ASSIGNMENT ANALYSIS OF HEALTH CAREDATA
"COST OF CARE"

Submitted By-

Name: Prachi Tripathi

GitHub: https://github.com/prachi-tripathi/HolMusk-Health-care-data-analysis/

College: Aegis school of business Data Science, Cyber Security and Telecommunication

TASKS

Step1: Reading data.

Step2: Merging all the data frames data.

Step3: Data understanding.

Step4: Data pre-processing.

Step5: Data Analysis using charts and graphs.

Step6: Extracting Insights of the Data.

TOOLS, TECHNIQUES AND APPORACH

1. Tools

- Jupyter Notebook.
- Python

2. Libraries

- Pandas- Applying data frame operations.
- Matplotlib- Visualization of bar and charts.
- Seaborn- Visualization of bar and charts.
- Numpy- deal with array operations.
- DateTime- deal with date time operations.

Importing required libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
from datetime import date
from datetime import datetime
import matplotlib.pyplot as plt
```

3. Approach

- Reading data frame using Pandas Library.

Reading dataframes

```
bill_amout_df= pd.read_csv("C:\\Users\\Prachi\\HolMusk\\bill_amount.csv")
bill_id_df = pd.read_csv("C:\\Users\\Prachi\\HolMusk\\bill_id.csv")
clinical_data_df = pd.read_csv("C:\\Users\\Prachi\\HolMusk\\clinical_data.csv")
demographics_df = pd.read_csv("C:\\Users\\Prachi\\HolMusk\\demographics.csv")
```

- Renaming the column name of the data frame and visualizing head records the data frame.

```
clinical_data_df.rename({'id' : 'patient_id'}, axis=1, inplace=True)
clinical_data_df.head()
```

- Checking for Count of null values.

```
print('Count of NUll values in bill_id_df:\n',bill_id_df.isnull().sum())
print('================================================')
print('Count of NUll values in bll_amout_df:\n',bill_amout_df.isnull().sum())
print('================================================')
print('Count of NUll values in clinical_data_df:\n',clinical_data_df.isnull().sum())
print('================================================')
print('Count of NUll values in demographics_df:\n',demographics_df.isnull().sum())
```

- Checking for the Unique values of all the features of the clinical_data and demographic data frame.

```
print(clinical_data_df['medical_history_2'].unique())
print(clinical_data_df['medical_history_5'].unique())
print(demographics_df['gender'].unique())
print(demographics_df['race'].unique())
print(demographics_df['resident_status'].unique())
print(clinical_data_df['symptom_1'].unique())
print(clinical_data_df['symptom_2'].unique())
print(clinical_data_df['symptom_3'].unique())
print(clinical_data_df['symptom_4'].unique())
print(clinical_data_df['symptom_5'].unique())

print(clinical_data_df['medical_history_1'].unique())
print(clinical_data_df['medical_history_2'].unique())
print(clinical_data_df['medical_history_3'].unique())
print(clinical_data_df['medical_history_4'].unique())
print(clinical_data_df['medical_history_5'].unique())
print(clinical_data_df['medical_history_6'].unique())
print(clinical_data_df['medical_history_7'].unique())

print(clinical_data_df['preop_medication_1'].unique())
print(clinical_data_df['preop_medication_2'].unique())
print(clinical_data_df['preop_medication_3'].unique())
print(clinical_data_df['preop_medication_4'].unique())
print(clinical_data_df['preop_medication_5'].unique())
print(clinical_data_df['preop_medication_6'].unique())
```

- For better understanding replacing with the appropriate values by considering below-
  - ✓ For medical history, if medical history= 1 replacing with Yes, if 0 than No
  - ✓ For Symptoms, if medical history= 1 replacing with Yes, if 0 than No
  - ✓ For Pre medical, if medical history= 1 replacing with Yes, if 0 than No
  - ✓ For Race, India and Indian citizen is conflicting replacing with Indian
  - ✓ For Resident status Singaporean and Singaporean citizen is conflicting
  - ✓ replacing with Singaporean.
  - ✓ For Gender there is ambiguity in the value so, replacing with only either Male or Female.

```python
demographics_df['gender'] = demographics_df['gender'].replace('f','Female')
demographics_df['gender'] = demographics_df['gender'].replace('m','Male')
demographics_df['race'] = demographics_df['race'].replace('India','Indian')
demographics_df['race'] = demographics_df['race'].replace('chinese','Chinese')
demographics_df['resident_status'] = demographics_df['resident_status'].replace('Singapore citizen','Singaporean')

clinical_data_df['medical_history_2'] = clinical_data_df['medical_history_2'].replace(np.nan,0)
clinical_data_df['medical_history_5'] = clinical_data_df['medical_history_5'].replace(np.nan,0)
clinical_data_df['medical_history_2'] = clinical_data_df['medical_history_2'].replace(np.nan,0)
clinical_data_df['medical_history_5'] = clinical_data_df['medical_history_5'].replace(np.nan,0)
clinical_data_df['symptom_1'] = clinical_data_df['symptom_1'].replace(1,"Yes")
clinical_data_df['symptom_1'] = clinical_data_df['symptom_1'].replace(0,"No")
clinical_data_df['symptom_2'] = clinical_data_df['symptom_2'].replace(1,"Yes")
clinical_data_df['symptom_2'] = clinical_data_df['symptom_2'].replace(0,"No")
clinical_data_df['symptom_3'] = clinical_data_df['symptom_3'].replace(1,"Yes")
clinical_data_df['symptom_3'] = clinical_data_df['symptom_3'].replace(0,"No")
clinical_data_df['symptom_4'] = clinical_data_df['symptom_4'].replace(1,"Yes")
clinical_data_df['symptom_4'] = clinical_data_df['symptom_4'].replace(0,"No")
clinical_data_df['symptom_5'] = clinical_data_df['symptom_5'].replace(1,"Yes")
clinical_data_df['symptom_5'] = clinical_data_df['symptom_5'].replace(0,"No")
clinical_data_df['medical_history_1'] = clinical_data_df['medical_history_1'].replace(1,"Yes")
clinical_data_df['medical_history_1'] = clinical_data_df['medical_history_1'].replace(0,"No")
clinical_data_df['medical_history_2'] = clinical_data_df['medical_history_2'].replace(1.,"Yes")
clinical_data_df['medical_history_2'] = clinical_data_df['medical_history_2'].replace(0.,"No")
clinical_data_df['medical_history_3'] = clinical_data_df['medical_history_3'].replace('1',"Yes")
clinical_data_df['medical_history_3'] = clinical_data_df['medical_history_3'].replace('0',"No")
clinical_data_df['medical_history_4'] = clinical_data_df['medical_history_4'].replace(1,"Yes")
clinical_data_df['medical_history_4'] = clinical_data_df['medical_history_4'].replace(0,"No")
clinical_data_df['medical_history_5'] = clinical_data_df['medical_history_5'].replace(1.,"Yes")
clinical_data_df['medical_history_5'] = clinical_data_df['medical_history_5'].replace(0.,"No")
clinical_data_df['medical_history_6'] = clinical_data_df['medical_history_6'].replace(1,"Yes")
clinical_data_df['medical_history_6'] = clinical_data_df['medical_history_6'].replace(0,"No")
clinical_data_df['medical_history_7'] = clinical_data_df['medical_history_7'].replace(1,"Yes")
clinical_data_df['medical_history_7'] = clinical_data_df['medical_history_7'].replace(0,"No")

clinical_data_df['preop_medication_1'] = clinical_data_df['preop_medication_1'].replace(1,"Yes")
clinical_data_df['preop_medication_1'] = clinical_data_df['preop_medication_1'].replace(0,"No")
clinical_data_df['preop_medication_2'] = clinical_data_df['preop_medication_2'].replace(1,"Yes")
clinical_data_df['preop_medication_2'] = clinical_data_df['preop_medication_2'].replace(0,"No")
clinical_data_df['preop_medication_3'] = clinical_data_df['preop_medication_3'].replace(1,"Yes")
clinical_data_df['preop_medication_3'] = clinical_data_df['preop_medication_3'].replace(0,"No")
clinical_data_df['preop_medication_4'] = clinical_data_df['preop_medication_4'].replace(1,"Yes")
clinical_data_df['preop_medication_4'] = clinical_data_df['preop_medication_4'].replace(0,"No")
clinical_data_df['preop_medication_5'] = clinical_data_df['preop_medication_5'].replace(1,"Yes")
clinical_data_df['preop_medication_5'] = clinical_data_df['preop_medication_5'].replace(0,"No")
clinical_data_df['preop_medication_6'] = clinical_data_df['preop_medication_6'].replace(1,"Yes")
clinical_data_df['preop_medication_6'] = clinical_data_df['preop_medication_6'].replace(0,"No")
```

- Analyzing the shape of data frames.

```
print('shape of bill_id_df :',bill_id_df.shape)
print('shape of bll_amout_df :',bill_amout_df.shape)
print('shape of clinical_data_df :',clinical_data_df.shape)
print('shape of demographics_df :',demographics_df.shape)
```

- Analyzing the duplicates records in the data frames which does not add any value.

```
print('Count of duplicates values in bill_id_df :',bill_id_df.duplicated().sum())
print('Count of duplicates values in bill_amout_df :',bill_amout_df.duplicated().sum())
print('Count of duplicates values in clinical_data_df :',clinical_data_df.duplicated().sum())
print('Count of duplicates values in demographics_df :',demographics_df.duplicated().sum())
```

- Merging the bill_amout_df and bill_id_df using bill id and create 1 table
- Merging the clinical_data_df and demographics_df using patient_id id and create 2 table
- Out of 4 data frame we have now 2 tables.
- Again, merging all 2 tables into single main data frame for further analysis using patient_id

```
mereged_bill_details_df = pd.merge(bill_amout_df,bill_id_df, on='bill_id')
mereged_patient_details_df = pd.merge(clinical_data_df,demographics_df, on='patient_id')
main_df= pd.merge(mereged_patient_details_df,mereged_bill_details_df, on='patient_id')
```

- Type casting features to the required data type.

```
main_df['weight'] = main_df['weight'].map('{:,.1f}'.format)
main_df['height'] = main_df['height'].map('{:,.1f}'.format)
main_df['weight'] = main_df.weight.astype(float)
main_df['height'] = main_df.height.astype(float)
main_df['number_of_days_admitted'] = main_df['number_of_days_admitted']

```

- Using date of birth calculating the Age of the patient and adding age column to the data frame.

```
def calculate_age(born):
        born = datetime.strptime(born, "%Y-%m-%d").date()
        today = date.today()
        return today.year - born.year - ((today.month, today.day) < (born.month, born.day))

main_df['age'] = main_df['date_of_birth'].apply(calculate_age)
```

- Using Admission date and Discharge date calculating the patient admitted number of days in the hospital and adding admitted days column to the data frame.

```
admited_date = main_df['date_of_admission_x']
discharged_date = main_df['date_of_discharge']

def calculate_No_Of_Admitted_Days(admited_date, discharged_date):
        admited_date = pd.to_datetime(admited_date, format= "%Y-%m-%d")
        discharged_date = pd.to_datetime(discharged_date, format= "%Y-%m-%d")
        number_of_days_admitted = (discharged_date- admited_date).astype('timedelta64[D]')
        main_df['number_of_days_admitted'] = number_of_days_admitted
        return main_df

calculate_No_Of_Admitted_Days(admited_date, discharged_date)
```

```
main_df['weight'] = main_df['weight'].map('{:,.1f}'.format)
main_df['height'] = main_df['height'].map('{:,.1f}'.format)
main_df['weight'] = main_df.weight.astype(float)
main_df['height'] = main_df.height.astype(float)
main_df['number_of_days_admitted'] = main_df['number_of_days_admitted']

```

- Fetching the subset symptoms, pre-medication, and medical history from main data frame. To see the analysis how these feature affects the cost of care expenses of the hospital.

```
Symptons_df = main_df[['symptom_1','symptom_2','symptom_3','symptom_4','symptom_5','amount']]
pre_Medication_df = main_df[['preop_medication_1','preop_medication_2','preop_medication_3','preop_medication_4',
Medical_history_df = main_df[['medical_history_1','medical_history_2','medical_history_3','medical_history_4','me
```

- Grouping the above subset data frame and considering only those records where having medical Conditions is 'Yes'.
- Merging above all grouped data frame of the symptom, pre-medication and medical-history.

```
grouped_sym1 = Symptons_df.groupby(['symptom_1']).sum().reset_index()
grouped_sym1 = grouped_sym1.loc[grouped_sym1['symptom_1'] == 'Yes']

grouped_sym2 = Symptons_df.groupby(['symptom_2']).sum().reset_index()
grouped_sym2 = grouped_sym2.loc[grouped_sym2['symptom_2'] == 'Yes']

grouped_sym3 = Symptons_df.groupby(['symptom_3']).sum().reset_index()
grouped_sym3 = grouped_sym3.loc[grouped_sym3['symptom_3'] == 'Yes']

grouped_sym4 = Symptons_df.groupby(['symptom_4']).sum().reset_index()
grouped_sym4 = grouped_sym4.loc[grouped_sym4['symptom_4'] == 'Yes']

grouped_sym5 = Symptons_df.groupby(['symptom_5']).sum().reset_index()
grouped_sym5 = grouped_sym5.loc[grouped_sym5['symptom_5'] == 'Yes']


#pre medication dataframe with respect to amount.

pre_Medication_df1 = pre_Medication_df.groupby(['preop_medication_1']).sum().reset_index()
pre_Medication_df1 = pre_Medication_df1.loc[pre_Medication_df1['preop_medication_1'] == 'Yes']

pre_Medication_df2 = pre_Medication_df.groupby(['preop_medication_2']).sum().reset_index()
pre_Medication_df2 = pre_Medication_df2.loc[pre_Medication_df2['preop_medication_2'] == 'Yes']

pre_Medication_df3 = pre_Medication_df.groupby(['preop_medication_3']).sum().reset_index()
pre_Medication_df3 = pre_Medication_df3.loc[pre_Medication_df3['preop_medication_3'] == 'Yes']
```

- Below steps are involving pre-processing of the above grouped data frame
  - ✓ Transposing of the data frame.
  - ✓ Reset-indexing of the data frame.
  - ✓ Data type casting of the data frame features.
  - ✓ Dropping not required feature of the data frame.
  - ✓ Renaming the required feature of the data frame.

```
sympton_newCopy = sympton_concated_df.drop('amount', axis=1).transpose().copy()
pre_medi_newCopy = PreMedi_concated_df.drop('amount', axis=1).transpose().copy()
medi_history_newCopy = MediHistory_concated_df.drop('amount', axis=1).transpose().copy()
```

```
sympton_newCopy['amount'] = list(sympton_concated_df['amount'])
pre_medi_newCopy['amount'] = list(PreMedi_concated_df['amount'])
medi_history_newCopy['amount'] = list(MediHistory_concated_df['amount'])
```

```
sympton_newCopy.drop(1, axis=1, inplace=True)
pre_medi_newCopy.drop(1, axis=1, inplace=True)
medi_history_newCopy.drop(1, axis=1, inplace=True)
```

```
sympton_newCopy['amount'] = sympton_newCopy['amount'].astype(float)
pre_medi_newCopy['amount'] = pre_medi_newCopy['amount'].astype(float)
medi_history_newCopy['amount'] = medi_history_newCopy['amount'].astype(float)
```

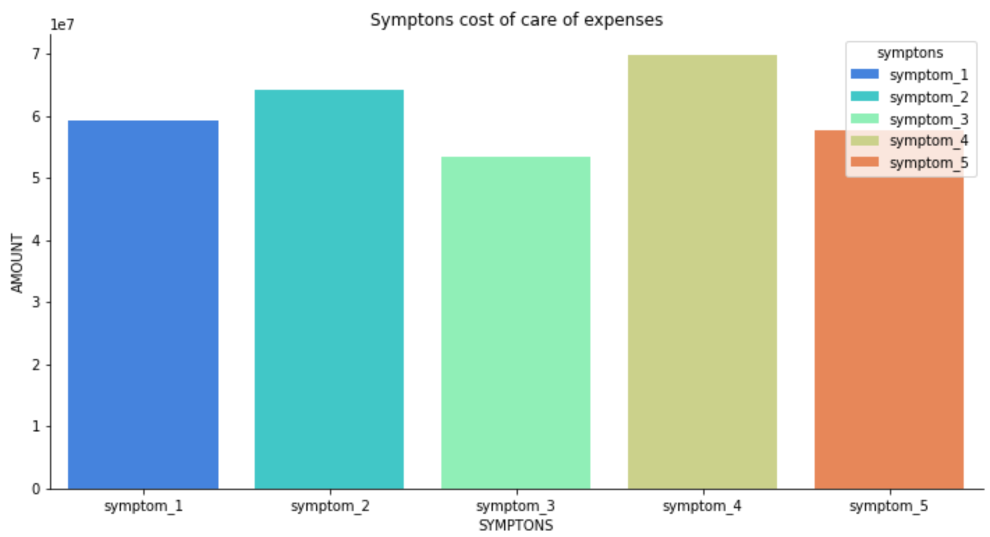
```
sympton_newCopy  = sympton_newCopy.reset_index()
sympton_newCopy.rename({'index' : 'symptons'}, axis=1, inplace=True)

pre_medi_newCopy  = pre_medi_newCopy.reset_index()
pre_medi_newCopy.rename({'index' : 'pre_medication'}, axis=1, inplace=True)

medi_history_newCopy  = medi_history_newCopy.reset_index()
medi_history_newCopy.rename({'index' : 'MediCated_History'}, axis=1, inplace=True)
```

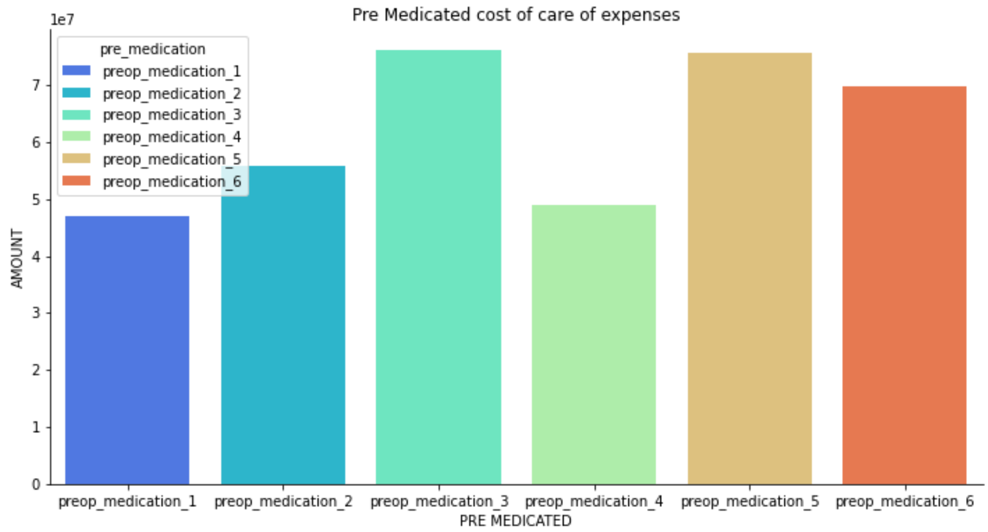
ANALYSIS CONCLUSION

Bar graph, analyzing the relationship between the symptoms & the amount



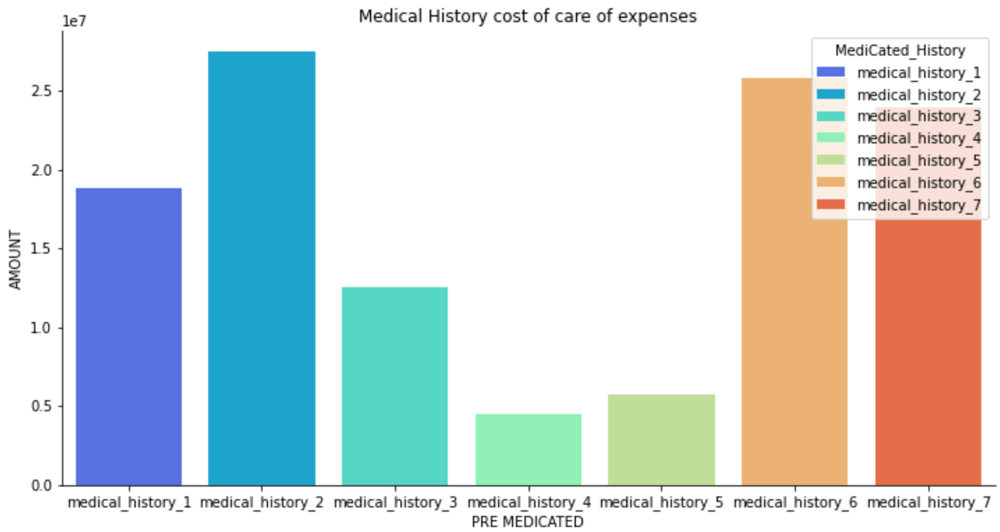
- Patient having symptoms 4 causes most increasing hospitalization cost of care of expenses.

Bar graph, analyzing the relationship between the Pre-Medication & the cost of care of expenses



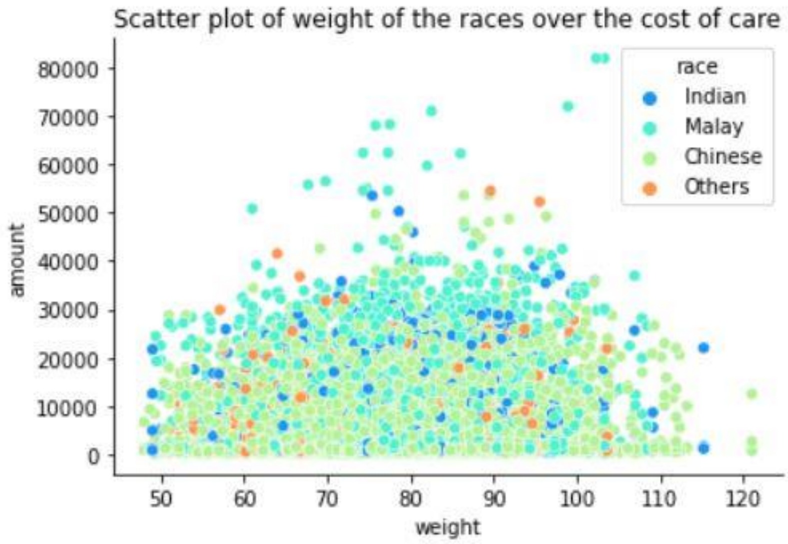
- Patient having Pre-Medicated 3 causes most increasing hospitalization cos of care of expenses.

Bar graph, analyzing the relationship between the Medical history & the cost of care of expenses



- Patient having Medical history 2 causes most increasing hospitalization cost of care of expenses.

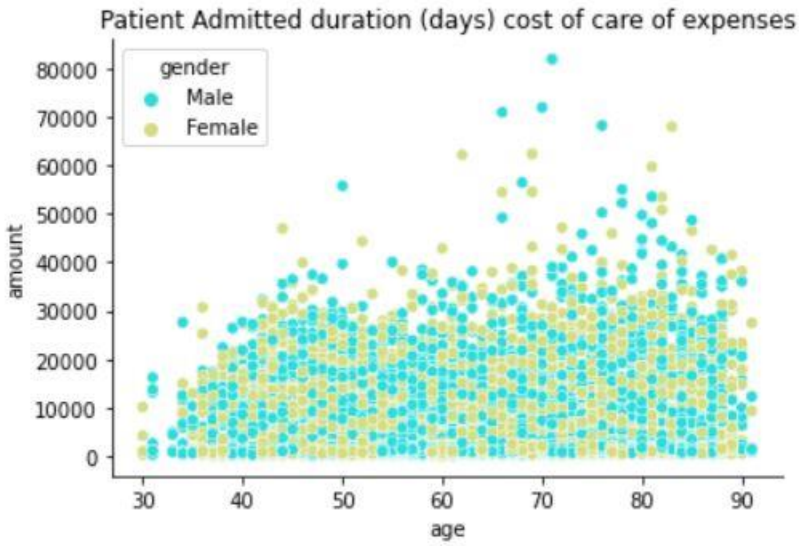
Scatter graph, analyzing the relationship between the weight of the gender over the amount



- As weight of the patient is increasing the cost of care is also increasing.
- As compared other races majorly hospital is captured by Chinese patients.

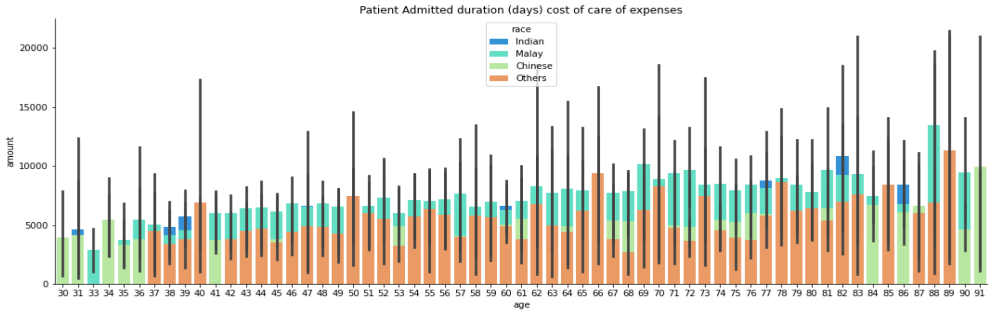
So, most of the cost of care getting by Chinese patients. Age is directly proportionate cost of care.

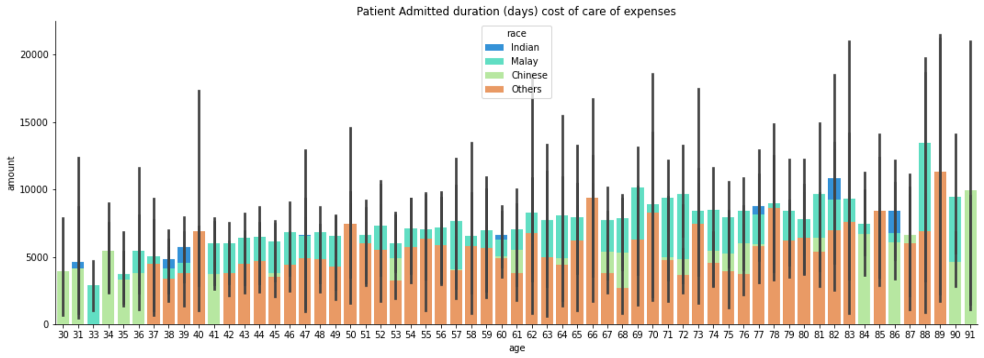
Scatter graph, analyzing the relationship between the age of the gender and amount



- Number of days patient is admitted in the hospital is directly proportionate cost of care.

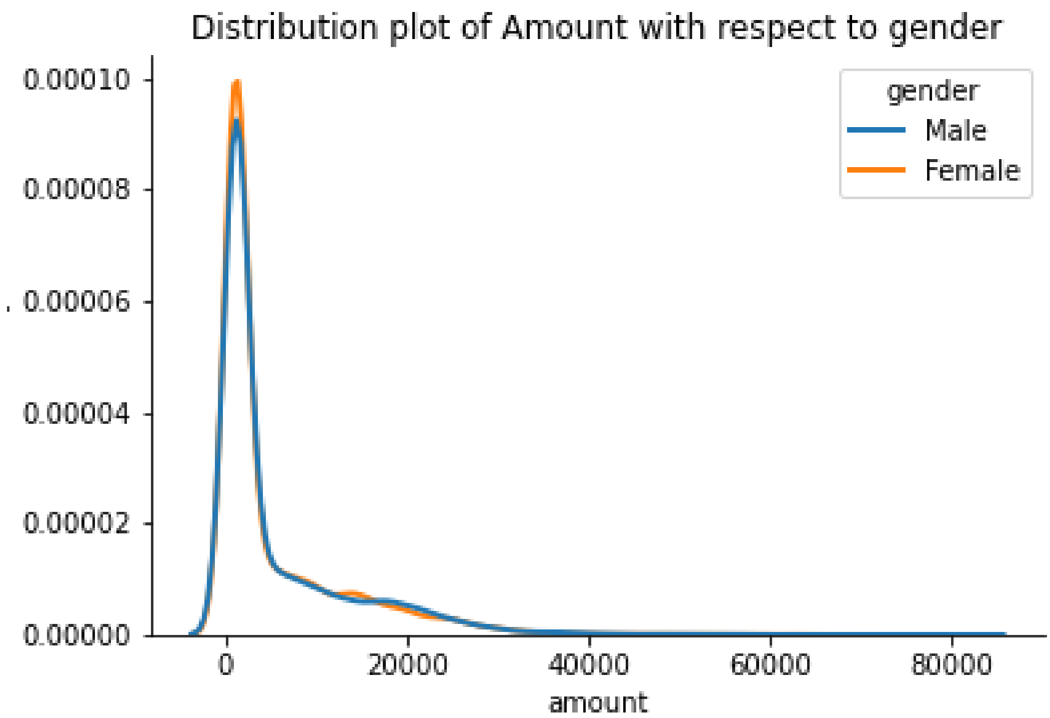
Bar graph, analyzing the relationship between the age of the gender and amount





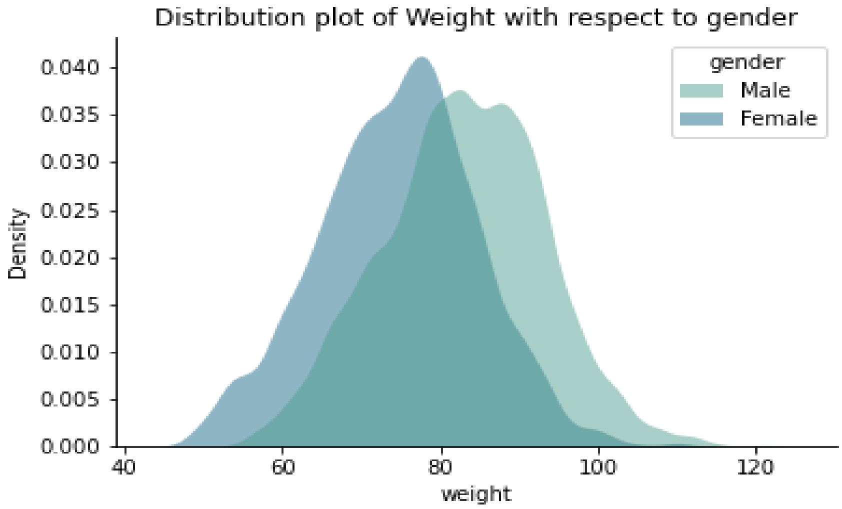
- As, we can conclude Age group 31-33 are only Male patients.
- As, we can conclude Age group 38-40 are only Female patients.
- As, we can conclude Age group 61-62 are only Female patients.
- As, we can conclude Age group 64-65 are only Female patients.
- Age group below 30 - 30 are Female patients.
- As, we can conclude Age group 67,72,75,89,90,91 is only Female patients.
- Hence, we can say that most of the cost of care amount is sweeping by female patients of above 40.

Distribution graph analyzing the distribution of amount among gender



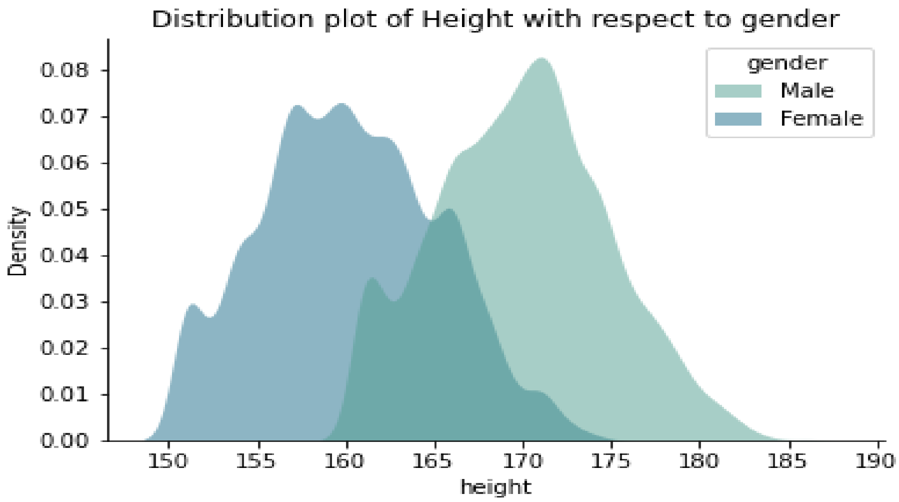
- Amount is positively skewed.

Distribution graph analyzing the distribution of weight among gender



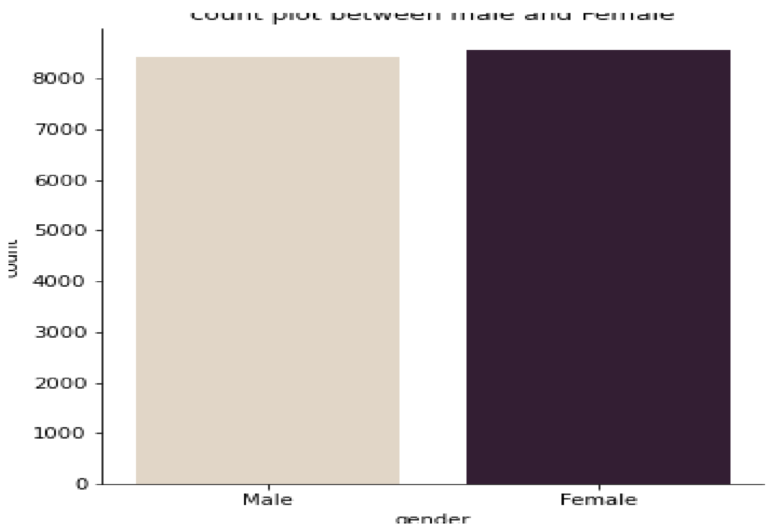
- Weight male and female is Normally Distributed

Distribution graph analyzing the distribution of Height among gender



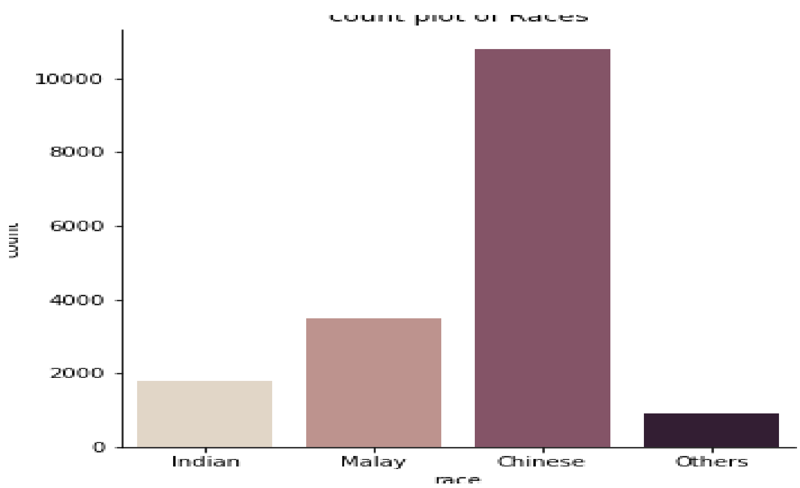
- Height male and female is Normally Distributed

Count Plot between genders



- As compared other races majorly hospital is captured by Chinese patients.

Count graph of different categories of Races

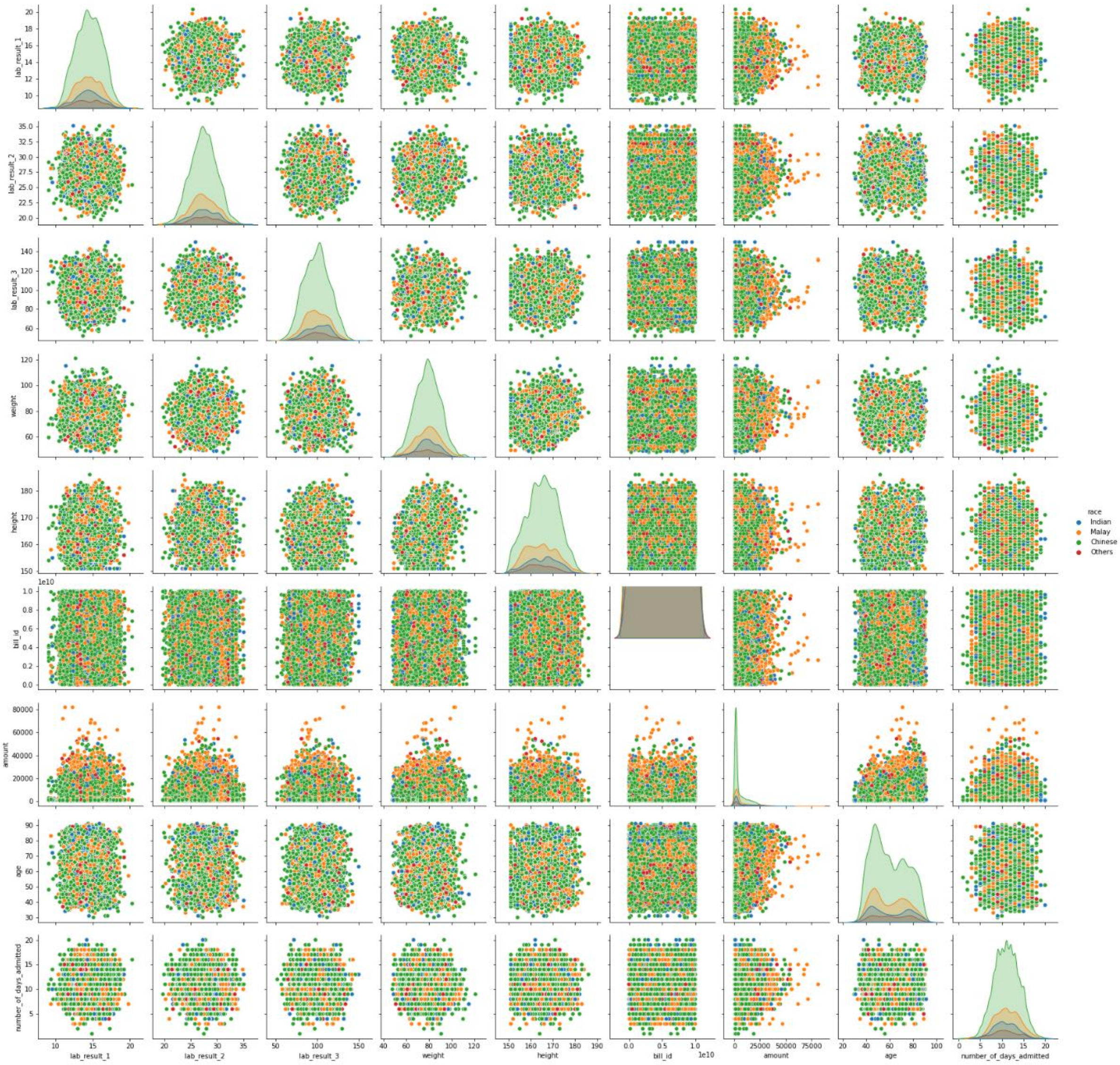


- As compared other races majorly hospital is captured by Chinese patients.

Analysing of the Correlation between all features

- Weight, Admission days, age are correlated with the amount.

Analysing all features Using Pair plot



CONCLUSION

- Patient having symptom 4 causes most increasing hospitalization cost of care of expenses.
- Patient having Pre-Medicated 3 causes most increasing hospitalization cost of care of expenses.
- Patient having Medical history 2 causes most increasing hospitalization cost of care of expenses.
- As weight of the patient is increasing the cost of care is also increasing.
- As compared other races majorly hospital is captured by Chinese patients.
- So, most of the cost of care getting by Chinese patients.
- Age is directly proportionate cost of care.
- Number of days patient is admitted in the hospital is directly proportionate cost of care.
- The most of the cost of care amount is sweeping by female patients of above 40.
- As patient's admission days increases the cost of care is also increases.