In [22]:

```
medical_charges_url = 'https://raw.githubusercontent.com/JovianML/opendatasets/master/dat
a/medical-charges.csv'
```

In [23]:

```
from urllib.request import urlretrieve
```

In [24]:

```
urlretrieve(medical_charges_url, 'medical.csv')
```

Out[24]:

```
('medical.csv', <http.client.HTTPMessage at 0x1f527dea070>)
```

In [25]:

```
!pip install pandas --quiet
```

In [26]:

```
import pandas as pd
```

In [27]:

```
medical_df = pd.read_csv('medical.csv')
```

In [28]:

```
medical_df
```

Out[28]:

	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
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

In [29]:

```
medical_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 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   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

Exploration Analysis and visuailization

In [30]:

```
!pip install plotly matplotlib seaborn --quiet
```

In [31]:

```
import plotly.express as px
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [32]:

```
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (10, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

In [33]:

```
medical_df.age.describe()
```

Out[33]:

```
count    1338.000000
mean       39.207025
std        14.049960
min        18.000000
25%        27.000000
50%        39.000000
75%        51.000000
max        64.000000
Name: age, dtype: float64
```

In [34]:

```
fig = px.histogram(medical_df,
                   x='age',
                   marginal='box',
                   nbins=47,
                   title='Distribution of Age')
fig.update_layout(bargap=0.1)
fig.show()
```

Body Mass Index

Let's look at the distribution of BMI (Body Mass Index) of customers, using a histogram and box plot.

In [35]:

```
fig = px.histogram(medical_df,
                   x='bmi',
                   marginal='box',
                   color_discrete_sequence=['red'],
                   title='Distribution of BMI (Body Mass Index)')
fig.update_layout(bargap=0.1)
fig.show()
```

Charges

Let's visualize the distribution of "charges" i.e. the annual medical charges for customers. This is the column we're trying to predict. Let's also use the categorical column "smoker" to distinguish the charges for smokers and non-smokers.

In [39]:

```
fig = px.histogram(medical_df,
                   x='charges',
                   marginal='box',
                   color='smoker',
                   color_discrete_sequence=['green', 'grey'],
                   title='Annual Medical Charges')
fig.update_layout(bargap=0.1)
fig.show()
```

Smoker

Let's visualize the distribution of the "smoker" column (containing values "yes" and "no") using a histogram.

In [40]:

```
medical_df.smoker.value_counts()
```

Out[40]:

```
no     1064
yes     274
Name: smoker, dtype: int64
```

In [41]:

```
px.histogram(medical_df, x='smoker', color='sex', title='Smoker')
```

Having looked at individual columns, we can now visualize the relationship between "charges" (the value we wish to predict) and other columns.

Age and Charges

Let's visualize the relationship between "age" and "charges" using a scatter plot. Each point in the scatter plot represents one customer. We'll also use values in the "smoker" column to color the points.

In [42]:

```
fig = px.scatter(medical_df,
                 x='age',
                 y='charges',
                 color='smoker',
                 opacity=0.8,
                 hover_data=['sex'],
                 title='Age vs. Charges')
fig.update_traces(marker_size=5)
fig.show()
```

BMI and Charges

Let's visualize the relationship between BMI (body mass index) and charges using another scatter plot. Once again, we'll use the values from the "smoker" column to color the points.

In [43]:

```
fig = px.scatter(medical_df,
                 x='bmi',
                 y='charges',
                 color='smoker',
                 opacity=0.8,
                 hover_data=['sex'],
                 title='BMI vs. Charges')
fig.update_traces(marker_size=5)
fig.show()
```

Correlation

As you can tell from the analysis, the values in some columns are more closely related to the values in "charges" compared to other columns. E.g. "age" and "charges" seem to grow together, whereas "bmi" and "charges" don't.

This relationship is often expressed numerically using a measure called the *correlation coefficient*, which can be computed using the `.corr` method of a Pandas series.

In [44]:

```
medical_df.charges.corr(medical_df.age)
```

Out[44]:

```
0.29900819333064765
```

In [45]:

```
medical_df.charges.corr(medical_df.bmi)
```

Out[45]:

```
0.19834096883362892
```

To compute the correlation for categorical columns, they must first be converted into numeric columns.

In [46]:

```
smoker_values = {'no': 0, 'yes': 1}
smoker_numeric = medical_df.smoker.map(smoker_values)
medical_df.charges.corr(smoker_numeric)
```

Out[46]:

```
0.7872514304984772
```

In [47]:

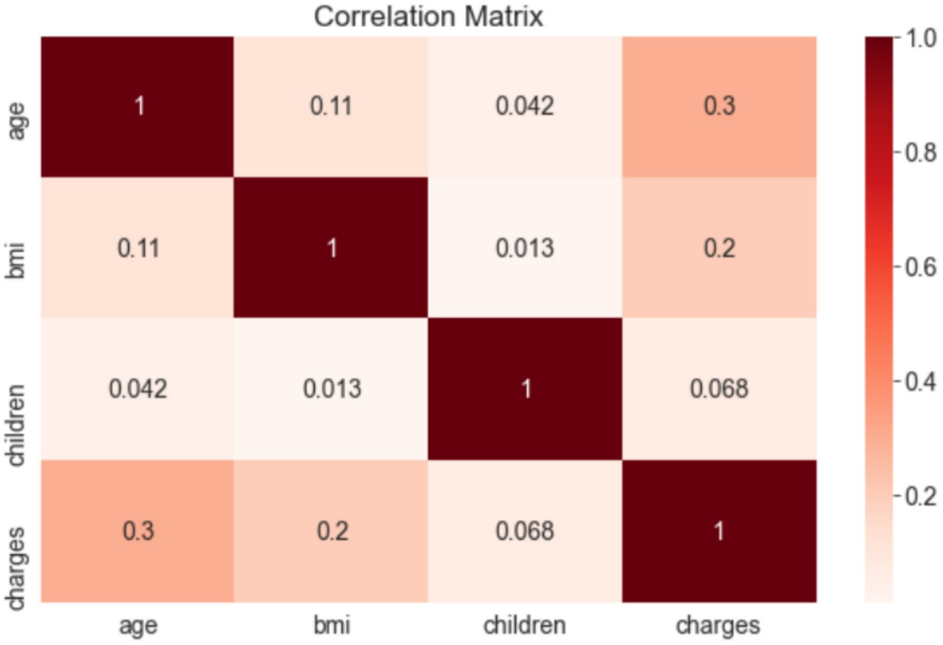
```
medical_df.corr()
```

Out[47]:

	age	bmi	children	charges
age	1.000000	0.109272	0.042469	0.299008
bmi	0.109272	1.000000	0.012759	0.198341
children	0.042469	0.012759	1.000000	0.067998
charges	0.299008	0.198341	0.067998	1.000000

In [48]:

```
sns.heatmap(medical_df.corr(), cmap='Reds', annot=True)
plt.title('Correlation Matrix');
```


Correlation Matrix
age
bmi
children
charges
age
bmi
children
charges