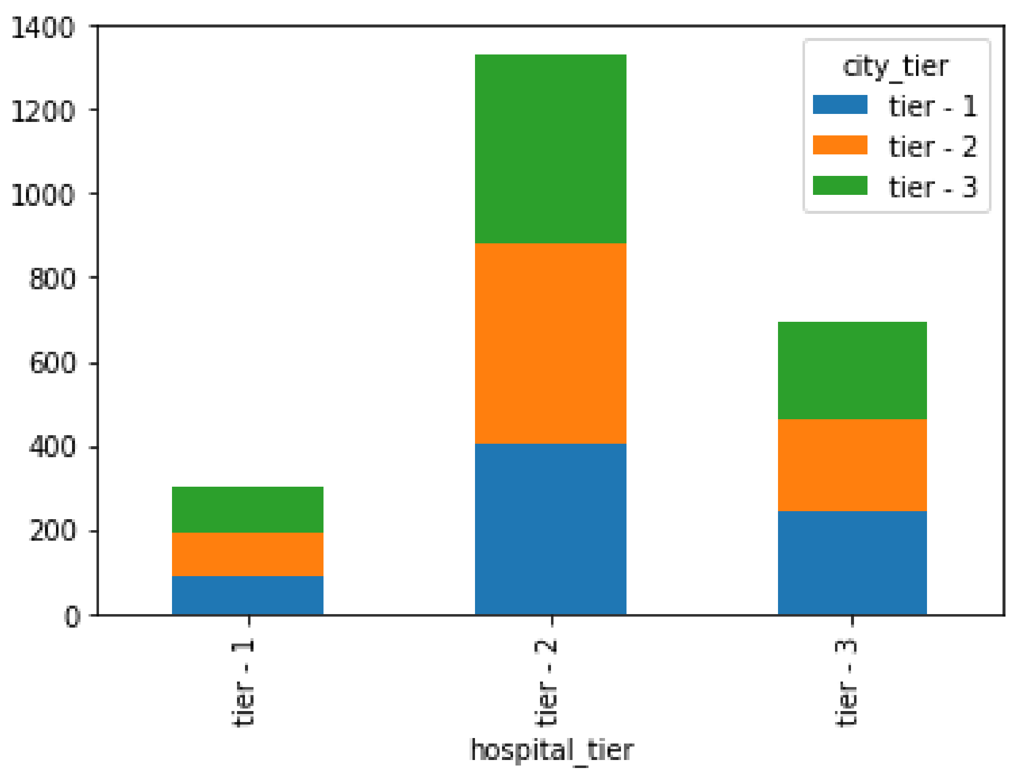
12. Create a frequency table and a stacked bar chart to visualize the count of people in the differenttiers of cities and hospitals

```
pd.crosstab(Customer_details_noQ['hospital_tier'],Customer_details_noQ
['city_tier'])
```

city_tier hospital_tier	tier - 1	tier - 2	tier - 3
tier - 1	85	106	109
tier - 2	403	479	452
tier - 3	241	222	228

```
#plt.figure(figsize=[12,6])
pd.crosstab(Customer_details_noQ['hospital_tier'],Customer_details_noQ
['city_tier']).plot.bar(stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x17cfa20efc8>
```



13. Test the following null hypotheses:

#a. The average hospitalization costs for the three types of hospitals are not significantly different.

#b. The average hospitalization costs for the three types of cities are not significantly different.

#c. The average hospitalization cost for smokers is not significantly different from the average cost for nonsmokers.

#d. Smoking and heart issues are independent.

a. The average hospitalization costs for the three types of hospitals are not significantly different.

```
from scipy.stats import ttest_1samp

Customer_details_noQ.columns

Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name',
'hospital_tier_ord',
       'city_tier_ord', 'State_ID_R1013', 'State_ID_R1011',
'State_ID_R1012',
       'age', 'title', 'gender'],
      dtype='object')
```

Annova Test

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

model = ols('charges ~ hospital_tier',data=Customer_details_noQ).fit()
res = sm.stats.anova_lm(model)
res

                   df        sum_sq       mean_sq           F
PR(>F)
hospital_tier     2.0  9.763011e+10  4.881505e+10  493.989566
1.773822e-179
Residual       2322.0  2.294554e+11  9.881799e+07         NaN
NaN
```

```
looking into pvalue(1.77 ) > alpha(0.005) , we cannot reject null
hypothesis and
can conclude costs for the three types of hospitals are significantly
different.

  File "<ipython-input-65-2e384f72ed92>", line 1
    looking into pvalue(1.77 ) > alpha(0.005) , we cannot reject null
hypothesis and
               ^
SyntaxError: invalid syntax
```

b. The average hospitalization costs for the three types of cities are not significantly different.

```
model = ols('charges ~ city_tier',data=Customer_details_noQ).fit()
res = sm.stats.anova_lm(model)
res

looking into pvalue(0.233 ) > alpha(0.005) , we cannot reject null
hypothesis and
can conclude costs for the three types of cities are significantly
different.
```

c. The average hospitalization cost for smokers is not significantly different from the average cost for nonsmokers.

T-test

```
cost_for_smokers=Customer_details_noQ.loc[Customer_details_noQ.smoker
== 'yes','charges']
cost_for_nonsmokers=Customer_details_noQ.loc[Customer_details_noQ.smok
er != 'yes','charges']
print(cost_for_smokers.count())
print(cost_for_nonsmokers.count())

import numpy as np
from scipy.stats import ttest_ind
```

```
Stat,P_value=ttest_ind(cost_for_smokers,cost_for_nonsmokers)
print('P value to check hypothesis ',P_value)

Looking at the p_value < alpha(0.05), we can reject the null
hypothesis and conclude that
Average hospitalization cost for smokers is significantly different
than non-smokers
```

d. Smoking and heart issues are independent.

Chi-squared test

```
observed =pd.crosstab(Customer_details_noQ.smoker,
Customer_details_noQ.heart_issues)
observed

from scipy.stats import chi2_contingency

# Defining the observed frequencies
#observed = [Customer_details_noQ.smoker,
Customer_details_noQ.heart_issues]  string will not be considered

# Perform chi-squared test
chi2, p, dof, expected = chi2_contingency(observed)

print("Observed frequencies:")
print(observed)
print("Expected frequencies:")
print(expected)
print("Chi-squared statistic:", chi2)
print("p-value:", p)
```

#observation of Chi Test: Pvalue(0.76) > alpha(0.05) cannot reject Null Hypothesis. means smoking and heart issue are indepndent event

-------------------------------- * END OF FIRST PART OF PROJECT * -------

Machine learning

```
Customer_details_noQ.columns
```

```
Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name',
'hospital_tier_ord',
       'city_tier_ord', 'State_ID_R1013', 'State_ID_R1011',
'State_ID_R1012',
       'age', 'title', 'gender'],
      dtype='object')
```

```
Customer_details_noQ.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2325 entries, 0 to 2334
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   customer_id             2325 non-null   object
 1   year                    2325 non-null   int32
 2   month                   2325 non-null   object
 3   date                    2325 non-null   int64
 4   children                2325 non-null   int64
 5   charges                 2325 non-null   float64
 6   hospital_tier           2325 non-null   object
 7   city_tier               2325 non-null   object
 8   state_id                2325 non-null   object
 9   bmi                     2325 non-null   float64
 10  hba1c                   2325 non-null   float64
 11  heart_issues            2325 non-null   object
 12  any_transplants         2325 non-null   object
 13  cancer_history          2325 non-null   object
 14  numberofmajorsurgeries  2325 non-null   float64
 15  smoker                  2325 non-null   object
 16  name                    2325 non-null   object
 17  hospital_tier_ord       2325 non-null   int32
 18  city_tier_ord           2325 non-null   int32
 19  State_ID_R1013          2325 non-null   int64
 20  State_ID_R1011          2325 non-null   int64
 21  State_ID_R1012          2325 non-null   int64
 22  age                     2325 non-null   int32
 23  title                   2325 non-null   object
 24  gender                  2325 non-null   object
dtypes: float64(4), int32(4), int64(5), object(12)
memory usage: 515.9+ KB
```

Problem - 1

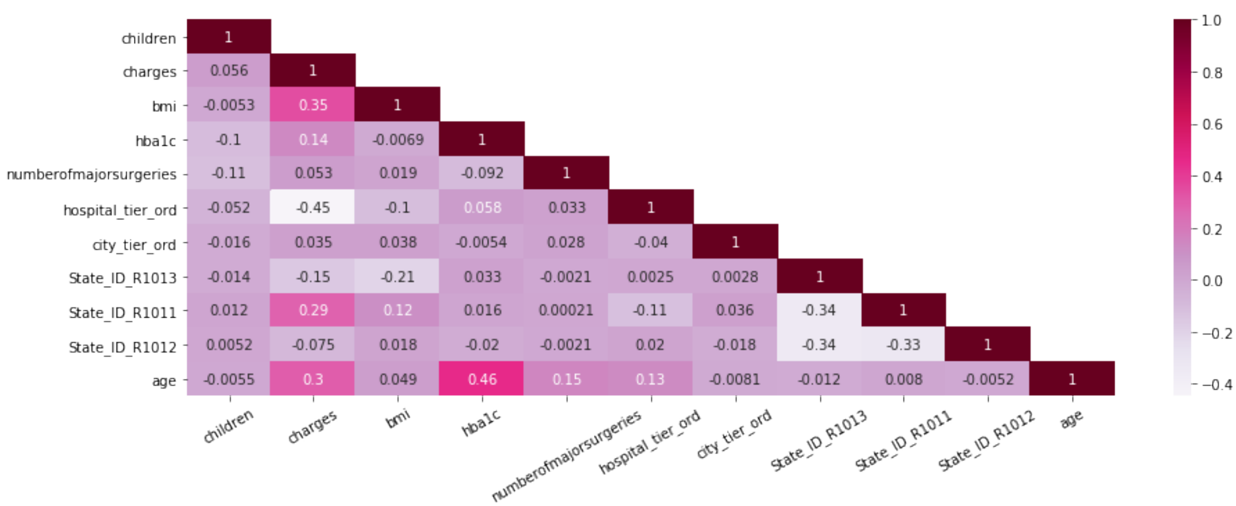
- Examine the correlation between predictors to identify highly correlated predictors

```
data = Customer_details_noQ[[ 'children', 'charges',
       'bmi', 'hba1c','numberofmajorsurgeries', 'hospital_tier_ord',
       'city_tier_ord', 'State_ID_R1013', 'State_ID_R1011',
'State_ID_R1012',
       'age', ]]

data_corr=data.corr()
data_corr
```

```
                        children   charges       bmi     hba1c  \
children                1.000000  0.055901 -0.005339 -0.101379
charges                 0.055901  1.000000  0.346730  0.139697
bmi                    -0.005339  0.346730  1.000000 -0.006920
hba1c                  -0.101379  0.139697 -0.006920  1.000000
numberofmajorsurgeries -0.113161  0.053308  0.018851 -0.091594
hospital_tier_ord      -0.052438 -0.446687 -0.104771  0.057855
city_tier_ord          -0.015760  0.035300  0.038123 -0.005404
State_ID_R1013         -0.013834 -0.150634 -0.208744  0.033453
State_ID_R1011          0.011666  0.286956  0.115671  0.015525
State_ID_R1012          0.005247 -0.074636  0.017939 -0.019513
age                    -0.005457  0.304395  0.049260  0.460558

                        numberofmajorsurgeries  hospital_tier_ord  \
children                             -0.113161          -0.052438
charges                               0.053308          -0.446687
bmi                                   0.018851          -0.104771
hba1c                                -0.091594           0.057855
numberofmajorsurgeries                1.000000           0.033230
hospital_tier_ord                     0.033230           1.000000
city_tier_ord                         0.027937          -0.039755
State_ID_R1013                       -0.002056           0.002455
State_ID_R1011                        0.000208          -0.114685
State_ID_R1012                       -0.002098           0.020272
age                                   0.151442           0.133771

                        city_tier_ord  State_ID_R1013  State_ID_R1011
\
children                    -0.015760       -0.013834        0.011666

charges                      0.035300       -0.150634        0.286956

bmi                          0.038123       -0.208744        0.115671

hba1c                       -0.005404        0.033453        0.015525

numberofmajorsurgeries       0.027937       -0.002056        0.000208

hospital_tier_ord           -0.039755        0.002455       -0.114685

city_tier_ord                1.000000        0.002766        0.036049
```

```
State_ID_R1013                    0.002766        1.000000       -0.341085

State_ID_R1011                    0.036049       -0.341085        1.000000

State_ID_R1012                   -0.018253       -0.340296       -0.327054

age                              -0.008070       -0.011926        0.008022


                        State_ID_R1012       age
children                      0.005247 -0.005457
charges                      -0.074636  0.304395
bmi                           0.017939  0.049260
hba1c                        -0.019513  0.460558
numberofmajorsurgeries       -0.002098  0.151442
hospital_tier_ord             0.020272  0.133771
city_tier_ord                -0.018253 -0.008070
State_ID_R1013               -0.340296 -0.011926
State_ID_R1011               -0.327054  0.008022
State_ID_R1012                1.000000 -0.005229
age                          -0.005229  1.000000
```

```python
plt.figure(figsize=(15,5))
ma = np.ones_like(data_corr)
ma[np.tril_indices_from(ma)]=0
sns.heatmap(data_corr,annot=True,mask=ma,cmap='PuRd')
plt.xticks(rotation=30)
plt.show()
```



Problem - 2

Develop a regression model Linear or Ridge. Evaluate the model with k-fold cross validation.Also, ensure that you apply all the following suggestions:

- Implement the stratified 5-fold cross validationtechnique for both model building and validation
- Utilize effective standardization techniques and hyperparameter tuning
- Incorporate sklearn-pipelines to streamline the workflow
- Apply appropriate regularization techniques to address the bias-variance trade-off
- Create five folds in the data, and introduce a variable to identify the folds
- Develop Gradient Boost model and determine the variable importance scores,and identify the redundant variables

```
# lets first seperate input and output data
data = Customer_details_noQ[['children', 'charges', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'city_tier_ord',
       'hospital_tier_ord', 'State_ID_R1013', 'State_ID_R1011',
'State_ID_R1012','age','gender']]

final_data = pd.get_dummies(data,drop_first=True,dtype='int')

X = final_data.drop(['charges'],axis=1)
y = final_data[['charges']]

X.head()
```

```
   children    bmi  hba1c  numberofmajorsurgeries  city_tier_ord  \
0         0  17.58   4.51                     1.0              2
1         0  17.60   4.39                     1.0              0
2         0  16.47   6.35                     1.0              0
3         0  17.70   6.28                     1.0              2
4         0  22.34   5.57                     1.0              2

   hospital_tier_ord  State_ID_R1013  State_ID_R1011  State_ID_R1012
age  \
0                  1               1               0               0
32
1                  1               1               0               0
32
2                  1               1               0               0
31
3                  2               1               0               0
32
4                  2               1               0               0
26

   heart_issues_yes  any_transplants_yes  cancer_history_Yes
smoker_yes  \
```

```
0                 0                    0                  0
0
1                 0                    0                  0
0
2                 0                    0                  1
0
3                 0                    0                  0
0
4                 0                    0                  0
0

   gender_Male
0            1
1            1
2            0
3            1
4            1
```

```
# setting up a pipe line
from sklearn.linear_model import SGDRegressor,Ridge
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import
KFold,StratifiedKFold,RandomizedSearchCV,GridSearchCV
from sklearn.metrics import mean_squared_error

pipeline = Pipeline(steps=[('scaler',StandardScaler()),
('regression',Ridge())])

# Defining the parameter for hyper parameter tuning
parameters = {'regression__alpha' : [0.001,0.01,0.1,1,10,100]}

# creating k-fold objects
kfold = KFold(n_splits=5,shuffle=True,random_state=3)

# creating the gradient search objects
model_ridge =
GridSearchCV(pipeline,param_grid=parameters,cv=kfold,scoring='neg_mean
_squared_error')

model_ridge.fit(X,y)
```

```
GridSearchCV(cv=KFold(n_splits=5, random_state=3, shuffle=True),
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('regression', Ridge())]),
             param_grid={'regression__alpha': [0.001, 0.01, 0.1, 1,
10, 100]},
             scoring='neg_mean_squared_error')
```

```
model_ridge.best_params_
```

```
{'regression__alpha': 10}
```

```
model_ridge.best_estimator_
```

```
Pipeline(steps=[('scaler', StandardScaler()), ('regression',
Ridge(alpha=10))])
```

Gradient Boosting Algorithms

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y)

model = GradientBoostingRegressor()
model.fit(X_train,y_train)
```

```
GradientBoostingRegressor()
```

```
model.feature_importances_
```

```
array([4.38853904e-03, 1.14916455e-01, 3.89144324e-03, 6.40714004e-04,
       1.54577429e-04, 2.03448674e-02, 4.78820728e-03, 8.91851976e-03,
       3.68398264e-04, 9.87755494e-02, 9.63030890e-05, 0.00000000e+00,
       8.27881010e-05, 7.41043275e-01, 1.59036286e-03])
```

Variable importance

```
pd.DataFrame({'features' : model.feature_names_in_,'importance' :
model.feature_importances_}).sort_values('importance',ascending=False)
```

```
                  features  importance
13              smoker_yes    0.741043
1                      bmi    0.114916
9                      age    0.098776
5        hospital_tier_ord    0.020345
7           State_ID_R1011    0.008919
6           State_ID_R1013    0.004788
0                 children    0.004389
2                    hba1c    0.003891
14             gender_Male    0.001590
3   numberofmajorsurgeries    0.000641
8           State_ID_R1012    0.000368
4            city_tier_ord    0.000155
10        heart_issues_yes    0.000096
12      cancer_history_Yes    0.000083
11     any_transplants_yes    0.000000
```

```
# train_score
model.score(X_train,y_train)
```

```
0.9355591853437756
```

```
# test score
model.score(X_test,y_test)
```

```
0.9132710744585373
```

Problem - 3

Estimate the cost of hospitalization for Christopher, Ms. Jayna (Dateof birth12/28/1988;height170 cm;and weight 85 kgs). She lives with her partner and two children in a tier-1 city,and her state's State ID is R1011.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.

```
Customer_details_noQ.columns
```

```
Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name',
'hospital_tier_ord',
       'city_tier_ord', 'State_ID_R1013', 'State_ID_R1011',
'State_ID_R1012',
       'age', 'title', 'gender'],
      dtype='object')
```

```
pred_data = pd.DataFrame({
    'name':['Christopher, Ms. Jayna'],
    'dob':['12/28/1988'],
    'children':[2],
    'bmi':[85/(1.7**2)],
    'hba1c':[5.8],
    'numberofmajorsurgeries':[0],
    'city_tier_ord':[1],
    'hospital_tier_ord':[1],
    'state_id_R1013':[0],
    'state_id_R1011':[1],
    'state_id_R1012':[0],
    'age':[36],
    'heart_issues_yes':[0],
    'any_transplants_yes':[0],
    'cancer_history_Yes':[1],
    'smoker_yes':[1],
```

```
    'gender_male':[0]

})

X.columns

Index(['children', 'bmi', 'hba1c', 'numberofmajorsurgeries',
'city_tier_ord',
       'hospital_tier_ord', 'State_ID_R1013', 'State_ID_R1011',
       'State_ID_R1012', 'age', 'heart_issues_yes',
'any_transplants_yes',
       'cancer_history_Yes', 'smoker_yes', 'gender_Male'],
      dtype='object')

pred_data

                     name         dob  children        bmi  hba1c  \
0  Christopher, Ms. Jayna  12/28/1988         2  29.411765    5.8

   numberofmajorsurgeries  city_tier_ord  hospital_tier_ord
state_id_R1013  \
0                       0              1                  1
0

   state_id_R1011  state_id_R1012  age  heart_issues_yes
any_transplants_yes  \
0               1               0   36                 0
0

   cancer_history_Yes  smoker_yes  gender_male
0                   1           1            0

pred_data['dob'] = pd.to_datetime(pred_data.dob,errors='coerce')

age=2024 - pred_data.dob.dt.year
age

0    36
Name: dob, dtype: int64

 test_data =pred_data.drop(['name','dob'],axis=1)
```

predicting the charges of test_data using best model

```
model.predict(test_data)

array([27777.69719166])
```

--------- End of Machine Learning Part ---------