Medical Insurance Cost Prediction

The objective of this case study is to predict the health insurance cost incurred by Individuals based on their customer_age, gender, BMI, number of dependents, smoking habit and geo-location.

Features available are:

- gender: insurance contractor gender, female, male
- body_mass_index: Body mass index (ideally 18.5 to 24.9)
- dependents: Number of dependents covered by health insurance / Number of dependents
- smoking_status: smoking habits
- location: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.
- insurance_cost: Individual medical costs billed by health insurance

Importing Essentials

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.ensemble import RandomForestRegressor
```

In [2]:
```
df = pd.read_csv('insurance.csv')
```

Exploratory Data Analysis

In [3]:
```
df.head()
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [4]:
```
df.shape
```
Out[4]:
```
(1338, 7)
```

In [5]:
```
df['location'].unique()
```
Out[5]:
```
array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

In [6]:
```
location_dummies = pd.get_dummies(df['location'], drop_first=True)
location_dummies.head()
```
Out[6]:

	northwest	southeast	southwest
0	0	0	1
1	0	1	0
2	0	1	0
3	1	0	0
4	1	0	0

In [7]:
```
df = pd.concat([df, location_dummies], axis=1)
```

In [8]:
```
df.drop(['location'],axis=1, inplace=True)
```

In [9]:
```
df.head()
```
Out[9]:

	age	sex	bmi	children	smoker	charges	northwest	southeast	southwest
0	19	female	27.900	0	yes	16884.92400	0	0	1
1	18	male	33.770	1	no	1725.55230	0	1	0
2	28	male	33.000	3	no	4449.46200	0	1	0
3	33	male	22.705	0	no	21984.47061	1	0	0
4	32	male	28.880	0	no	3866.85520	1	0	0

In [10]:
```
df.isnull().sum()
```
Out[10]:
```
age          0
sex          0
bmi          0
children     0
smoker       0
charges      0
northwest    0
southeast    0
southwest    0
dtype: int64
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   age        1338 non-null   int64  
 1   sex        1338 non-null   object 
 2   bmi        1338 non-null   float64
 3   children   1338 non-null   int64  
 4   smoker     1338 non-null   object 
 5   charges    1338 non-null   float64
 6   northwest  1338 non-null   uint8  
 7   southeast  1338 non-null   uint8  
 8   southwest  1338 non-null   uint8  
dtypes: float64(2), int64(2), object(2), uint8(3)
memory usage: 66.8+ KB
```

In [12]:
```
df_customer_age = df.groupby(by='customer_age').mean()
df_customer_age
```

Out[12]:

age	bmi	children	charges	northwest	southeast	southwest
18	31.326159	0.449275	7086.217556	0.000000	0.536232	0.000000
19	28.596912	0.426471	9747.909335	0.500000	0.044118	0.455882
20	30.632759	0.862069	10159.697736	0.241379	0.275862	0.275862
21	28.185714	0.785714	4730.464330	0.250000	0.250000	0.250000
22	31.087679	0.714286	10012.932802	0.250000	0.285714	0.214286
23	31.454464	1.000000	12419.820040	0.250000	0.250000	0.250000
24	29.142679	0.464286	10648.015962	0.250000	0.250000	0.250000
25	29.693929	1.285714	9838.365311	0.250000	0.250000	0.250000
26	29.428929	1.071429	6133.825309	0.250000	0.250000	0.250000
27	29.333571	0.964286	12184.701721	0.214286	0.321429	0.214286
28	29.482143	1.285714	9069.187564	0.214286	0.285714	0.250000
29	29.383148	1.259259	10430.158727	0.259259	0.259259	0.222222
30	30.557593	1.555556	12719.110358	0.222222	0.296296	0.259259
31	29.918333	1.407407	10196.980573	0.259259	0.259259	0.222222
32	31.597692	1.269231	9220.300291	0.269231	0.307692	0.230769
33	31.163077	1.538462	12351.532987	0.230769	0.307692	0.269231
34	30.274038	1.153846	11613.528121	0.230769	0.230769	0.269231
35	31.394800	1.680000	11307.182031	0.240000	0.240000	0.280000
36	29.374200	1.240000	12204.476138	0.240000	0.280000	0.200000
37	31.216600	1.520000	18019.911877	0.280000	0.240000	0.240000
38	28.996600	1.480000	8102.733674	0.240000	0.280000	0.240000
39	29.910200	2.200000	11778.242945	0.240000	0.240000	0.280000
40	30.139074	1.592593	11772.251310	0.259259	0.296296	0.185185
41	31.506852	1.407407	9653.745650	0.259259	0.296296	0.222222
42	30.328148	1.000000	13061.038669	0.222222	0.296296	0.259259
43	30.204444	1.629630	19267.278653	0.185185	0.296296	0.259259
44	30.844259	1.222222	15859.396587	0.259259	0.296296	0.222222
45	29.778966	1.482759	14830.199856	0.241379	0.241379	0.275862
46	31.340862	1.620690	14342.590639	0.241379	0.241379	0.241379
47	30.664310	1.379310	17653.999593	0.206897	0.310345	0.241379
48	31.925690	1.310345	14632.500445	0.241379	0.310345	0.206897
49	30.313929	1.500000	12696.006264	0.250000	0.250000	0.250000
50	31.132241	1.310345	15663.003301	0.241379	0.241379	0.275862
51	31.727069	1.103448	15682.255867	0.206897	0.310345	0.241379
52	32.936034	1.482759	18256.269719	0.275862	0.241379	0.241379

	bmi	children	charges	northwest	southeast	southwest
age						
53	30.360893	1.250000	16020.930755	0.250000	0.250000	0.250000
54	31.234286	1.428571	18758.546475	0.250000	0.250000	0.250000
55	31.950000	0.961538	16164.545488	0.230769	0.269231	0.269231
56	31.600962	0.769231	15025.515837	0.230769	0.230769	0.269231
57	30.844423	0.615385	16447.185250	0.269231	0.230769	0.230769
58	32.718200	0.240000	13878.928112	0.280000	0.240000	0.240000
59	30.572000	1.200000	18895.869532	0.200000	0.320000	0.240000
60	30.332826	0.347826	21979.418507	0.217391	0.260870	0.260870
61	32.548261	0.739130	22024.457609	0.260870	0.217391	0.260870
62	32.342609	0.565217	19163.856573	0.260870	0.260870	0.217391
63	31.923478	0.565217	19884.998461	0.260870	0.260870	0.260870
64	32.976136	0.772727	23275.530837	0.227273	0.363636	0.227273

In [13]:
```
df.describe()
```

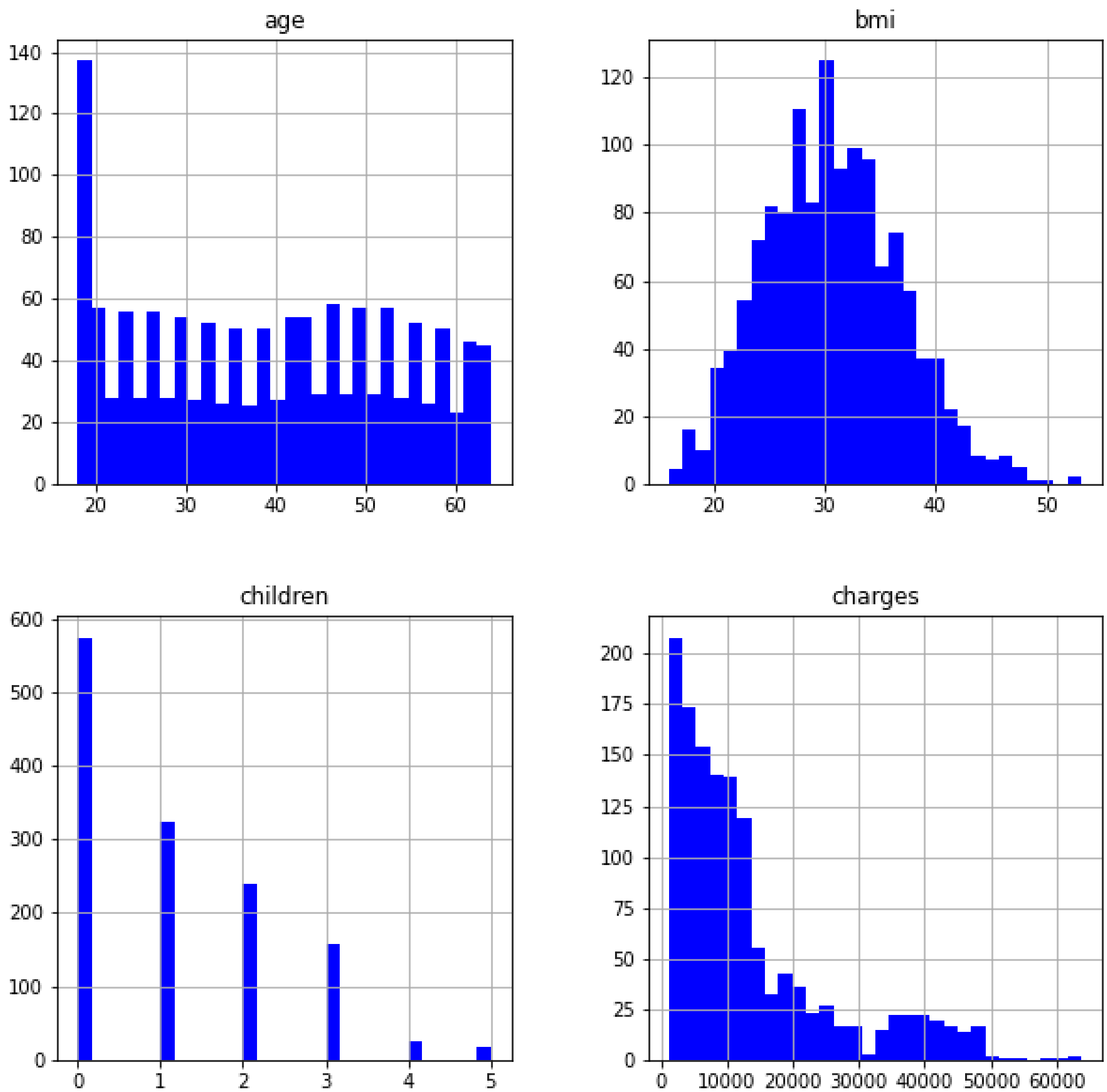
Out[13]:

	age	bmi	children	charges	northwest	southeast	southw
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000(
mean	39.207025	30.663397	1.094918	13270.422265	0.242900	0.272048	0.242(
std	14.049960	6.098187	1.205493	12110.011237	0.428995	0.445181	0.428(
min	18.000000	15.960000	0.000000	1121.873900	0.000000	0.000000	0.000(
25%	27.000000	26.296250	0.000000	4740.287150	0.000000	0.000000	0.000(
50%	39.000000	30.400000	1.000000	9382.033000	0.000000	0.000000	0.000(
75%	51.000000	34.693750	2.000000	16639.912515	0.000000	1.000000	0.000(
max	64.000000	53.130000	5.000000	63770.428010	1.000000	1.000000	1.000(

Data Visualization

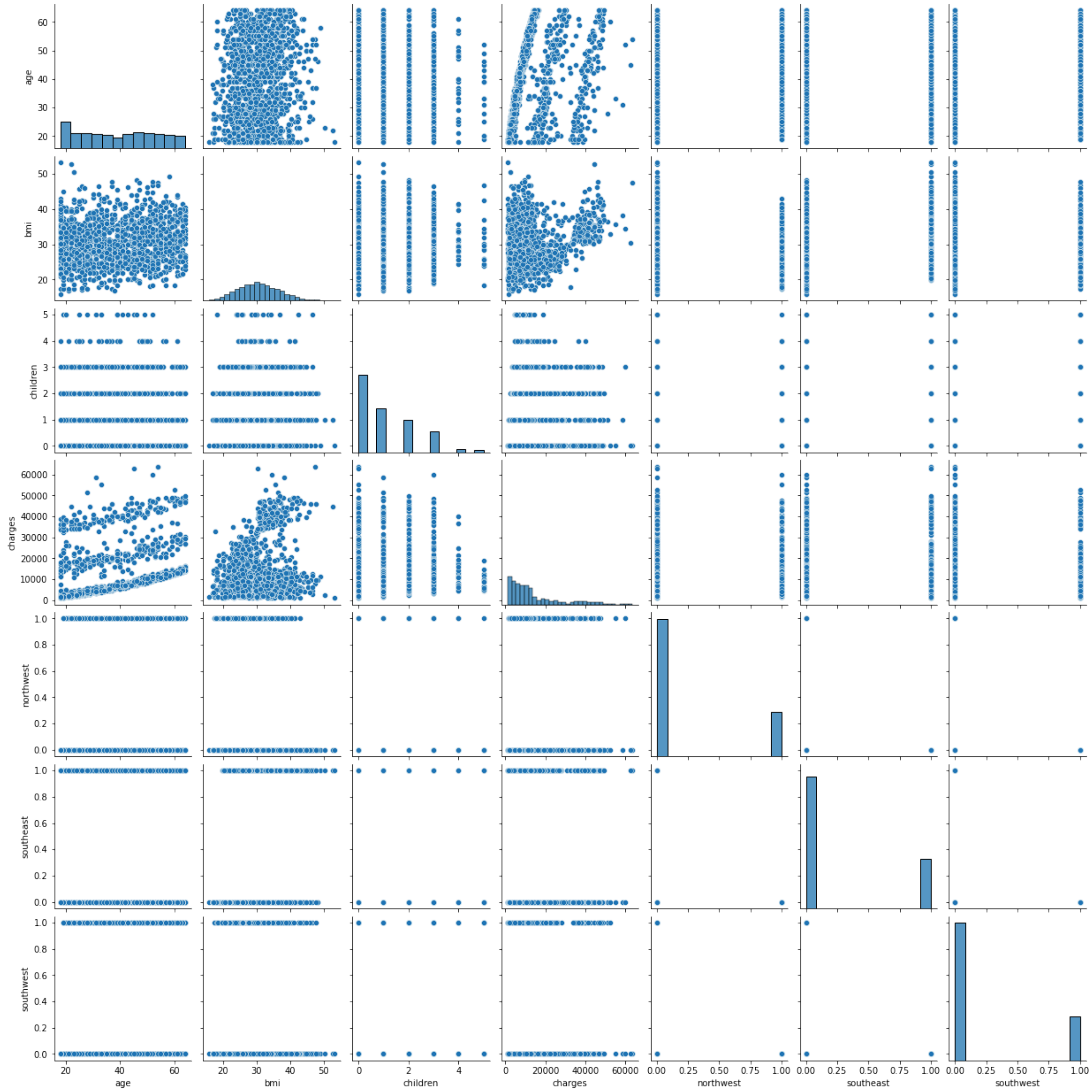
In [14]:
```
df[['customer_age', 'gender', 'body_mass_index', 'dependents', 'smoking_status
```

Out[14]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'bmi'}>],
       [<AxesSubplot:title={'center':'children'}>,
        <AxesSubplot:title={'center':'charges'}>]], dtype=object)
```



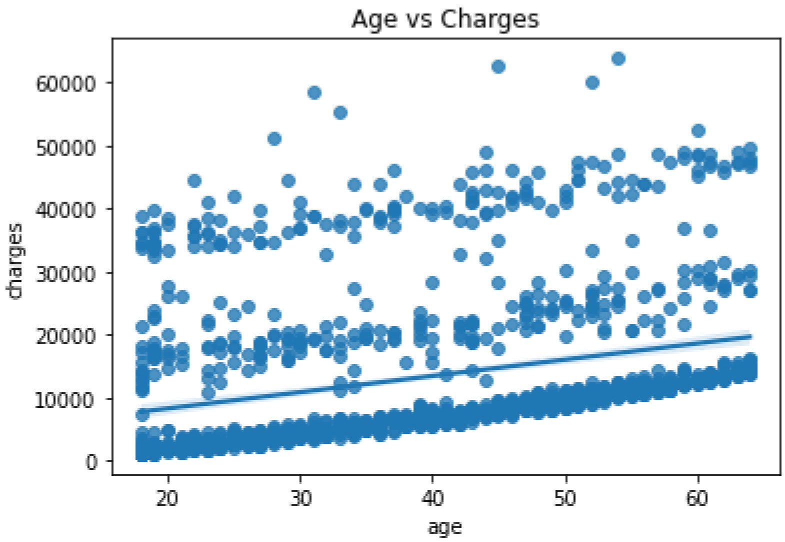
In [15]:
```
sns.pairplot(df)
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x2bc82aae6a0>

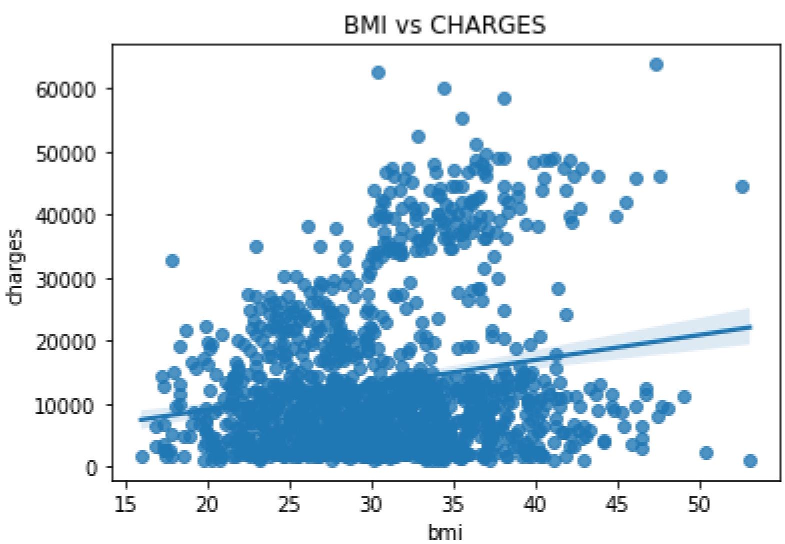


Check the realtionship between the customer_age and Charges, we can see the it is more complex than a linear relationship.

In [16]:
```
sns.regplot(x='customer_age', y='insurance_cost', data=df)
plt.title('Age vs Charges')
plt.show()
```



In [17]:
```
sns.regplot(x='body_mass_index', y='insurance_cost', data=df)
plt.title('BMI vs CHARGES')
plt.show()
```



Smoke have the most positive relationship with the insurance_cost

As we know that machine can understand only numbers, so let's convert it into numbers.

In [18]:
```python
from sklearn.preprocessing import LabelEncoder
#gender
le = LabelEncoder()
le.fit(df.gender.drop_duplicates())
df.gender = le.transform(df.gender)
#smoking_status or not
le.fit(df.smoking_status.drop_duplicates())
df.smoking_status = le.transform(df.smoking_status)
#location
# le.fit(df.location.drop_duplicates())
# df.location = le.tranaform(df.location)
```

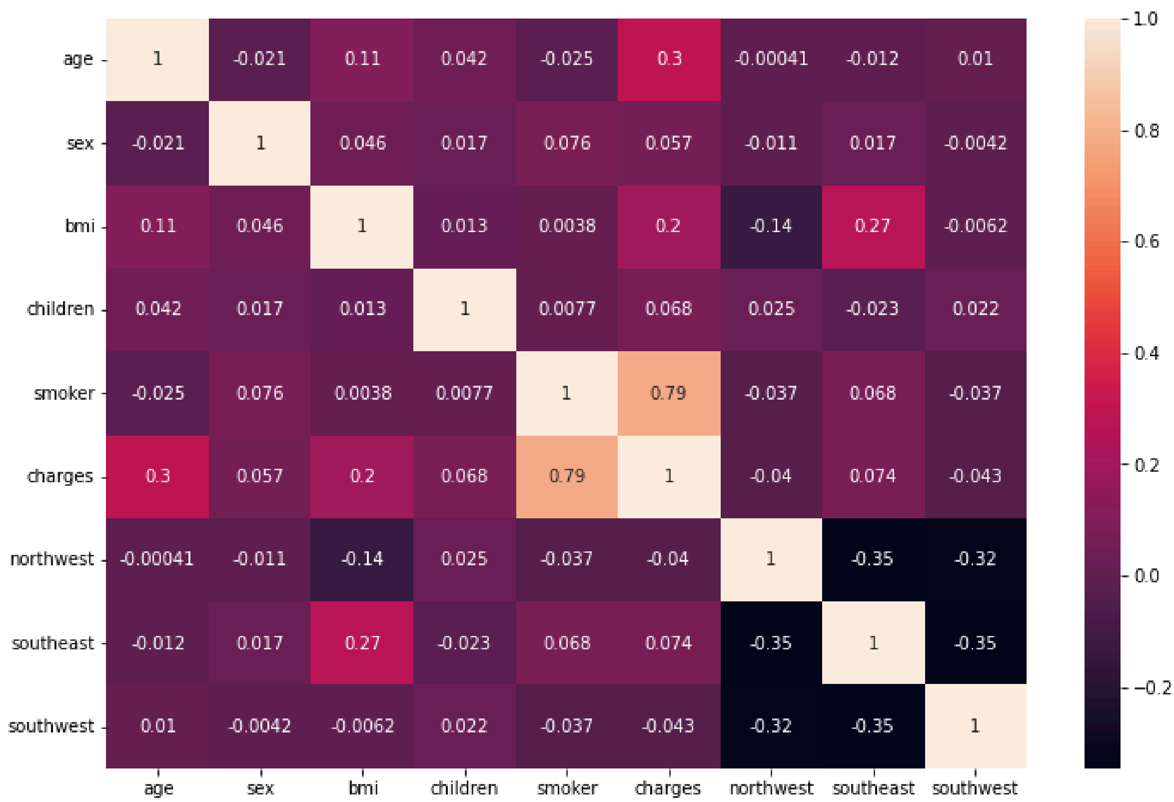
In [19]:
```python
df.head()
```

Out[19]:

	age	sex	bmi	children	smoker	charges	northwest	southeast	southwest
0	19	0	27.900	0	1	16884.92400	0	0	1
1	18	1	33.770	1	0	1725.55230	0	1	0
2	28	1	33.000	3	0	4449.46200	0	1	0
3	33	1	22.705	0	0	21984.47061	1	0	0
4	32	1	28.880	0	0	3866.85520	1	0	0

Correlation between different attriubtes of the data

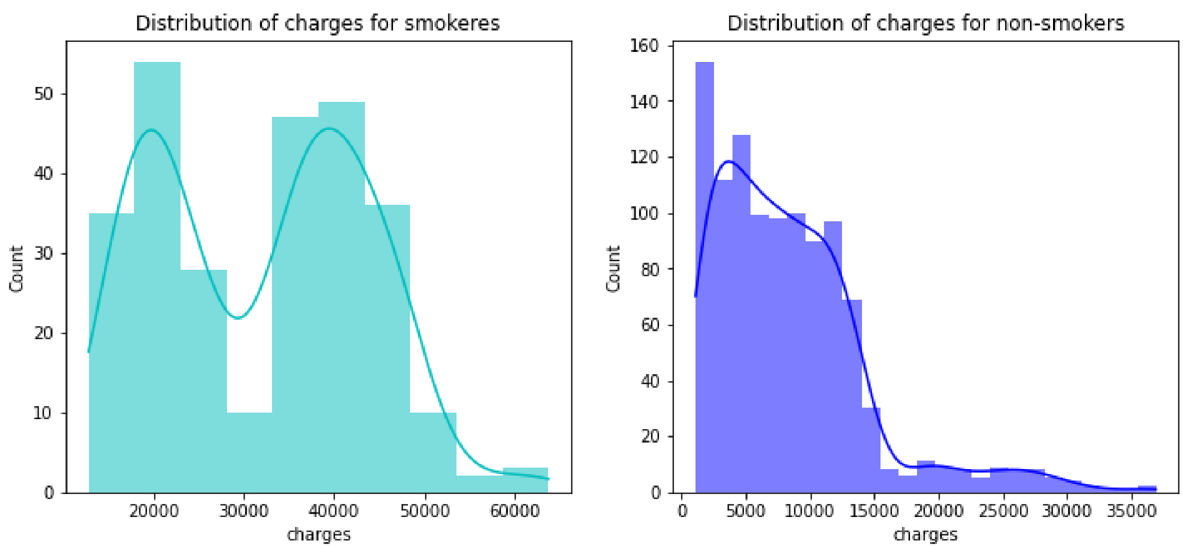
In [20]:
```python
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



Distrubtion of insurance_cost between smoking_statuss and non-smoking_statuss

In [21]:
```
f = plt.figure(figsize=(12,5))

ax = f.add_subplot(121)
sns.histplot(df[(df.smoking_status==1)]['insurance_cost'], color='c',ax=ax,kde
ax.set_title('Distribution of insurance_cost for smoking_statuses')

ax = f.add_subplot(122)
sns.histplot(df[(df.smoking_status==0)]['insurance_cost'],color='b',ax=ax,kde=
ax.set_title('Distribution of insurance_cost for non-smoking_statuss')
```

Out[21]:
```
Text(0.5, 1.0, 'Distribution of charges for non-smokers')
```

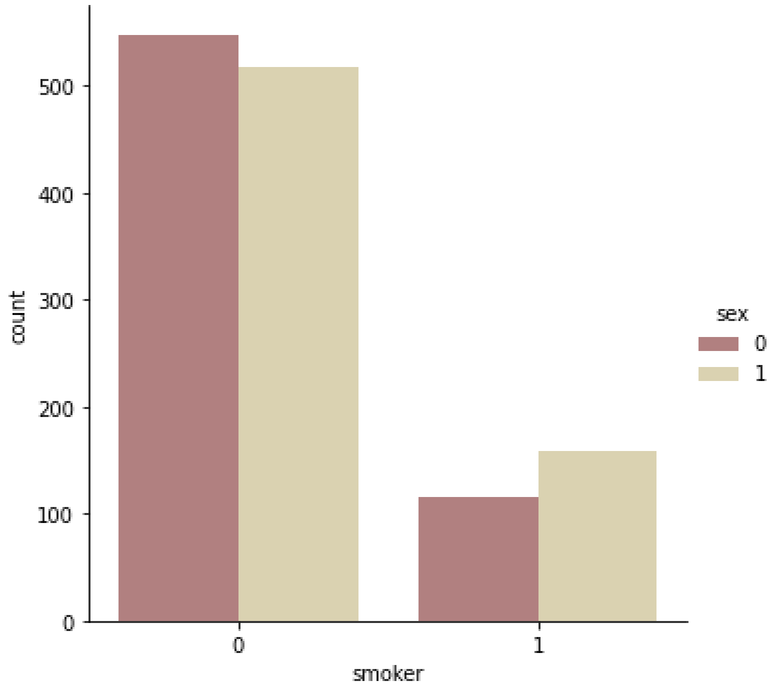


Smoking patients spend more on treatment. But there is a feeling that the number of non-smoking patients is greater. Going to check it.

In [22]:
```
plt.figure(figsize=(8,8))
sns.catplot(x='smoking_status', kind='count', hue='gender', palette='pink', da
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x2bc824c7ee0>
```

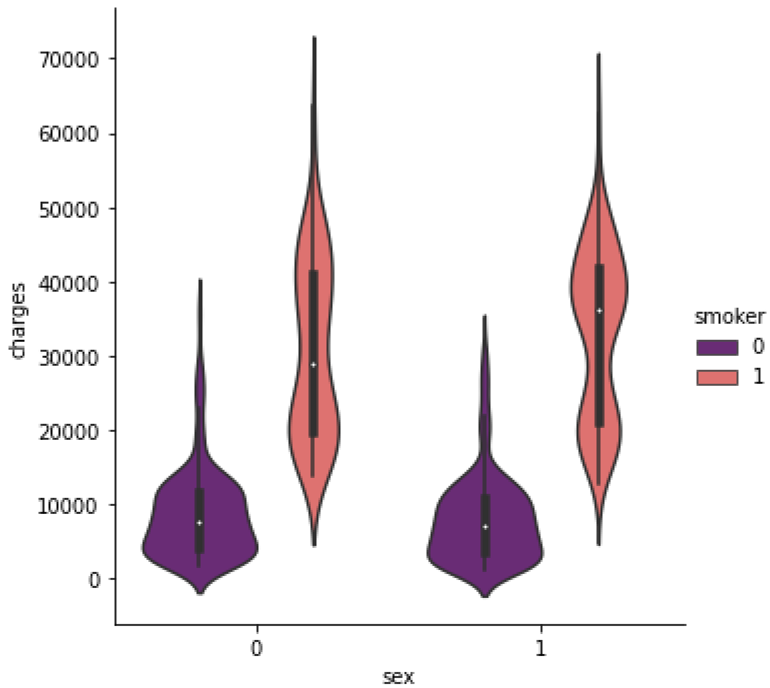
```
<Figure size 576x576 with 0 Axes>
```



Please note that women are coded with the symbol " 1 "and men "0". Thus non-smoking people and the truth more. Also we can notice that more male smoking_statuss than women smoking_statuss. It can be assumed that the total cost of treatment in men will be more than in women, given the impact of smoking. Maybe we'll check it out later. And some more useful visualizations.

In [23]:
```
sns.catplot(x="gender", y="insurance_cost", hue="smoking_status",
            kind="violin", data=df, palette = 'magma')
```

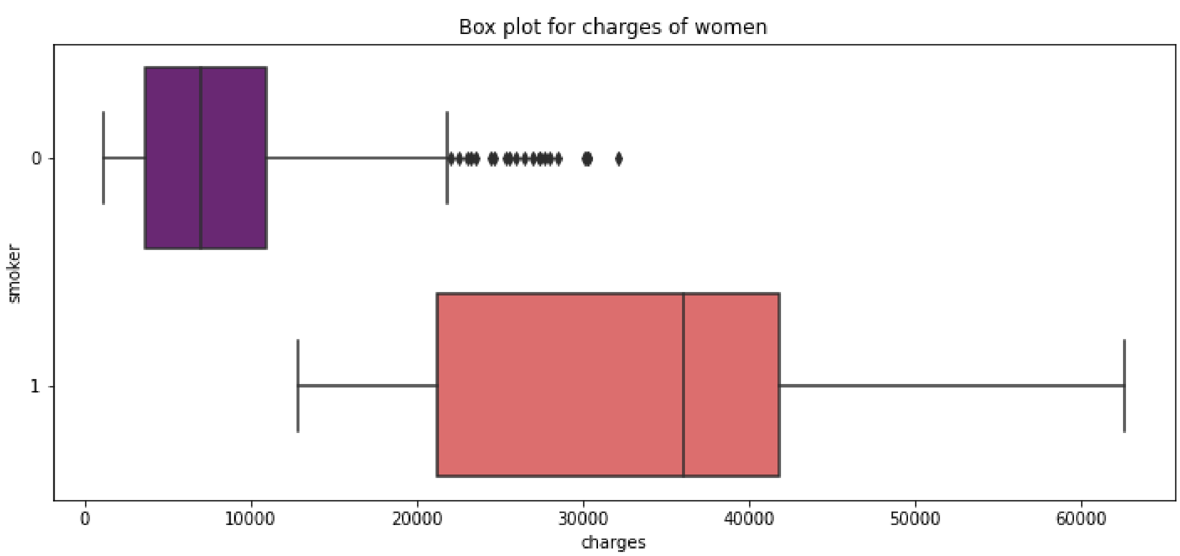
Out[23]: <seaborn.axisgrid.FacetGrid at 0x2bc85cc5b50>



Sex wise insurance_cost

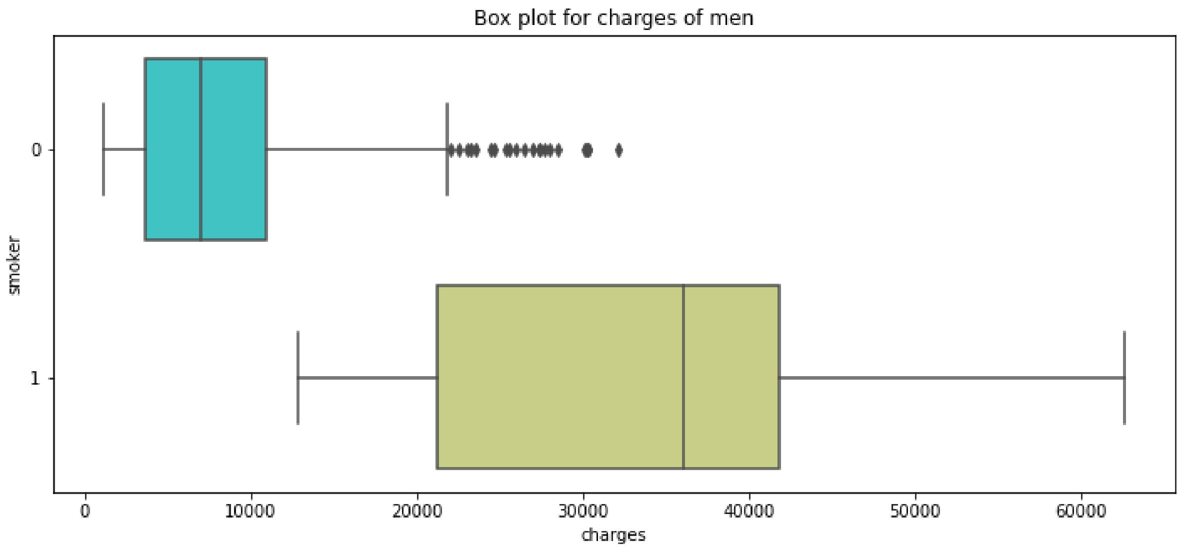
In [24]:
```
plt.figure(figsize=(12,5))
plt.title("Box plot for insurance_cost of women")
sns.boxplot(y='smoking_status',x='insurance_cost',data=df[(df.gender == 1)] , 
```

Out[24]: <AxesSubplot:title={'center':'Box plot for charges of women'}, xlabel='charges', ylabel='smoker'>



In [25]:
```
plt.figure(figsize=(12,5))
plt.title("Box plot for insurance_cost of men")
sns.boxplot(y='smoking_status',x='insurance_cost',data=df[(df.gender == 1)] , 
```

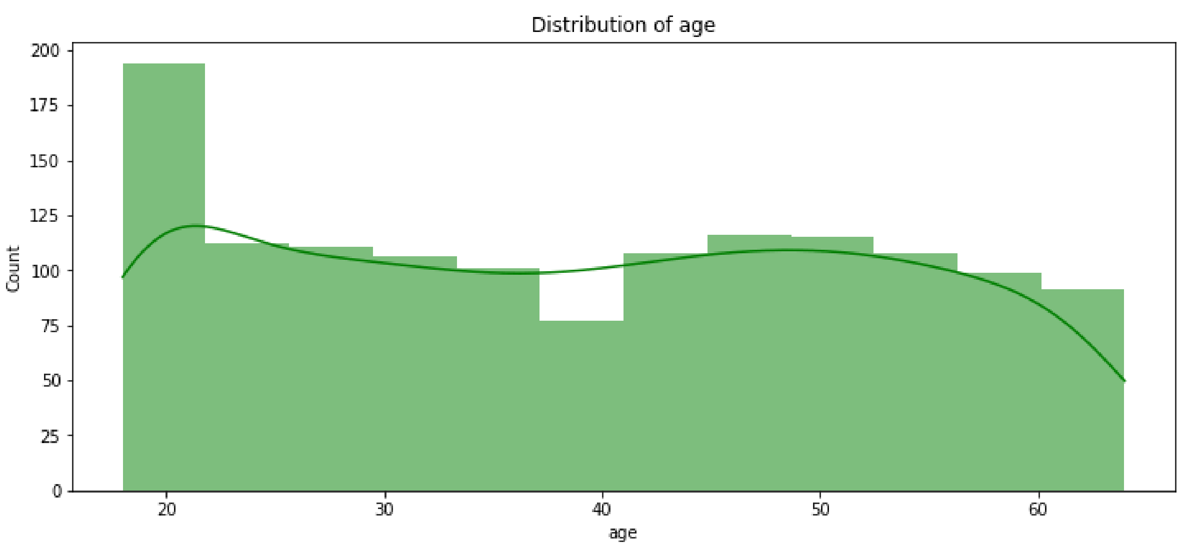
Out[25]:
```
<AxesSubplot:title={'center':'Box plot for charges of men'}, xlabel='charge
s', ylabel='smoker'>
```



Now let's pay attention to the customer_age of the patients. First, let's look at how customer_age affects the cost of treatment, and also look at patients of what customer_age more in our data set.

In [26]:
```
plt.figure(figsize=(12,5))
plt.title("Distribution of customer_age")
sns.histplot(df['customer_age'],color='g',kde=True, linewidth=0)
```

Out[26]:
```
<AxesSubplot:title={'center':'Distribution of age'}, xlabel='age', ylabel='Co
unt'>
```

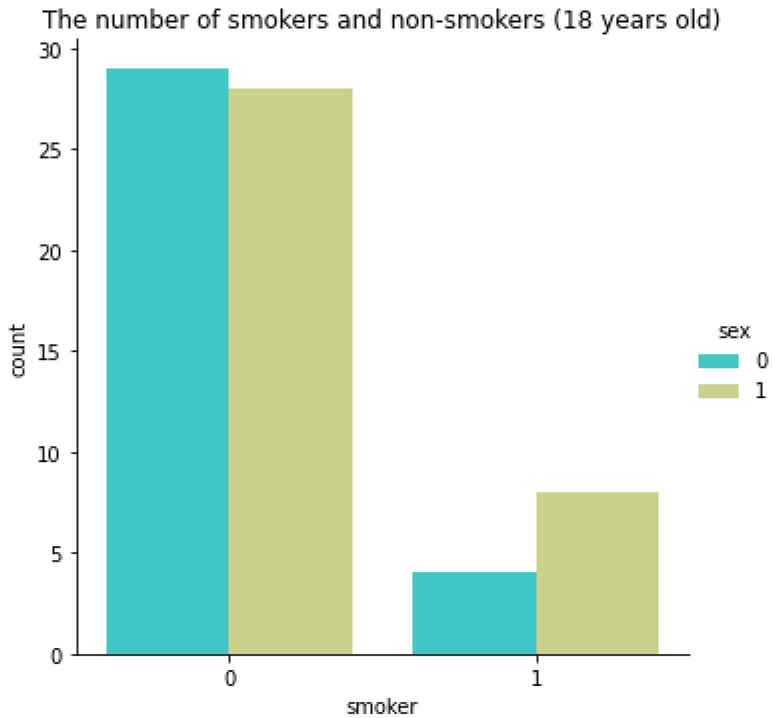


Young customer_age insurance_cost

Let's check whether there are smoking_statuss among pateints 18 years.

In [27]:
```
sns.catplot(x='smoking_status', kind='count', hue='gender', palette='rainbow',
plt.title("The number of smoking_statuss and non-smoking_statuss (18 years old
```

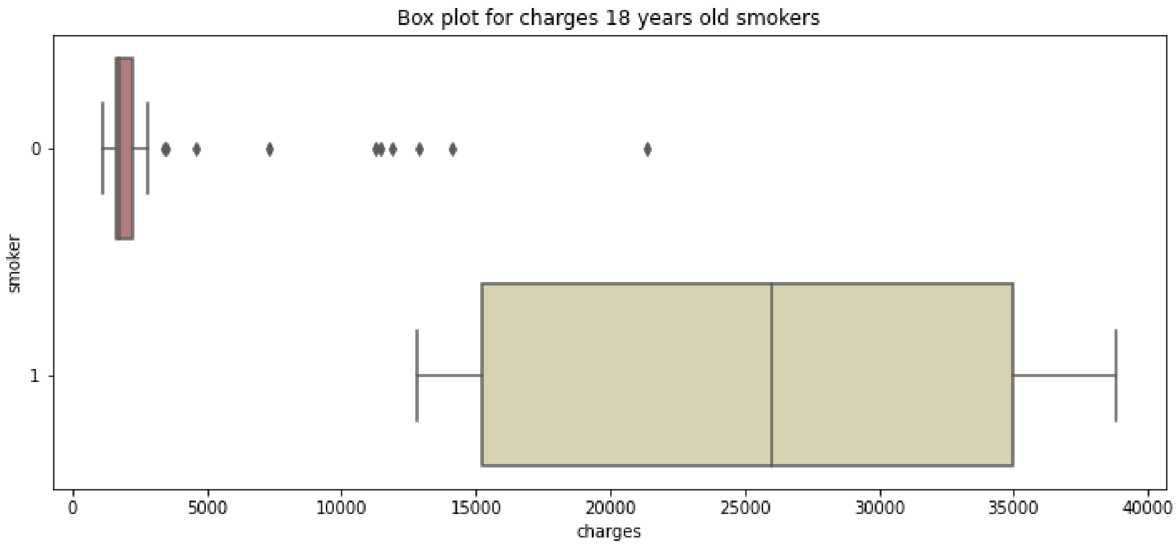
Out[27]:
```
Text(0.5, 1.0, 'The number of smokers and non-smokers (18 years old)')
```



For 18 years old, let's check the treatment cost of this customer_age.

In [28]:
```
plt.figure(figsize=(12,5))
plt.title("Box plot for insurance_cost 18 years old smoking_statuss")
sns.boxplot(y='smoking_status', x='insurance_cost', data=df[(df.customer_age ==
```

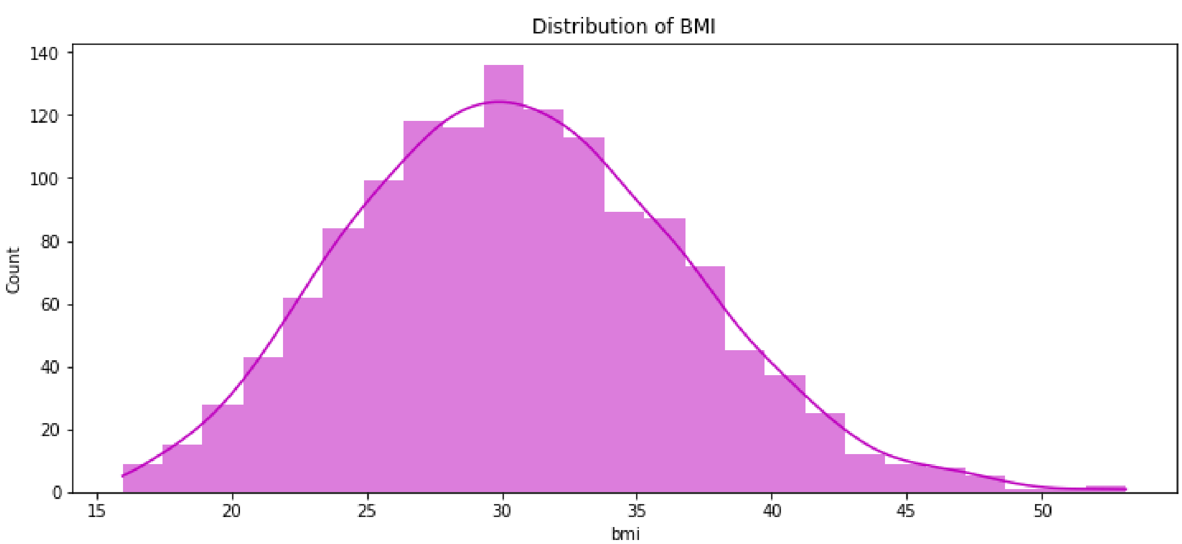
Out[28]:
```
<AxesSubplot:title={'center':'Box plot for charges 18 years old smokers'}, xl
abel='charges', ylabel='smoker'>
```



From above insight we can say that at the customer_age of 18 smoking_statuss spend much more on treatment than non-smoking_statuss. Also non-smoking_statuss are seeing some "tails". I can assume that this is due to serious disease or accidents.

Now let's check about BMI

In [29]:
```
plt.figure(figsize=(12,5))
plt.title("Distribution of BMI")
ax = sns.histplot(df['body_mass_index'],color="m",kde=True, linewidth=0)
```



The avercustomer_age BMI in patients is 30.

BMI range

BMI ----> Weight status

below 18.5 ----> Underweight

18.5 - 24.9 ----> Normal Weight

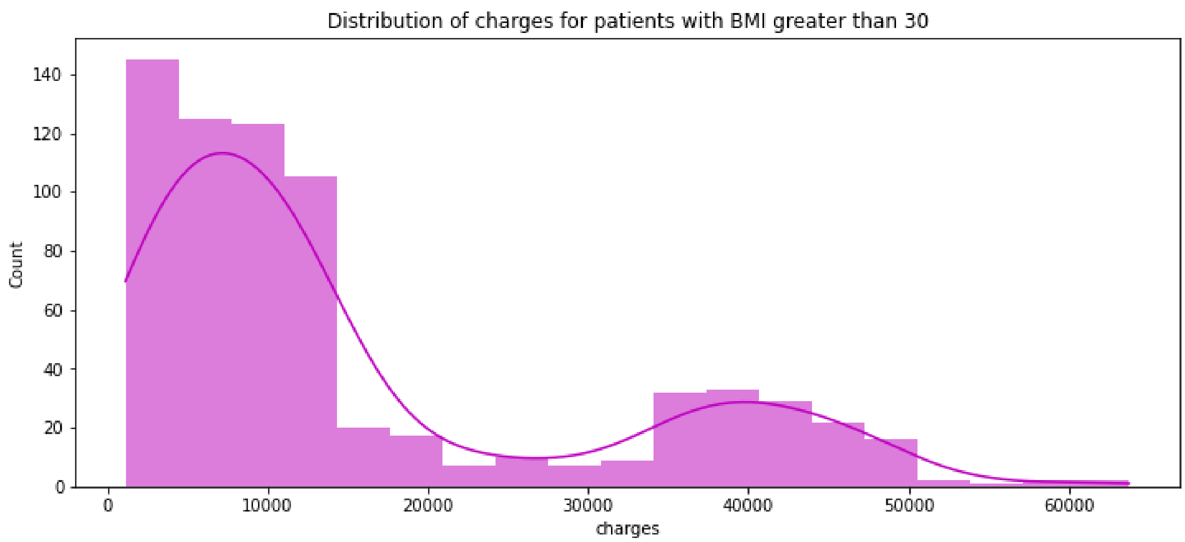
25.0 - 29.9 ----> Overweight

30.0 - 34.9 ----> Obeisty Class I

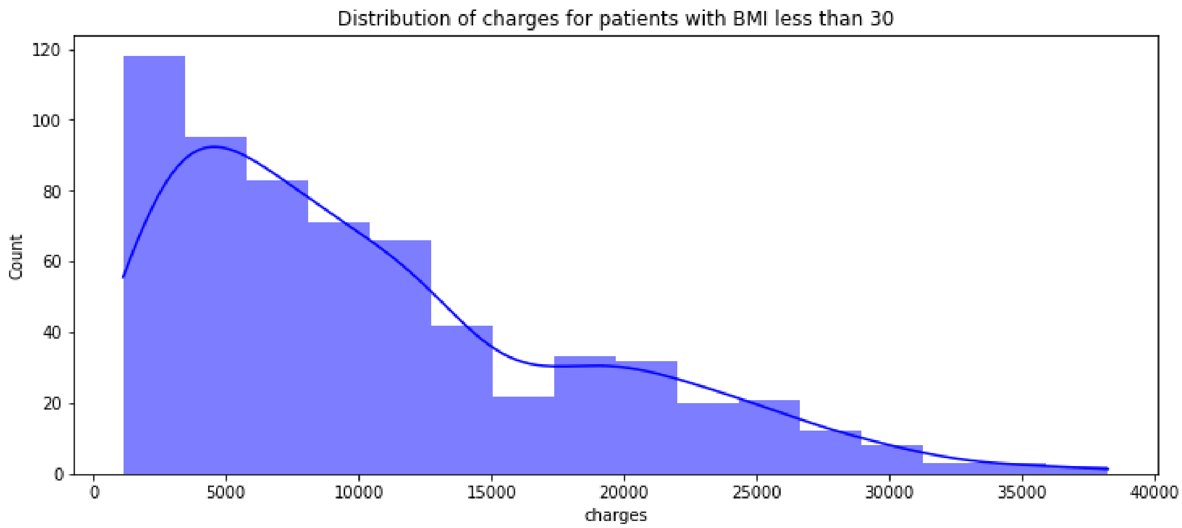
35.0 - 39.9 ----> Obesity Class II

Above 40 ----> Obesity Class III

In [30]:
```
plt.figure(figsize=(12,5))
plt.title("Distribution of insurance_cost for patients with BMI greater than 30
ax = sns.histplot(df[(df.body_mass_index >= 30)]['insurance_cost'], color = 'm
```



In [31]:
```
plt.figure(figsize=(12,5))
plt.title("Distribution of insurance_cost for patients with BMI less than 30")
ax = sns.histplot(df[(df.body_mass_index<30)]['insurance_cost'], color='b', kd
```

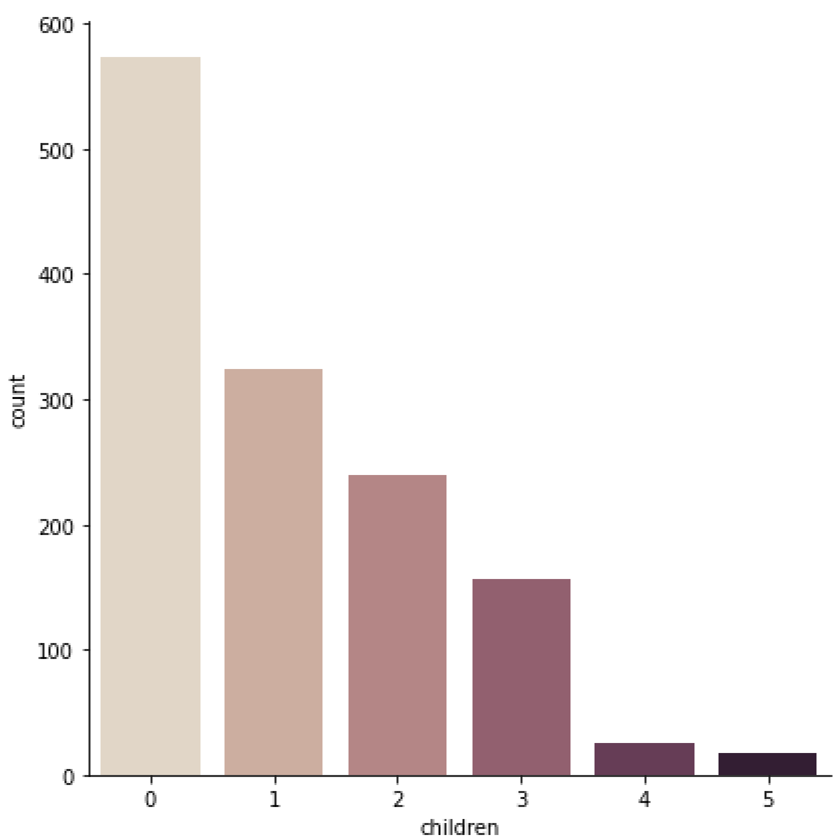


Patients with BMI above 30 spend more on treatment

Let's pay attention to dependents. First, let's see how many dependents our patients have.

In [32]:
```
sns.catplot(x="dependents", kind="count", palette="ch:.25",data=df, height=6)
```

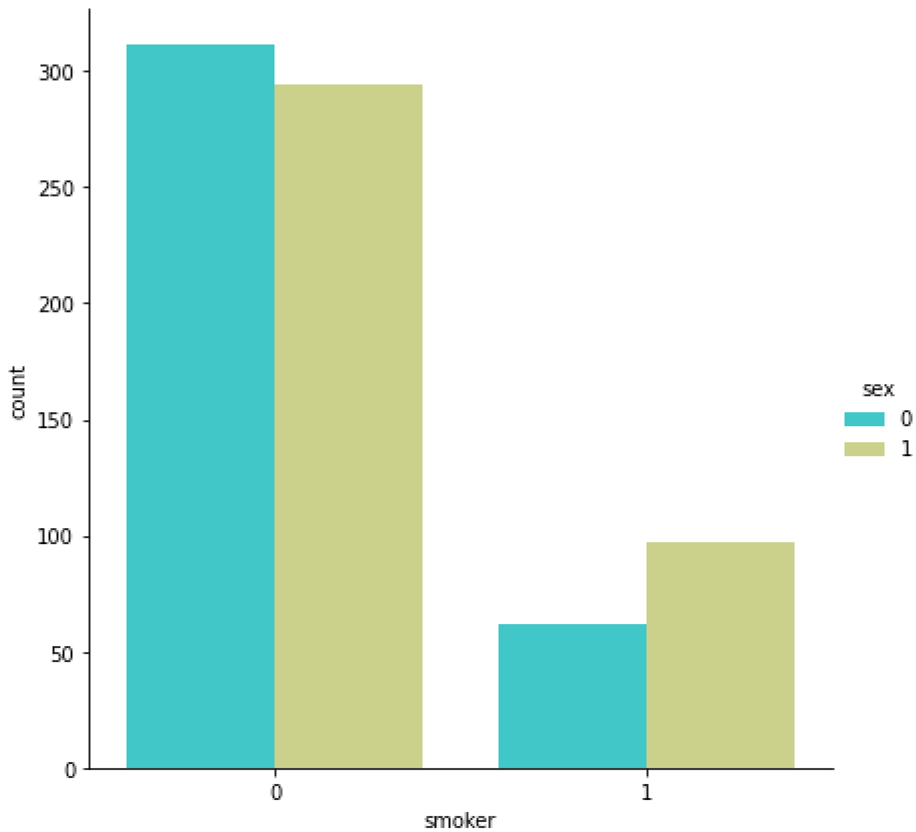
Out[32]: <seaborn.axisgrid.FacetGrid at 0x2bc8615b0d0>



Most patients do not have dependents.

In [33]:
```
sns.catplot(x="smoking_status", kind="count", palette="rainbow",hue="gender",d
ax.set_title("Smokers and non-smoking_statuss who have dependents")
```

Out[33]:
```
Text(0.5, 1.0, 'Smokers and non-smokers who have children')
```



Splitting the data into training and tesring

Charges is out target

In [34]:
```
X = df.drop(['insurance_cost'],axis=1)
y = df.insurance_cost

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_
print(X_train.shape)
print(X_test.shape)
```

```
(1070, 8)
(268, 8)
```

Creating function for model fitting and sccuracy

In [35]:
```
def model_pred(model):
    model.fit(X_train, y_train)
    print(model.score(X_test, y_test))
```

Linear Regression

In [36]:
```
from sklearn.linear_model import LinearRegression
```

In [37]:
```
lr = LinearRegression()
```

Hyperparameter optimization on the Random Forest Regressor for better result

Random Forest Regressor

In [38]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

#Random Forest Regressor
rfr = RandomForestRegressor()
param_grid = {
    'n_estimators': [100,200],
    'max_depth': [4,5,6,7,8],
    "max_features": ["auto", "sqrt", "log2"],
    'random_state': [0,1,42]
}
```

In [39]:
```
# RandomizedSearchCV
cv_random = RandomizedSearchCV(rfr, param_grid, cv=5)
cv_random.fit(X_train, y_train)
```

Out[39]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(),
                   param_distributions={'max_depth': [4, 5, 6, 7, 8],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'n_estimators': [100, 200],
                                        'random_state': [0, 1, 42]})
```

In [44]:
```
# Best Combination of parameters
cv_random.best_params_
```

Out[44]:
```
{'random_state': 42,
 'n_estimators': 100,
 'max_features': 'auto',
 'max_depth': 4}
```

In [45]:
```
rfReg = RandomForestRegressor(n_estimators=100, max_features='auto', max_depth
```

XGBoost Regressor

In [46]:
```
from xgboost import XGBRegressor

xgbR = XGBRegressor()

xgb_params = {
    'n_estimators' : [100, 200, 300],
    'max_depth' : [4, 6, 8, 10],
    'min_child_weight' : [2,4,10,12]
}
```

In [47]:
```
# GridSearchCV
cv_xgbR = GridSearchCV(xgbR, xgb_params, cv=5)
cv_xgbR.fit(X_train, y_train)
```

Out[47]:
```
GridSearchCV(cv=5,
             estimator=XGBRegressor(base_score=None, booster=None,
                                    colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None, gamma=None,
                                    gpu_id=None, importance_type='gain',
                                    interaction_constraints=None,
                                    learning_rate=None, max_delta_step=None,
                                    max_depth=None, min_child_weight=None,
                                    missing=nan, monotone_constraints=None,
                                    n_estimators=100, n_jobs=None,
                                    num_parallel_tree=None, random_state=Non
e,
                                    reg_alpha=None, reg_lambda=None,
                                    scale_pos_weight=None, subsample=None,
                                    tree_method=None, validate_parameters=Non
e,
                                    verbosity=None),
             param_grid={'max_depth': [4, 6, 8, 10],
                         'min_child_weight': [2, 4, 10, 12],
                         'n_estimators': [100, 200, 300]})
```

In [48]:
```
# Best combination of parameters
cv_xgbR.best_params_
```

Out[48]:
```
{'max_depth': 4, 'min_child_weight': 10, 'n_estimators': 100}
```

In [49]:
```
xgbReg = XGBRegressor(max_depth=4, min_child_weight=10, n_estimators=100)
```

Performance of the model

In [50]:
```
print("Score of Linear Regression:\n")
model_pred(lr)
```

```
Score of Linear Regression:

0.7999876970680433
```

In [51]:
```
print("Score of Random Forest Regressor:\n")
model_pred(rfReg)
```

```
Score of Random Forest Regressor:

0.8979119932825297
```

In [52]:
```
print("Score of XGBoost Regressor:\n")
model_pred(xgbReg)
```

```
Score of XGBoost Regressor:

0.8821278713659225
```

From above result we can say that Random Forest Regressor is better choice for this task

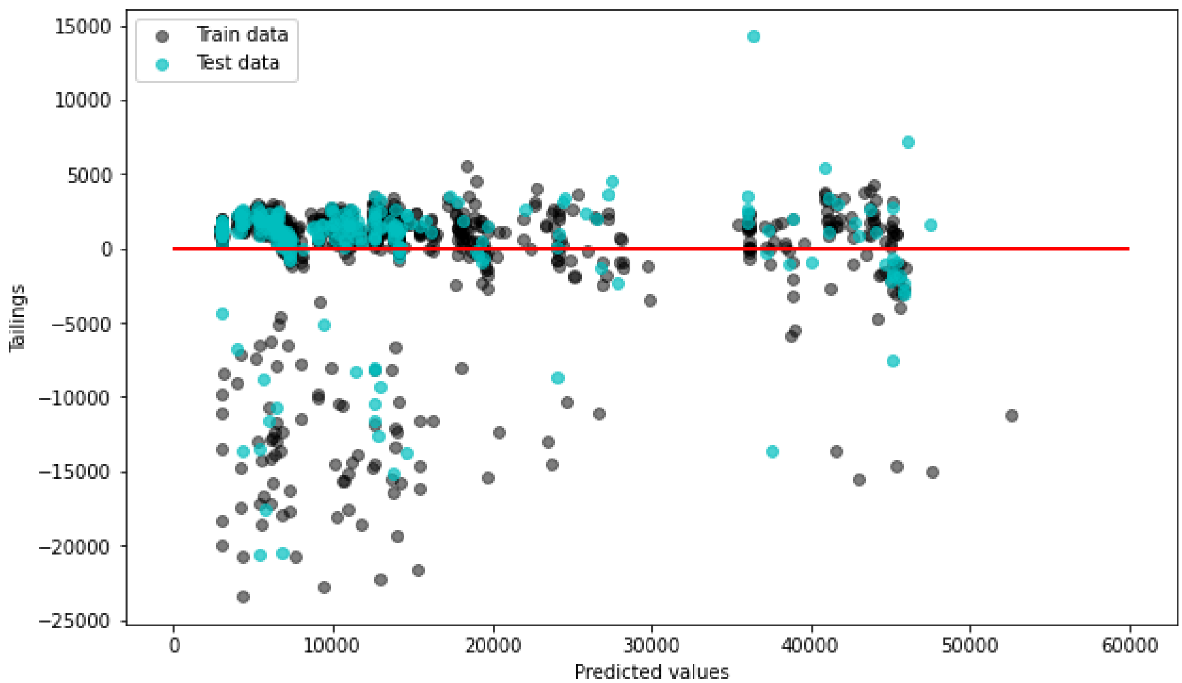
Evaluate Model with Random Forest Regressor

In [53]:
```
rfReg_train_pred = rfReg.predict(X_train)
rfReg_test_pred = rfReg.predict(X_test)

print("MSE train data: %.3f, MSE test data: %.3f"%(mean_squared_error(y_train,
print("R2 train data: %.3f, R2 test data: %.3f"%(r2_score(y_train, rfReg_train_
```

```
MSE train data: 19045053.031, MSE test data: 16245260.662
R2 train data: 0.867, R2 test data: 0.898
```

Actual vs Predicted

In [54]:
```
plt.figure(figsize=(10,6))

plt.scatter(rfReg_train_pred, rfReg_train_pred - y_train,
          c = 'black', marker = 'o', s = 35, alpha = 0.5,
          label = 'Train data')
plt.scatter(rfReg_test_pred,rfReg_test_pred - y_test,
          c = 'c', marker = 'o', s = 35, alpha = 0.7,
          label = 'Test data')
plt.xlabel('Predicted values')
plt.ylabel('Tailings')
plt.legend(loc = 'upper left')
plt.hlines(y = 0, xmin = 0, xmax = 60000, lw = 2, color = 'red')
plt.show()
```



Type *Markdown* and LaTeX: α^2