Regression model to predict the cost to hospital

Agenda

1. Dataset Description

This dataset contains multiple fetures that depicts the health condition of a patient arrived to hospital for a treatment. It has features as below

- Age
- Gender
- Marital Statuts
- Patient height, weight
- Patient current health condition like BP,HB and present and past compli ants
- Any implant done and
- The total cost to the hospital

2. Package importing and loading the dataset

In [105]:

```
# Firstly we import the basic libraries to read the input data set.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [106]:

```
# Firstly we import the basic libraries to read the input data set.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [107]:

```
# Visualize the first 5 rows starting with row #0 to row#4 to ensure the data has
hospital=pd.read_csv("/Users/saisankar/Downloads/Case_Study/Hospital.csv")
hospital.head()
```

Out[107]:

	SL.	AGE	GENDER	MARITAL STATUS	KEY COMPLAINTS -CODE	BODY WEIGHT	BODY HEIGHT	HR PULSE	BP - HIGH	BP- LOW	...
0	1	58.0	M	MARRIED	other- heart	49.2	160	118	100.0	80.0	...
1	2	59.0	M	MARRIED	CAD-DVD	41.0	155	78	70.0	50.0	...
2	3	82.0	M	MARRIED	CAD-TVD	46.6	164	100	110.0	80.0	...
3	4	46.0	M	MARRIED	CAD-DVD	80.0	173	122	110.0	80.0	...
4	5	60.0	M	MARRIED	CAD-DVD	58.0	175	72	180.0	100.0	...

5 rows × 24 columns

3. Exploratory Data Analysis

Let us do the EDA (Exploratory Data Analysis)

#Below steps were performed as part of EDA

#A rounded value were considered which are less than 1 for features like AGE

#Redundant columns which are giving same details from other columns were Dropped.

#NAN , NULL Values have b een filled either with the average or the ideal health condition values. values

#Categorical features were converted to numeric values

#For categorical features where only 2 values are expected like MARITAL STATUS have been converted to 1 and 0 using Lambda

In [108]:

```
# we observe that the column SL. has no significance and has no dependency with t
#hence droping that column

hospital.drop(['SL.'],axis=1,inplace=True)
hospital.head()
```

Out[108]:

	AGE	GENDER	MARITAL STATUS	KEY COMPLAINTS -CODE	BODY WEIGHT	BODY HEIGHT	HR PULSE	BP - HIGH	BP- LOW	RR	...
0	58.0	M	MARRIED	other- heart	49.2	160	118	100.0	80.0	32	...
1	59.0	M	MARRIED	CAD-DVD	41.0	155	78	70.0	50.0	28	...
2	82.0	M	MARRIED	CAD-TVD	46.6	164	100	110.0	80.0	20	...
3	46.0	M	MARRIED	CAD-DVD	80.0	173	122	110.0	80.0	24	...
4	60.0	M	MARRIED	CAD-DVD	58.0	175	72	180.0	100.0	18	...

5 rows × 23 columns

In [109]:

```
#Further it is observed that the feature Age has a decimal value and few values a
#to be able to consider the feature AGE into the model we need to round it to 1 f
#integer part only for the rest. Similarly for other numeric columns
hospital['AGE'] = hospital['AGE'].apply(lambda x: 1  if x < 1 else x)
hospital['AGE'] =(hospital.AGE.astype(int))
```

In [110]:

```
#we do not need the feature Total Length of stay as we can get the same data by a
#Hence dropping the total length of stay as it is redundant variable
# Similarly Implant Used can also be dropped as the cost of Implant value==0 impl
hospital.drop(['TOTAL LENGTH OF STAY'],axis=1,inplace=True)
hospital.drop(['IMPLANT USED (Y/N)'],axis=1,inplace=True)
```

In [111]:

```
# split train test, fit model, predict the model
hospital.head()
```

Out[111]:

	AGE	GENDER	MARITAL STATUS	KEY COMPLAINTS -CODE	BODY WEIGHT	BODY HEIGHT	HR PULSE	BP - HIGH	BP- LOW	RR	...
0	58	M	MARRIED	other- heart	49.2	160	118	100.0	80.0	32	...
1	59	M	MARRIED	CAD-DVD	41.0	155	78	70.0	50.0	28	...
2	82	M	MARRIED	CAD-TVD	46.6	164	100	110.0	80.0	20	...
3	46	M	MARRIED	CAD-DVD	80.0	173	122	110.0	80.0	24	...
4	60	M	MARRIED	CAD-DVD	58.0	175	72	180.0	100.0	18	...

5 rows × 21 columns

In [112]:

```
#Filling the NULL values for the categorical and string variables.
hospital['KEY COMPLAINTS -CODE'].fillna("NO COMPLAINT", inplace = True)
hospital['PAST MEDICAL HISTORY CODE'].fillna("NO HISTORY", inplace = True)
```

```
#Filling the NULL values for HB, Low-BP, HIgh-BP with the ideal values as these c
#For UREA and CREATININE  men and women have a different ranges, an average value
hospital['BP-LOW'].fillna("80", inplace = True)
hospital['BP -HIGH'].fillna("120", inplace = True)
hospital['HB'].fillna("15", inplace = True)
hospital['UREA'].fillna("26", inplace = True)
hospital['CREATININE'].fillna("0.7", inplace = True)
```

In [113]:

```
hospital.describe()
# There are no NULL values in any of the features.
```

Out[113]:

	AGE	BODY WEIGHT	BODY HEIGHT	HR PULSE	RR	LENGTH OF STAY - ICU	LENGTH OF STAY-WARD
count	248.000000	248.000000	248.000000	248.000000	248.000000	248.000000	248.000000
mean	28.891129	37.524677	130.221774	92.229839	23.540323	3.475806	8.153226
std	25.887022	23.118822	39.170901	20.308740	3.840756	3.853520	3.755793
min	1.000000	2.020000	19.000000	41.000000	12.000000	0.000000	0.000000
25%	6.000000	15.000000	105.000000	78.000000	22.000000	1.000000	6.000000
50%	15.500000	40.900000	147.500000	90.000000	24.000000	2.000000	7.000000
75%	55.000000	58.250000	160.000000	104.000000	24.000000	4.000000	10.000000
max	88.000000	85.000000	185.000000	155.000000	42.000000	30.000000	22.000000

In [114]:

```
hospital.shape
```

Out[114]:

```
(248, 21)
```

4. Feature Engineering

#scikit-learn expects all features to be numeric. So how do we include a categorical feature in our model?

#Ordered categories: transform them to sensible numeric values (example: small=1, medium=2, large=3)

#Unordered categories: use dummy encoding (0/1) What are the categorical features in our dataset? Below are the Unordered categories:

MARITAL STATUS, KEY COMPLAINTS -CODE, PAST MEDICAL HISTORY CODE, GENDER, MODE OF ARRIVAL, STATE AT THE TIME OF ARRIVAL, TYPE OF ADMSN

In [115]:

```
## Renaming few feature names to be convinient.

hospital.rename(columns={'KEY COMPLAINTS -CODE':'COMPLAINTS'}, inplace=True)
hospital.rename(columns={'MARITAL STATUS':'MARRIED'}, inplace=True)
hospital.rename(columns={'PAST MEDICAL HISTORY CODE':'PAST_HISTORY'}, inplace=Tru
hospital.rename(columns={'STATE AT THE TIME OF ARRIVAL':'ARRIVAL_STATE'}, inplace
hospital.rename(columns={'MODE OF ARRIVAL':'ARRIVAL_MODE'}, inplace=True)
hospital.rename(columns={'TYPE OF ADMSN':'ADMISSION'}, inplace=True)
hospital.rename(columns={'TOTAL COST TO HOSPITAL ':'TOTAL_COST'}, inplace=True)
## convert the cateogrical variables to Numerical values.


hospital['MARRIED'] = hospital['MARRIED'].apply(lambda x: 1  if x=='MARRIED' else
hospital['GENDER'] = hospital['GENDER'].apply(lambda x: 1  if x=='M' else 0)
hospital['ADMISSION'] = hospital['ADMISSION'].apply(lambda x: 1  if x=='EMERGENCY
hospital['ARRIVAL_STATE'] = hospital['ARRIVAL_STATE'].apply(lambda x: 1  if x=='A
hospital['COMPLAINTS'] = hospital['COMPLAINTS'].apply(lambda x: 0  if x=='NO COMP
hospital['PAST_HISTORY'] = hospital['PAST_HISTORY'].apply(lambda x: 0  if x=='NO
##Creating dummy variables.

HISTORY_dummies =pd.get_dummies(hospital.PAST_HISTORY, prefix='HISTORY')
COMPLAINTS_dummies = pd.get_dummies(hospital.COMPLAINTS, prefix='COMPLAINTS')
ARRIVAL_dummies = pd.get_dummies(hospital.ARRIVAL_MODE, prefix='MODE')

HISTORY_dummies.sample(n=5, random_state=1)
#COMPLAINTS_dummies.sample(n=5, random_state=1)
#ARRIVAL_dummies.sample(n=5, random_state=1)
```

Out[115]:

	HISTORY_0	HISTORY_1
67	1	0
247	1	0
210	1	0
224	1	0
90	1	0

In [116]:

```
hospital.shape
```

Out[116]:

```
(248, 21)
```

In [117]:

```
COMPLAINTS_dummies.sample(n=5, random_state=1)
```

Out[117]:

	COMPLAINTS_0	COMPLAINTS_1
67	1	0
247	0	1
210	0	1
224	0	1
90	0	1

In [118]:

```
ARRIVAL_dummies.sample(n=20, random_state=1)
```

Out[118]:

	MODE_AMBULANCE	MODE_TRANSFERRED	MODE_WALKED IN
67	0	0	1
247	0	0	1
210	0	0	1
224	0	0	1
90	0	0	1
222	0	0	1
58	0	0	1
127	0	0	1
179	0	0	1
4	1	0	0
78	0	0	1
85	0	0	1
95	0	0	1
112	0	0	1
174	0	0	1
160	0	0	1
183	0	0	1
51	0	1	0
27	1	0	0
73	0	0	1

In [119]:

```
hospital.head(16)
```

Out[119]:

	AGE	GENDER	MARRIED	COMPLAINTS	BODY WEIGHT	BODY HEIGHT	HR PULSE	BP - HIGH	BP-LOW	RR	..
0	58	1	1	1	49.2	160	118	100	80	32	..
1	59	1	1	1	41.0	155	78	70	50	28	..
2	82	1	1	1	46.6	164	100	110	80	20	..
3	46	1	1	1	80.0	173	122	110	80	24	..
4	60	1	1	1	58.0	175	72	180	100	18	..
5	75	1	1	1	45.0	140	130	215	140	42	..
6	73	1	1	1	60.0	170	108	160	90	24	..
7	71	1	1	1	43.8	164	60	130	90	22	..
8	72	1	1	1	72.0	174	95	100	50	25	..
9	61	1	1	1	76.6	175	66	140	90	22	..
10	61	1	1	1	64.0	170	99	140	80	24	..
11	45	0	1	1	50.0	151	60	110	60	19	..
12	40	1	1	1	71.4	165	100	110	70	22	..
13	64	1	1	0	56.0	168	105	130	80	22	..
14	68	0	0	1	51.0	123	66	120	80	20	..
15	78	0	1	1	70.0	154	63	150	90	20	..

16 rows × 21 columns

5. Data Visualization

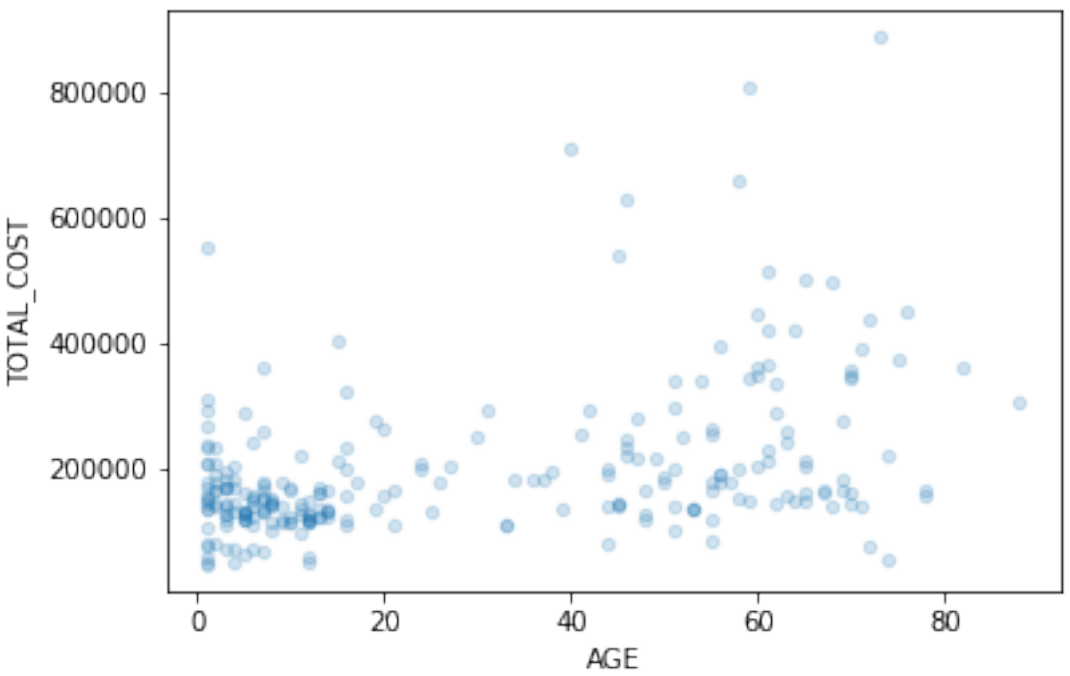
Let us plot a scatter plot between AGE and cost to hospital

In [120]:

```
hospital.plot(kind='scatter', x='AGE', y='TOTAL_COST', alpha=0.2)
```

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b18ecc0>
```



Observation :

From the above scatter plot it is evident that the age is directly proportional to the cost and similarly other attributes are also directly dependent on the cost, hence we can use a linear regression model to predict the total cost

In [121]:

```
hospital = pd.concat([hospital,ARRIVAL_dummies], axis=1)
```

In [122]:

```
hospital.head(5)
```

Out[122]:

	AGE	GENDER	MARRIED	COMPLAINTS	BODY WEIGHT	BODY HEIGHT	HR PULSE	BP - HIGH	BP- LOW	RR	...
0	58	1	1	1	49.2	160	118	100	80	32	...
1	59	1	1	1	41.0	155	78	70	50	28	...
2	82	1	1	1	46.6	164	100	110	80	20	...
3	46	1	1	1	80.0	173	122	110	80	24	...
4	60	1	1	1	58.0	175	72	180	100	18	...

5 rows × 24 columns

As we created the dummy variables for complaints, past history and mode of arrival. We also concatenated these dummy variables to the dataset, hence we need to remove the actual categorical variables from the dataset.

In [123]:

```
hospital.drop(['ARRIVAL_MODE'],axis=1,inplace=True)
```

In [124]:

```
hospital.head(5)
```

Out[124]:

	AGE	GENDER	MARRIED	COMPLAINTS	BODY WEIGHT	BODY HEIGHT	HR PULSE	BP - HIGH	BP-LOW	RR	...
0	58	1	1	1	49.2	160	118	100	80	32	...
1	59	1	1	1	41.0	155	78	70	50	28	...
2	82	1	1	1	46.6	164	100	110	80	20	...
3	46	1	1	1	80.0	173	122	110	80	24	...
4	60	1	1	1	58.0	175	72	180	100	18	...

5 rows × 23 columns

Now let us see the correlation or the multicolinearity among the attributes.

In [125]:

```
hospital.shape
```

Out[125]:

```
(248, 23)
```

In [126]:

```
corr=hospital.corr()
```

In [127]:

```
corr
```

Out[127]:

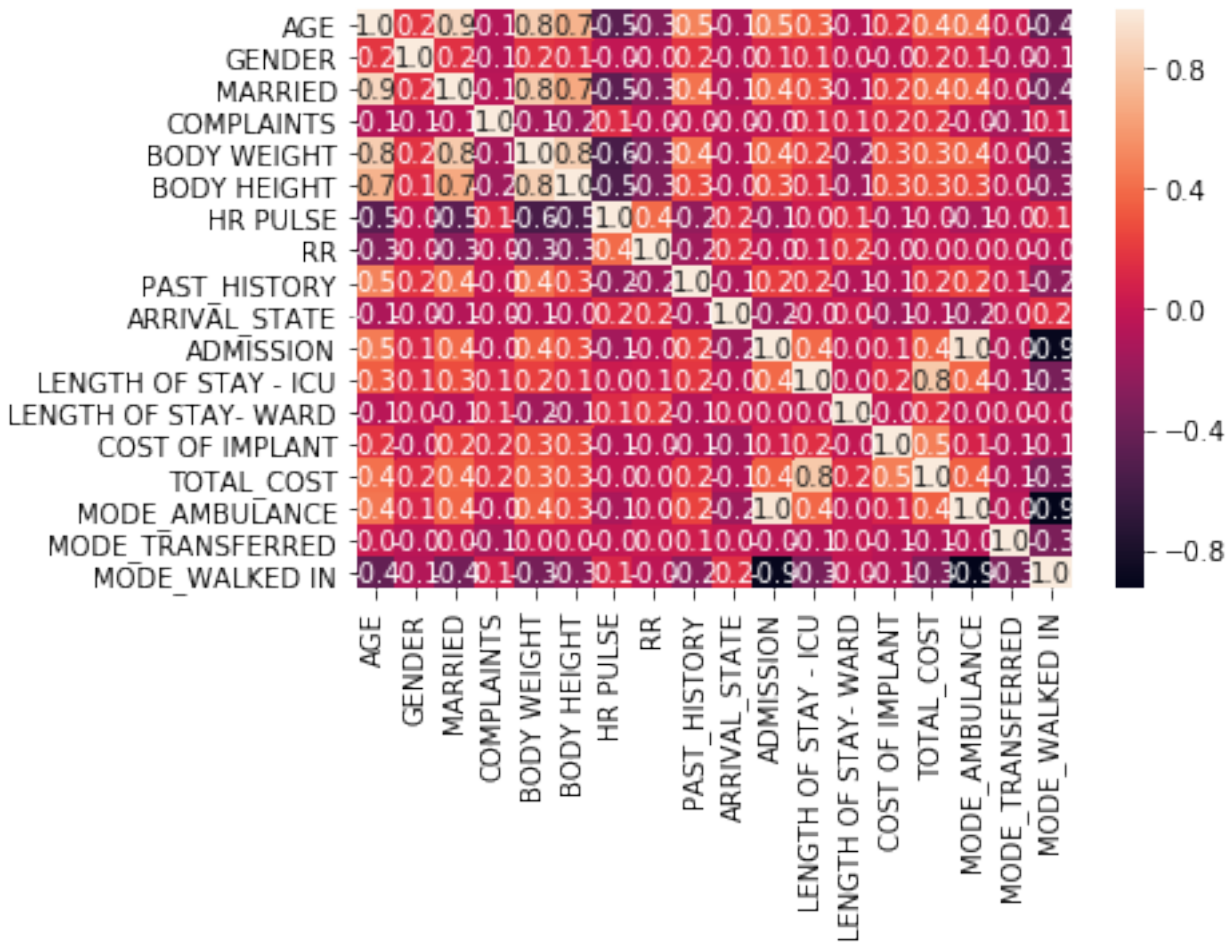
	AGE	GENDER	MARRIED	COMPLAINTS	BODY WEIGHT	BODY HEIGHT
AGE	1.000000	0.192781	0.898646	-0.075730	0.848246	0.717074
GENDER	0.192781	1.000000	0.167849	-0.070640	0.174573	0.140584
MARRIED	0.898646	0.167849	1.000000	-0.053620	0.831409	0.673006
COMPLAINTS	-0.075730	-0.070640	-0.053620	1.000000	-0.135944	-0.168374
BODY WEIGHT	0.848246	0.174573	0.831409	-0.135944	1.000000	0.849963
BODY HEIGHT	0.717074	0.140584	0.673006	-0.168374	0.849963	1.000000
HR PULSE	-0.483149	-0.017826	-0.475414	0.137960	-0.564886	-0.535004
RR	-0.283955	-0.012732	-0.253236	-0.028515	-0.316843	-0.300503
PAST_HISTORY	0.495505	0.182849	0.430943	-0.002814	0.422948	0.268000
ARRIVAL_STATE	-0.101247	-0.044720	-0.072444	-0.026220	-0.061983	-0.048469
ADMISSION	0.466735	0.117116	0.389710	-0.012116	0.352266	0.274429
LENGTH OF STAY - ICU	0.345533	0.144904	0.280433	0.122419	0.228660	0.137965
LENGTH OF STAY-WARD	-0.051795	0.012725	-0.053263	0.108463	-0.159878	-0.139425
COST OF IMPLANT	0.182193	-0.024966	0.227179	0.152789	0.277585	0.277072
TOTAL_COST	0.420852	0.152880	0.375130	0.180963	0.348270	0.293828
MODE_AMBULANCE	0.449089	0.103019	0.372483	-0.022647	0.353704	0.264552
MODE_TRANSFERRED	0.030275	-0.046092	0.016660	-0.128973	0.023726	0.016469
MODE_WALKED IN	-0.436858	-0.080788	-0.359243	0.068708	-0.344027	-0.256847

In [128]:

```
sns.heatmap(corr,annot=True,fmt=".1f")
```

Out[128]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b18ccc0>
```



Observation:

From the above correlation matrix it is observed that ADMISSION and MODE_ABULANCE are strongly correlated. Also body_height and body_weight are also strongly correlated. so we can drop one of the columns of the combination mentioned above which are strongly correlated.

In [129]:

```
hospital.drop(['ADMISSION'],axis=1,inplace=True)
```

In [130]:

```
hospital.drop(['BODY HEIGHT'],axis=1,inplace=True)
```

In [131]:

```
hospital.shape
```

Out[131]:

```
(248, 21)
```

In [132]:

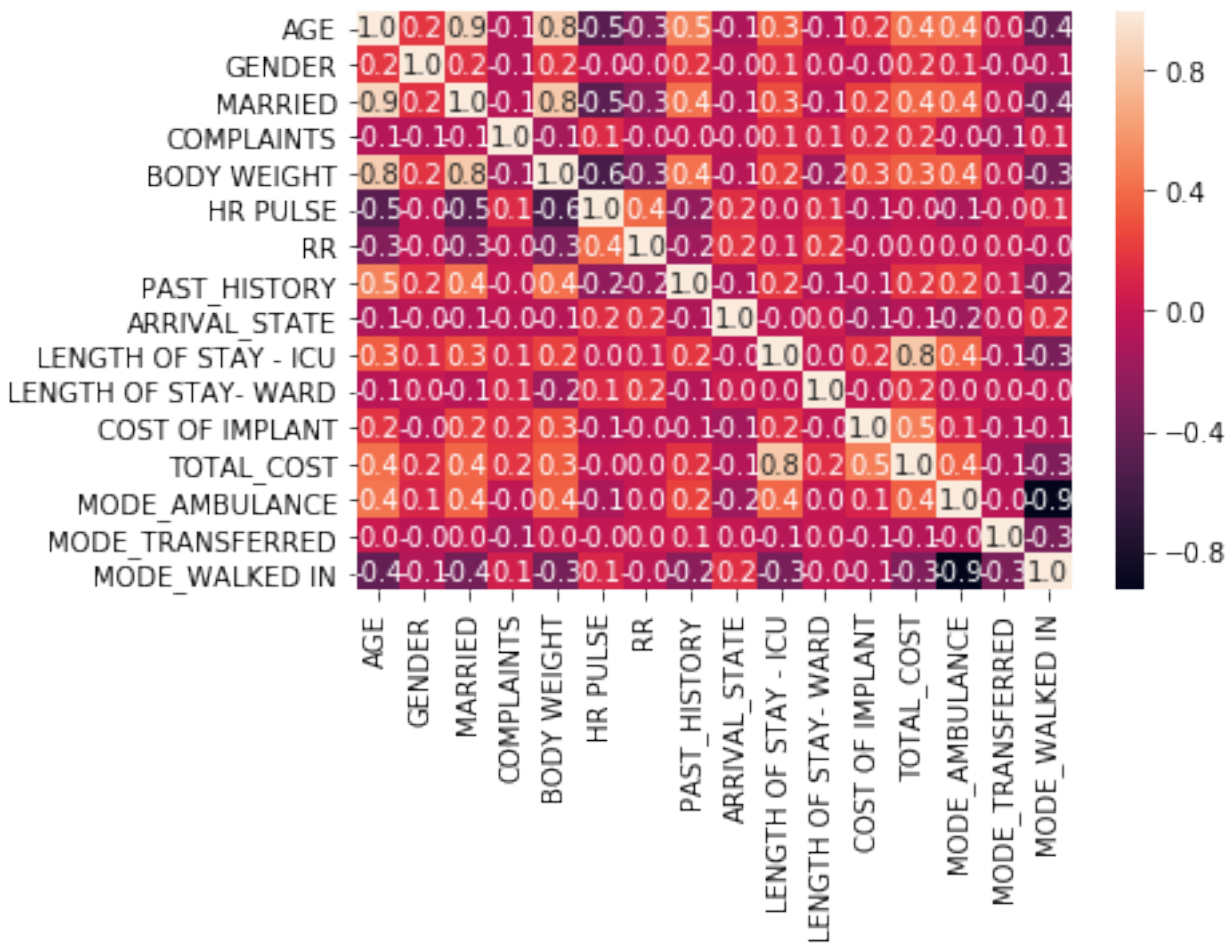
```
corr=hospital.corr()
```

In [133]:

```
sns.heatmap(corr,annot=True,fmt=".1f")
```

Out[133]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b609ef0>
```



Observation :

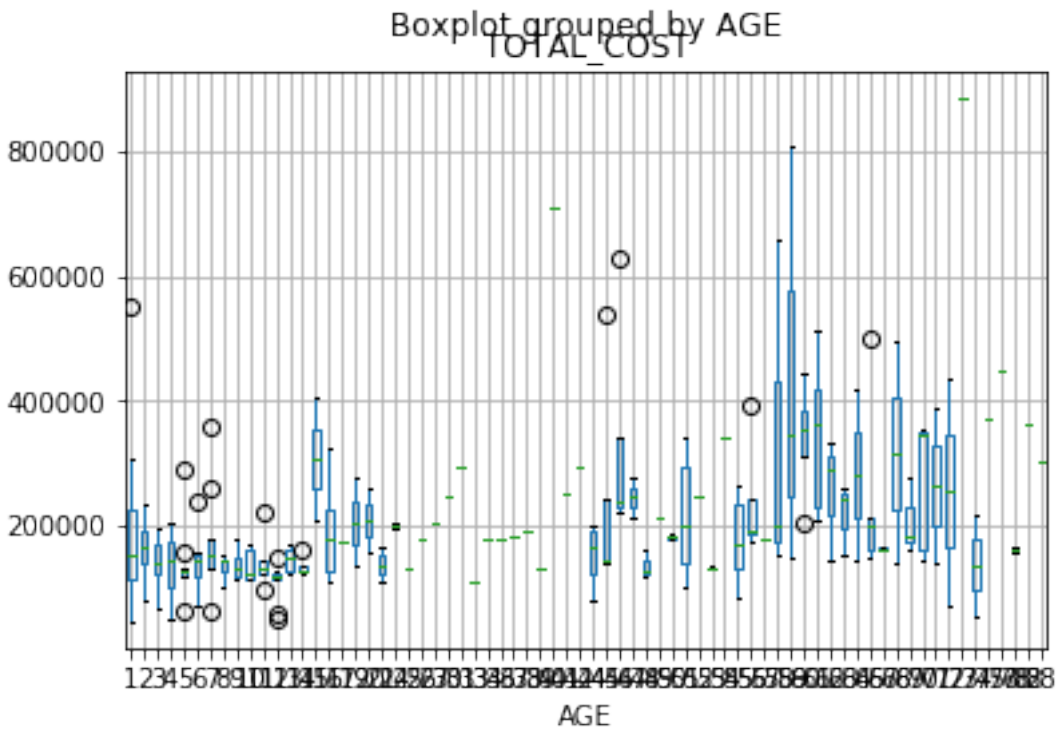
we have AGE and Married are correlated to 0.9 . we can drop MARRIED as AGE is much important for a treatment and is assocaited to cost From the above heatmap none of the independent variables have no multi-colinearity.

In [134]:

```
# box plot of rentals, grouped by season
hospital.boxplot(column='TOTAL_COST', by='AGE')
```

Out[134]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b8e7a58>
```



6. Train Test Split

In [135]:

```
# Isolating the output variable into a Target/Actual output
Y=hospital.TOTAL_COST
```

In [136]:

```
hospital.head(5)
```

Out[136]:

	AGE	GENDER	MARRIED	COMPLAINTS	BODY WEIGHT	HR PULSE	BP - HIGH	BP-LOW	RR	PAST_HISTC
0	58	1	1	1	49.2	118	100	80	32	
1	59	1	1	1	41.0	78	70	50	28	
2	82	1	1	1	46.6	100	110	80	20	
3	46	1	1	1	80.0	122	110	80	24	
4	60	1	1	1	58.0	72	180	100	18	

5 rows × 21 columns

In [137]:

```
X=hospital
```

In [138]:

```
X_train, X_test, y_train, y_test = train_test_split(X,Y, random_state=123)
```

In [139]:

```
X_train.shape
```

Out[139]:

```
(186, 21)
```

In [140]:

```
X_test.head(5)
```

Out[140]:

	AGE	GENDER	MARRIED	COMPLAINTS	BODY WEIGHT	HR PULSE	BP - HIGH	BP- LOW	RR	PAST_HIS
204	12	0	0	1	26.0	58	90	60	26	
91	3	1	0	1	13.0	140	90	60	24	
145	1	0	0	1	6.8	112	80	50	24	
52	71	1	1	1	56.0	72	130	70	20	
19	47	1	1	1	59.0	80	110	80	20	

5 rows × 21 columns

7. Creating Linear Regression Model

In [141]:

```
    linreg = LinearRegression()
    linreg.fit(X_train, y_train)
```

Out[141]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

In [142]:

```
y_pred = linreg.predict(X_test)
```

8. Model Performance Metrics

In [145]:

```
from sklearn import metrics
import numpy as np
np.sqrt(metrics.mean_squared_error(y_test, y_pred))

print ('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print ('MSE:', metrics.mean_squared_error(y_test, y_pred))
print ('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 6.196299387562659e-11
MSE: 5.9411847247937716e-21
RMSE: 7.707908098046948e-11
```

Out[145]:

```
<function sklearn.metrics.regression.r2_score>
```

In [ ]:

```

```