In [189…

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
import plotly.express as px
from statsmodels.formula.api import ols
import statsmodels.api  as sm
import scipy.stats as stats
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from sklearn.linear_model import SGDRegressor, Ridge
from sklearn.model_selection import KFold, StratifiedKFold, RandomizedSearchCV, train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error as mse, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
```

In [2]:

```python
hospital = pd.read_csv('Hospitalisation details.csv')
medic = pd.read_csv('Medical Examinations.csv')
names = pd.read_excel('Names.xlsx')
```

Checking Missing Values

In [4]:

```python
hospital.info()
medical.info()
names.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
<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
<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
```

In [5]:

```python
data = pd.merge(hospital, medical, how = 'inner', on = 'Customer ID')
```

In [6]:

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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: 292.0+ KB
```

In [7]:

```python
data = data.merge(names,on='Customer ID')
```

In [8]:

```python
data.head()
```

Out[8]:

	Customer ID	year	month	date	children	charges	Hospital tier	City tier	State ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	name
0	Id2335	1992	Jul	9	0	563.84	tier - 2	tier - 3	R1013	17.58	4.51	No	No	No	1	No	German, Mr. Aaron K
1	Id2334	1992	Nov	30	0	570.62	tier - 2	tier - 1	R1013	17.60	4.39	No	No	No	1	No	Rosendahl, Mr. Evan P
2	Id2333	1993	Jun	30	0	600.00	tier - 2	tier - 1	R1013	16.47	6.35	No	No	Yes	1	No	Albano, Ms. Julie
3	Id2332	1992	Sep	13	0	604.54	tier - 3	tier - 3	R1013	17.70	6.28	No	No	No	1	No	Riveros Gonzalez, Mr. Juan D. Sr.
4	Id2331	1998	Jul	27	0	637.26	tier - 3	tier - 3	R1013	22.34	5.57	No	No	No	1	No	Brietzke, Mr. Jordan

In [9]:

```python
##Replacing caps to lower string and removing spaces from data
```

In [11]:

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

```
data.columns
```

Out[11]:
```
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')
```

In [12]:
```
data.city_tier
```

Out[12]:
```
0       tier - 3
1       tier - 1
2       tier - 1
3       tier - 3
4       tier - 3
          ...
2330    tier - 2
2331    tier - 3
2332    tier - 1
2333    tier - 3
2334    tier - 3
Name: city_tier, Length: 2335, dtype: object
```

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

In [13]:
```
(data == '?').sum()
```

Out[13]:
```
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
```

In [28]:
```
#removing '?' and calculating percentage

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

Out[28]:
```
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
```

In [29]:
```
percentage[percentage>0].index
```

Out[29]:
```
Index([11, 13, 17, 542, 1046, 1049, 1700, 1775, 2165, 2332], dtype='int64')
```

In [30]:
```
percentage_col = (data == '?').sum(axis = 0)/data.shape[0] * 100
percentage_col.sort_values(ascending= False)
```

Out[30]:
```
month                     0.128480
state_id                  0.085653
smoker                    0.085653
year                      0.085653
hospital_tier             0.042827
city_tier                 0.042827
heart_issues              0.000000
numberofmajorsurgeries    0.000000
cancer_history            0.000000
any_transplants           0.000000
customer_id               0.000000
hba1c                     0.000000
bmi                       0.000000
charges                   0.000000
children                  0.000000
date                      0.000000
name                      0.000000
dtype: float64
```

In [33]:
```
master_data = data.drop(index = percentage[percentage>0].index)
master_data.shape
master_data.isna().sum()
```

Out[33]:
```
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
```

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

In [34]:
```
master_data[['city_tier', 'hospital_tier']]
```

Out[34]:

	city_tier	hospital_tier
0	tier - 3	tier - 2
1	tier - 1	tier - 2
2	tier - 1	tier - 2

3	tier - 3	tier - 3
4	tier - 3	tier - 3
...	...	...
2329	tier - 3	tier - 1
2330	tier - 2	tier - 1
2331	tier - 3	tier - 1
2333	tier - 3	tier - 2
2334	tier - 3	tier - 1

2325 rows × 2 columns

In [35]:
```
master_data.state_id.value_counts()
```

Out[35]:
```
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: count, dtype: int64
```

In [40]:
```
# Using ordinalencoder to deal with ordinal categorical variables - city tier and hospital tier

ordinal = OrdinalEncoder(categories= [['tier - 3', 'tier - 2', 'tier - 1'],['tier - 3', 'tier - 2', 'tier - 1']])
master_data[['city_tier_ord','hospital_tier_ord']] = ordinal.fit_transform(master_data[['city_tier', 'hospital_tier']])
pd.crosstab(master_data['city_tier_ord'],master_data['city_tier'])
```

Out[40]:

city_tier	tier - 1	tier - 2	tier - 3
city_tier_ord			
0.0	0	0	789
1.0	0	807	0
2.0	729	0	0

In [41]:
```
pd.crosstab(master_data['hospital_tier_ord'],master_data['hospital_tier'])
```

Out[41]:

hospital_tier	tier - 1	tier - 2	tier - 3
hospital_tier_ord			
0.0	0	0	691
1.0	0	1334	0
2.0	300	0	0

In [42]:
```
master_data.head()
```

Out[42]:

	customer_id	year	month	date	children	charges	hospital_tier	city_tier	state_id	bmi	hba1c	heart_issues	any_transplants	cancer_history	numberofmajorsurgeries	smoker	name	city_tier_ord	hospital_tier_ord
0	Id2335	1992	Jul	9	0	563.84	tier - 2	tier - 3	R1013	17.58	4.51	No	No	No	1	No	German, Mr. Aaron K	0.0	1.0
1	Id2334	1992	Nov	30	0	570.62	tier - 2	tier - 1	R1013	17.60	4.39	No	No	No	1	No	Rosendahl, Mr. Evan P	2.0	1.0
2	Id2333	1993	Jun	30	0	600.00	tier - 2	tier - 1	R1013	16.47	6.35	No	No	Yes	1	No	Albano, Ms. Julie	2.0	1.0
3	Id2332	1992	Sep	13	0	604.54	tier - 3	tier - 3	R1013	17.70	6.28	No	No	No	1	No	Riveros Gonzalez, Mr. Juan D. Sr.	0.0	0.0
4	Id2331	1998	Jul	27	0	637.26	tier - 3	tier - 3	R1013	22.34	5.57	No	No	No	1	No	Brietzke, Mr. Jordan	0.0	0.0

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.

In [43]:
```
values = master_data.state_id.value_counts()
values[:3].index
```

Out[43]:
```
Index(['R1013', 'R1011', 'R1012'], dtype='object', name='state_id')
```

In [44]:
```
for i in values[:3].index:
    var_name = 'state_id_' +i   # create name for the dummy varible
    print(var_name)
    master_data[var_name] = 0    # giving a dummy value 0 to dummy variable
    master_data.loc[master_data.state_id == i,var_name] = 1  # replacing 0 by 1 where state id is equal to category of the dummy variable
```

```
state_id_R1013
state_id_R1011
state_id_R1012
```

In [45]:
```
master_data.state_id.value_counts()
```

Out[45]:
```
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: count, dtype: int64
```

In [48]:
```
# checking the no of records

master_data['state_id_R1013'].value_counts()
```

Out[48]:
```
state_id_R1013
0    1716
1     609
Name: count, dtype: int64
```

In [49]:
```
master_data['state_id_R1012'].value_counts()
```

Out[49]:
```
state_id_R1012
0    1753
1     572
Name: count, dtype: int64
```

In [50]:
```
master_data['state_id_R1011'].value_counts()
```

Out[50]:
```
state_id_R1011
0    1751
1     574
Name: count, dtype: int64
```

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

In [51]:
```
master_data.numberofmajorsurgeries.unique()
```

Out[51]:
```
array(['1', 'No major surgery', '2', '3'], dtype=object)
```

In [52]:
```
master_data.loc[master_data.numberofmajorsurgeries == 'No major surgery','numberofmajorsurgeries' ] = 0
```

In [53]:
```
master_data.numberofmajorsurgeries = master_data.numberofmajorsurgeries.astype(int)
```

In [56]:
```
master_data[['numberofmajorsurgeries']]
```

Out[56]:

	numberofmajorsurgeries
0	1
1	1
2	1
3	1
4	1
...	...
2329	0
2330	0
2331	0
2333	0
2334	0

2325 rows × 1 columns

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

In [57]:
```
master_data.year = master_data.year.astype(int)
```

In [60]:
```
master_data['age'] = 2024 - master_data.year
```

In [63]:
```
master_data[['name','age']].head(3)
```

Out[63]:

	name	age
0	German, Mr. Aaron K	32
1	Rosendahl, Mr. Evan P	32
2	Albano, Ms. Julie	31

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.

In [66]:
```
master_data['title'] = master_data.name.str.split('[,.]').str[1].str.strip()
master_data.title.value_counts()
```

Out[66]:
```
title
Mr     1160
Ms     1023
Mrs     142
Name: count, dtype: int64
```

In [70]:
```
master_data.title.shape
```

Out[70]:
```
(2325,)
```

In [71]:
```
master_data['gender'] = 'female'
master_data.loc[master_data.title == 'Mr', 'gender'] = 'male'
```

In [74]:
```
master_data.loc[master_data.title == 'Mrs'].head(3)
```

Out[74]:

	customer_id	year	month	date	children	charges	hospital_tier	city_tier	state_id	bmi	...	smoker	name	city_tier_ord	hospital_tier_ord	state_id_R1013	state_id_R1011	state_id_R1012	age	title	gender
24	Id2311	2001	Aug	19	0	964.71	tier - 3	tier - 2	R1013	25.19	...	No	Keys, Mrs. Kathleen	1.0	0.0	1	0	0	23	Mrs	female
172	Id2163	2004	Dec	27	0	1863.45	tier - 3	tier - 1	R1025	27.06	...	No	Stanislav, Mrs. Grace H	2.0	0.0	0	0	0	20	Mrs	female
197	Id2138	2004	Jun	12	0	2094.10	tier - 3	tier - 2	R1025	27.74	...	No	Padula, Mrs. Lauren	1.0	0.0	0	0	0	20	Mrs	female

3 rows × 25 columns

In [78]:
```
master_data['gender'].value_counts()
```

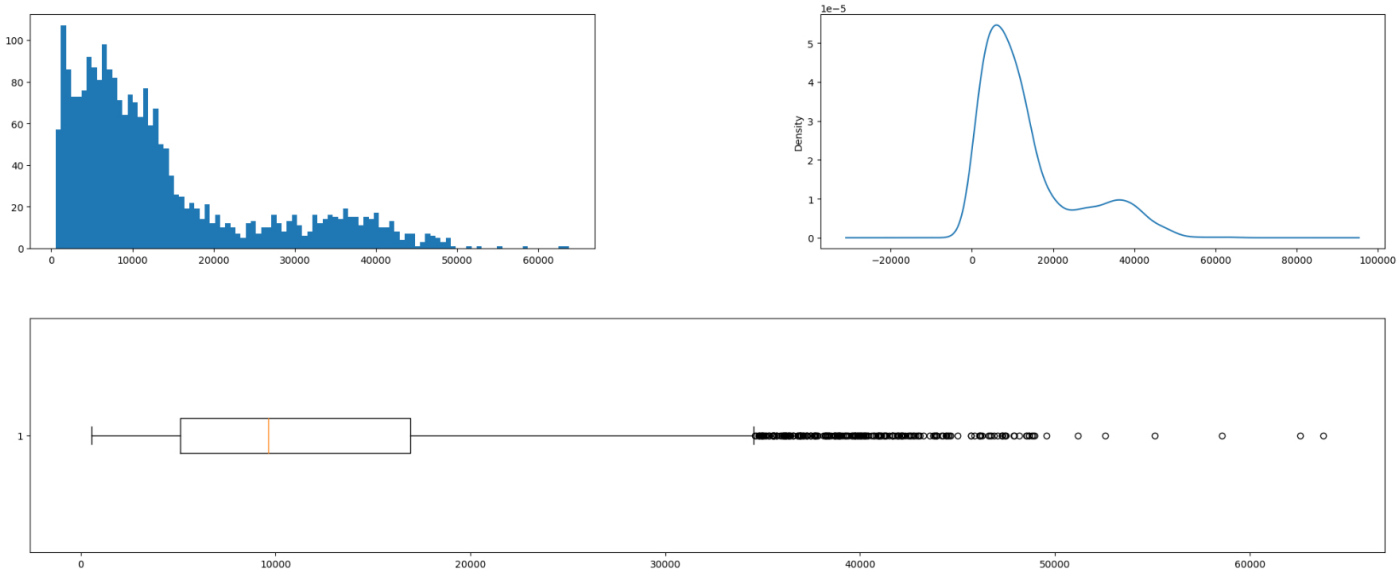
Out[78]:
```
gender
female    1165
male      1160
Name: count, dtype: int64
```

In [79]:
```
1165+1160
```

Out[79]:
```
2325
```

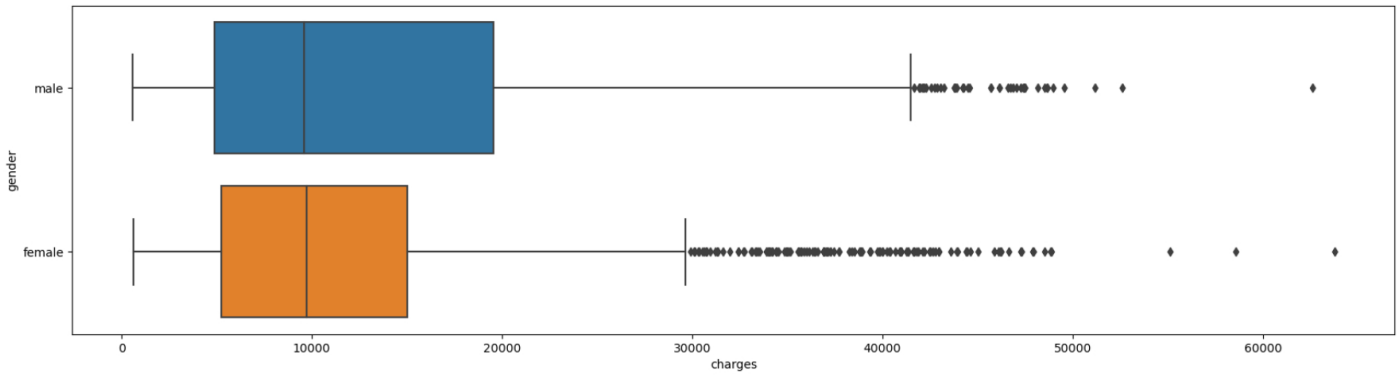
You should also visualize the distribution of costs using a histogram, box and whisker plot, and swarm plot.

In [80]:
```
plt.figure(figsize = (25,10))
grid = plt.GridSpec(2, 2, wspace=0.4, hspace=0.3)
plt.subplot(grid[0, 0])
plt.hist(master_data.charges, bins = 100)
plt.subplot(grid[0, 1])
master_data.charges.plot.kde()
plt.subplot(grid[1, :])
plt.boxplot(master_data.charges, vert = False)
plt.show()
```

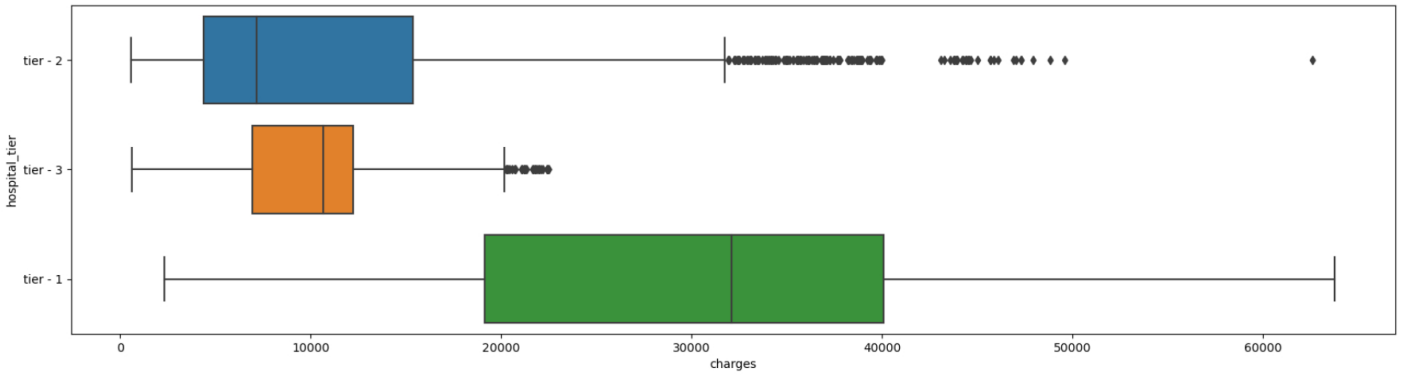


State how the distribution is different across gender and tiers of hospitals

In [87]:
```
#WRT Gender

plt.figure(figsize = (20,5))
sns.boxplot(x = "charges",y = "gender", data = master_data)
plt.show()
```

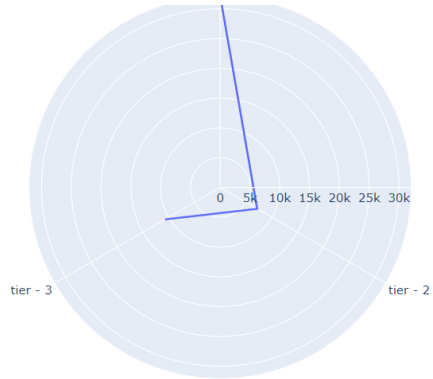


In [91]:
```
#WRT Hospital

plt.figure(figsize = (20,5))
sns.boxplot(x = "charges",y = "hospital_tier", data = master_data)
plt.show()
```



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

In [92]:
```
median = master_data.groupby('hospital_tier')[['charges']].median().reset_index()
fig = px.line_polar(median, r='charges', theta='hospital_tier')
fig.show()
```



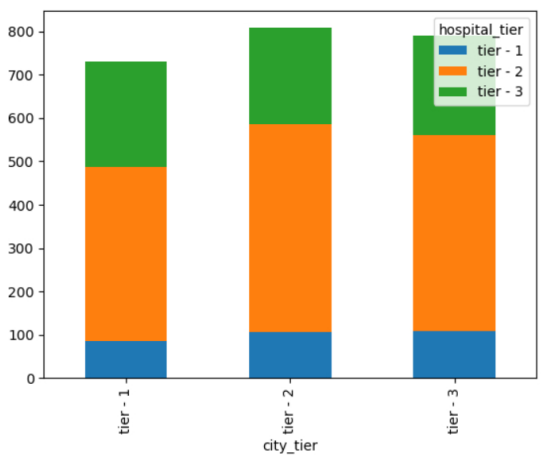
Create a frequency table and a stacked bar chart to visualize the count of people in the different tiers of cities and hospitals

In [99]:
```
#Frequency Table

pd.crosstab(master_data.city_tier, master_data.hospital_tier)
```

Out[99]:

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

In [100…
```
#Visual

pd.crosstab(master_data.city_tier, master_data.hospital_tier).plot.bar(stacked = True)
plt.show()
```



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.

In [101…
```
#The average hospitalization costs for the three types of hospitals are not significantly different.

mod = ols('charges ~ hospital_tier', data = master_data).fit()
res = sm.stats.anova_lm(mod)
res
```

Out[101]:

	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

In [102…
```
#The average hospitalization costs for the three types of cities are not significantly different.

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

Out[102]:

	df	sum_sq	mean_sq	F	PR(>F)
city_tier	2.0	4.092192e+08	2.046096e+08	1.454356	0.233763
Residual	2322.0	3.266763e+11	1.406874e+08	NaN	NaN

In [103…
```
#The average hospitalization cost for smokers is not significantly different from the average cost for nonsmokers.

sample1 = master_data.loc[master_data.smoker == 'yes', 'charges']
sample2 = master_data.loc[master_data.smoker != 'yes', 'charges']
stats.ttest_ind(sample1, sample2)
```

Out[103…
```
TtestResult(statistic=74.15560699695726, pvalue=0.0, df=2323.0)
```

Out[103]:

In [104…
```
#Smoking and heart issues are independent.

observed = pd.crosstab(master_data.smoker, data.heart_issues)
observed
```

Out[104]:

heart_issues	No	yes
smoker		
No	1108	731
yes	297	189

In [107…
```
chi, p, df, expected = stats.chi2_contingency(observed)
chi, p, df, expected
```

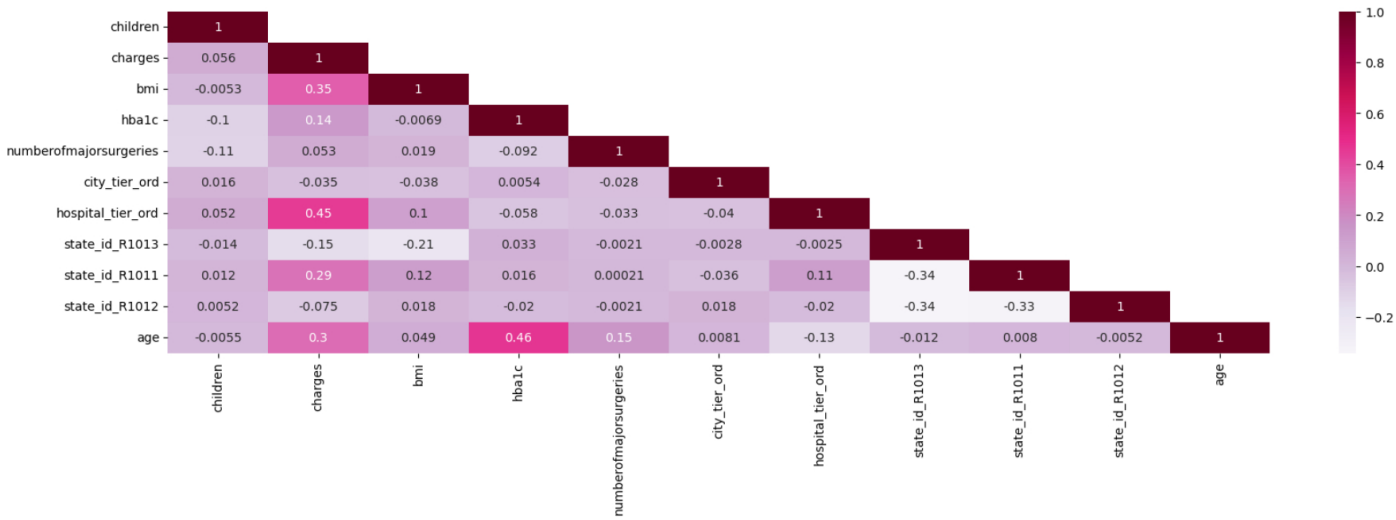
Out[107]:
```
(0.08588150449910657,
 0.7694797581780767,
 1,
 array([[1111.30967742,  727.69032258],
        [ 293.69032258,  192.30967742]]))
```

Examine the correlation between predictors to identify highly correlated predictors

In [108…
```
new_data = master_data.drop(columns = ['customer_id','name', 'year', 'month', 'date','hospital_tier', 'city_tier', 'state_id' , 'title'])
```

In [109…
```
corr_plot = new_data.select_dtypes(exclude='object').corr()
ma = np.ones_like(corr_plot)
ma[np.tril_indices_from(ma)] = 0
```

In [110…
```
plt.figure(figsize = (20,5))
sns.heatmap(corr_plot, annot= True , mask = ma, cmap='PuRd')
plt.show()
```



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

In [111…
```
# Implement the stratified 5-fold cross validation technique 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
```

In [113…
```
data_2 = pd.get_dummies(new_data, drop_first=True)
data_2.reset_index(drop=True, inplace = True)
data_2.head()
```

Out[113]:

	children	charges	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
0	0	563.84	17.58	4.51	1	0.0	1.0	1	0	0	32	False	False	False	False	True
1	0	570.62	17.60	4.39	1	2.0	1.0	1	0	0	32	False	False	False	False	True
2	0	600.00	16.47	6.35	1	2.0	1.0	1	0	0	31	False	False	True	False	False
3	0	604.54	17.70	6.28	1	0.0	0.0	1	0	0	32	False	False	False	False	True
4	0	637.26	22.34	5.57	1	0.0	0.0	1	0	0	26	False	False	False	False	True

In [114…
```
# rearrange data to put 'charges' as first column or last

model_data = data_2.drop(columns = 'charges')
model_data['charges'] = data_2.charges
model_data.head()
```

Out[114]:

	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	charges
0	0	17.58	4.51	1	0.0	1.0	1	0	0	32	False	False	False	False	True	563.84
1	0	17.60	4.39	1	2.0	1.0	1	0	0	32	False	False	False	False	True	570.62
2	0	16.47	6.35	1	2.0	1.0	1	0	0	31	False	False	True	False	False	600.00
3	0	17.70	6.28	1	0.0	0.0	1	0	0	32	False	False	False	False	True	604.54
4	0	22.34	5.57	1	0.0	0.0	1	0	0	26	False	False	False	False	True	637.26

In [115…
```
model_data.columns = model_data.columns.str.lower()
```

In [117…
```
model_data.head(3)
```

Out[117]:

	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	charges
0	0	17.58	4.51	1	0.0	1.0	1	0	0	32	False	False	False	False	True	563.84
1	0	17.60	4.39	1	2.0	1.0	1	0	0	32	False	False	False	False	True	570.62
2	0	16.47	6.35	1	2.0	1.0	1	0	0	31	False	False	True	False	False	600.00

In [120…

```
y = model_data['charges']
X = model_data.drop(columns = 'charges')

X.head()
```

Out[120]:

	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
0	0	17.58	4.51	1	0.0	1.0	1	0	0	32	False	False	False	False	True
1	0	17.60	4.39	1	2.0	1.0	1	0	0	32	False	False	False	False	True
2	0	16.47	6.35	1	2.0	1.0	1	0	0	31	False	False	True	False	False
3	0	17.70	6.28	1	0.0	0.0	1	0	0	32	False	False	False	False	True
4	0	22.34	5.57	1	0.0	0.0	1	0	0	26	False	False	False	False	True

In [131…

```
pipeline = Pipeline(steps=[
    ('scaler', StandardScaler()),   # Scaling step
    ('regressor', Ridge())          # Regression step
])
```

In [132…

```
# Defining the parameters for hyperparameter tuning

parameters = {'regressor__alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
```

In [133…

```
# Creating the KFold object

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

In [134…

```
# Creating the grid search object
model_ridge = GridSearchCV(pipeline, parameters, cv=kfold, scoring='neg_mean_squared_error')
```

In [135…

```
model_ridge.fit(X, y)
```

Out[135]:

```
▸ GridSearchCV
▸ estimator: Pipeline
  ▸ StandardScaler
    ▸ Ridge
```

In [136…

```
# Getting the best parameters and the best model
model_ridge.best_params_
```

Out[136]:

```
{'regressor__alpha': 10}
```

In [137…

```
model_ridge.best_estimator_
```

Out[137]:

```
▸ Pipeline
  ▸ StandardScaler
    ▸ Ridge
```

In [138…

```
from sklearn.ensemble import GradientBoostingRegressor
```

In [139…

```
# Split the data into training and testing sets

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

# Train the XGBoost model
model = GradientBoostingRegressor()
model.fit(X_train, y_train)

print(model.feature_importances_)
```

```
[6.61255527e-03 1.21201186e-01 4.59497256e-03 5.45111980e-04
 5.59347330e-05 1.88461314e-02 5.58419814e-03 6.69662205e-03
 1.08502509e-04 9.09314349e-02 1.64282667e-04 0.00000000e+00
 1.06052378e-05 7.44398783e-01 2.49679220e-04]
```

In [140…

```
pd.DataFrame({'Features':model.feature_names_in_,'Importance':model.feature_importances_}).sort_values("Importance",ascending=False)
```

Out[140]:

	Features	Importance
13	smoker_yes	0.744399
1	bmi	0.121201
9	age	0.090931
5	hospital_tier_ord	0.018846
7	state_id_r1011	0.006697
0	children	0.006613
6	state_id_r1013	0.005584
2	hba1c	0.004595
3	numberofmajorsurgeries	0.000545
14	gender_male	0.000250
10	heart_issues_yes	0.000164
8	state_id_r1012	0.000109
4	city_tier_ord	0.000056
12	cancer_history_yes	0.000011
11	any_transplants_yes	0.000000

In [143…

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

Out[143]:

```
0.9348634134694542
```

In [144…

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

Out[144]:

```
0.9083580164956976
```

Case scenario: Estimate the cost of hospitalization for Christopher, Ms. Jayna (Date of birth 12/28/1988; height 170 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.

In [145…
```
model_data.columns
```

Out[145]:
```
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', 'charges'],
      dtype='object')
```

In [173…
```
pred_data = pd.DataFrame({'Name' : ['Christopher, Ms. Jayna'],
                        'DOB' : ['12/28/1988'],
                        'city_tier' : ['tier - 1'], 'children' :[ 2],
                         'HbA1c' : [5.8],
                         'smoker_yes' : [1],
                         'heart_issues_yes' : [0],
                         'any_transplants_yes' : [0],
                         'numberofmajorsurgeries' :[ 0],
                         'cancer_history_yes' : [1],
                         'hospital_tier' : ['tier - 1'],
                         'bmi' : [85/(1.70 **2)],
                         'state_id_R1011' : [1]
                        })
```

In [174…
```
pred_data.columns = pred_data.columns.str.lower()
```

In [175…
```
pred_data
```

Out[175]:

	name	dob	city_tier	children	hba1c	smoker_yes	heart_issues_yes	any_transplants_yes	numberofmajorsurgeries	cancer_history_yes	hospital_tier	bmi	state_id_r1011
0	Christopher, Ms. Jayna	12/28/1988	tier - 1	2	5.8	1	0	0	0	1	tier - 1	29.411765	1

In [176…
```
print(pred_data)
```

```
                     name         dob city_tier  children  hba1c  smoker_yes  \
0  Christopher, Ms. Jayna  12/28/1988  tier - 1         2    5.8           1

   heart_issues_yes  any_transplants_yes  numberofmajorsurgeries  \
0                 0                    0                       0

   cancer_history_yes hospital_tier        bmi  state_id_r1011
0                   1      tier - 1  29.411765               1
```

In [177…
```
pred_data['gender_male'] = 0
```

In [179…
```
pred_data.loc[pred_data.name.str.split('[,.]').str[1] == 'Mr', 'gender_male'] = 1
pred_data.drop(columns = 'name', inplace = True)
```

In [180…
```
pred_data
```

Out[180]:

	dob	city_tier	children	hba1c	smoker_yes	heart_issues_yes	any_transplants_yes	numberofmajorsurgeries	cancer_history_yes	hospital_tier	bmi	state_id_r1011	gender_male
0	12/28/1988	tier - 1	2	5.8	1	0	0	0	1	tier - 1	29.411765	1	0

In [181…
```
pred_data.drop(columns = 'dob', inplace = True)
```

In [182…
```
pred_data[['city_tier_ord', 'hospital_tier_ord']] = ordinal.transform(pred_data[['city_tier', 'hospital_tier']])
```

In [183…
```
pred_data.drop(columns =['city_tier', 'hospital_tier'], inplace = True )
```

In [184…
```
for col in model_data.columns:
    if col not in pred_data.columns and col != 'charges':
        pred_data[col] = 0
```

In [185…
```
pred_data
```

Out[185]:

	children	hba1c	smoker_yes	heart_issues_yes	any_transplants_yes	numberofmajorsurgeries	cancer_history_yes	bmi	state_id_r1011	gender_male	city_tier_ord	hospital_tier_ord	state_id_r1013	state_id_r1012	age
0	2	5.8	1	0	0	0	1	29.411765	1	0	2.0	2.0	0	0	0

In [186…
```
model_data.columns
```

Out[186]:
```
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', 'charges'],
      dtype='object')
```

In [188…
```
pred_data=pred_data[model_data.drop(columns='charges').columns]
model.predict(pred_data)
```

Out[188]:
```
array([22772.13304111])
```