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

```
df = pd.read_csv('/content/insurance.csv')
```

```
df.head()
```

	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

Next steps: Generate code with df View recommended plots

```
df.shape
```

```
(1338, 7)
```

```
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
```

```
df.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

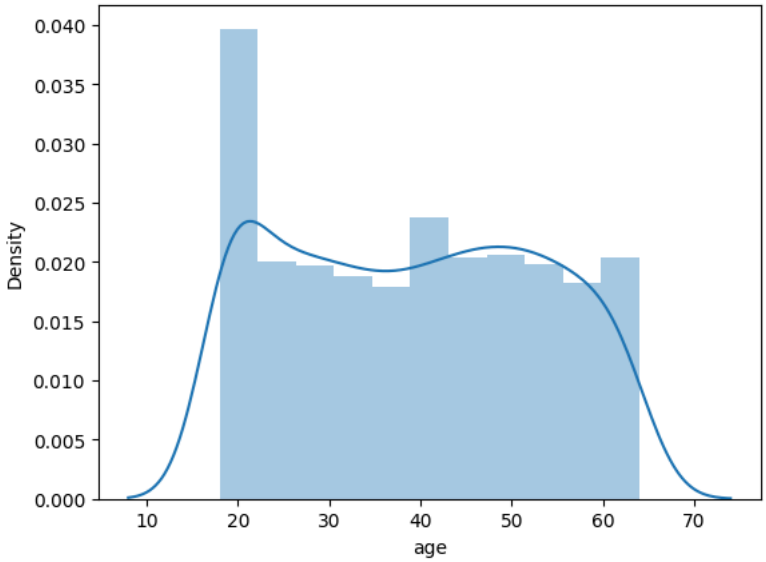
Handling Null values

```
df.isnull().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

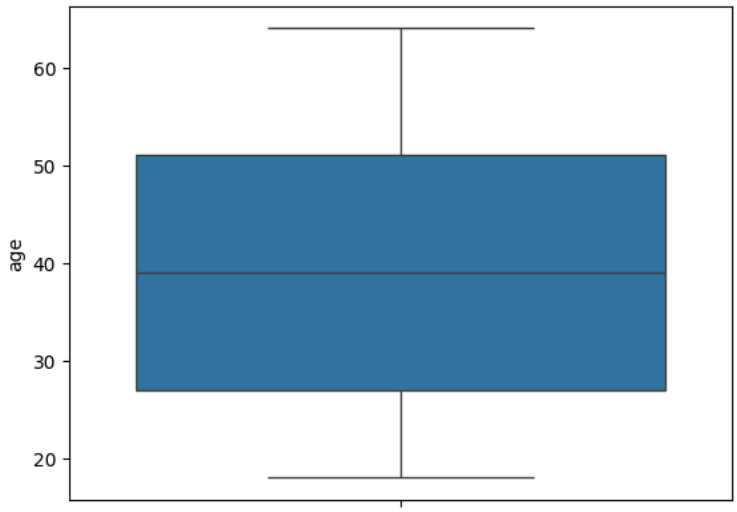
```
sns.distplot(df['age'])
```

```
<ipython-input-11-7452d86f8334>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['age'])
<Axes: xlabel='age', ylabel='Density'>
```




```
sns.boxplot(df['age'])
```

```
<Axes: ylabel='age'>
```



Detect Outlies of age column

1. Calculate first quartile(q1) and third quartile(q3)

```
q1 = df['age'].quantile(0.25)
q1
```

```
27.0
```

```
q3 = df['age'].quantile(0.75)
q3
```

```
51.0
```

```
IQR = q3-q1
IQR
```

```
24.0
```

Calculate Lower Bound q1 - 1.5IQR Calculate Upper Bound q3 + 1.5IQR

```
lowerBound = q1-(1.5*IQR)
upperBound = q3+(1.5*IQR)
print(lowerBound)
print(upperBound)
```

```
-9.0
87.0
```

```
df['age']=np.where(df['age']>upperBound,upperBound,df['age'])

print(df[df['age']>upperBound])
print(df[df['age']<lowerBound])
```

```
Empty DataFrame
Columns: [age, sex, bmi, children, smoker, region, charges]
Index: []
Empty DataFrame
Columns: [age, sex, bmi, children, smoker, region, charges]
Index: []
```

Now there are no outliers in age column

Encoding data using LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['sex']=le.fit_transform(df['sex'])
df['smoker']=le.fit_transform(df['smoker'])
df['region']=le.fit_transform(df['region'])
df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19.0	0	27.900	0	1	3	16884.92400
1	18.0	1	33.770	1	0	2	1725.55230
2	28.0	1	33.000	3	0	2	4449.46200
3	33.0	1	22.705	0	0	1	21984.47061
4	32.0	1	28.880	0	0	1	3866.85520

Next steps: Generate code with df View recommended plots

Seperating data into input and ouput/target

```
x = df.drop(columns=['charges'], axis=1)
y = df['charges']
print(x)
print(y)
```

```
        age  sex     bmi  children  smoker  region
0      19.0    0  27.900         0       1       3
1      18.0    1  33.770         1       0       2
2      28.0    1  33.000         3       0       2
3      33.0    1  22.705         0       0       1
4      32.0    1  28.880         0       0       1
...     ...  ...     ...       ...     ...     ...
1333   50.0    1  30.970         3       0       1
1334   18.0    0  31.920         0       0       0
1335   18.0    0  36.850         0       0       2
1336   21.0    0  25.800         0       0       3
1337   61.0    0  29.070         0       1       1

[1338 rows x 6 columns]
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
            ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
```

```
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

Scaling

```
from sklearn.preprocessing import StandardScaler
Scaler = StandardScaler()
```

```
X = Scaler.fit_transform(x)
X
```

```
array([[-1.43876426, -1.0105187 , -0.45332   , -0.90861367,  1.97058663,
         1.34390459],
       [-1.50996545,  0.98959079,  0.5096211 , -0.07876719, -0.5074631 ,
         0.43849455],
       [-0.79795355,  0.98959079,  0.38330685,  1.58092576, -0.5074631 ,
         0.43849455],
       ...,
       [-1.50996545, -1.0105187 ,  1.0148781 , -0.90861367, -0.5074631 ,
         0.43849455],
       [-1.29636188, -1.0105187 , -0.79781341, -0.90861367, -0.5074631 ,
         1.34390459],
       [ 1.55168573, -1.0105187 , -0.26138796, -0.90861367,  1.97058663,
        -0.46691549]])
```

```
x = pd.DataFrame(X)
```

Splitting Data

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
pred = lr.predict(x_test)
pred
```

```
array([11051.54909755,  9821.28110689, 37867.57220923, 16125.70579228,
        6920.27132517,  3879.38549816,  1448.91928088, 14390.17797974,
        9022.95151353,  7458.83371884,  4584.60125463, 10309.9886336 ,
        8693.03891958,  4085.25393494, 27551.60737718, 11151.0640722 ,
       11243.0536825 ,  5962.9521121 ,  8181.9015666 , 26750.7993431 ,
       33448.59842228, 14350.03320383, 11672.89478465, 32235.7832204 ,
        4326.07702625,  9096.53607025,  1045.25196369, 10177.76672094,
        4042.60346751, 10384.28681219,  9035.98983755, 40123.71002379,
       15507.20819083, 13678.264976  , 24481.97362849,  5059.21988589,
       12889.80609711, 30333.92625689, 33301.25091403,  3431.35787088,
        3941.81614597,  4203.90901434, 30219.19050725, 39245.56885373,
       27762.83744249,  4994.74188765, 11042.48621304,  7760.15047885,
        3569.09734756, 10613.61535955,  5544.95921408,  3397.80923785,
       32701.67144343, 38285.57836702, 16290.50463759,  6965.99677468,
        5895.27536963,  9364.94083823,  9395.1780384 , 11722.13868077,
        1611.87873326, 38750.4981005 , 15296.11225478, 11708.42958487,
       14076.39653066, 13904.28564489, 25798.46519738, 31953.12169371,
        1168.25915489, 10184.5995492 , 12273.00414884, 11867.15734569,
       24808.10644113, 15908.53043993, 11198.67421883, 12631.50869281,
        6433.71238434,  9915.55343233, 29953.19794316, 38768.07351788,
       12011.54405754, 37253.64166612,  4056.21325429,  9255.50826428,
       34537.73817683, 28976.62623495,  8444.02316285,  4738.69241453,
       11959.22562859, 30006.0695852 , 10041.58386562, 11243.48874027,
        8183.6075869 ,  9280.51490529,  8255.40224617,  7239.23538241,
       35731.00350944, 32878.29978853,  7591.7717691 , 14921.91368481,
        4184.53547122,  8690.01064385,  6619.75457992, 31535.59819898,
       32775.00677547,  1887.67848916,  8933.68024017,  6520.27249906,
       14475.77105663, 36880.82790297, 10252.51955517, 10775.16399139,
       10192.95246113, 26581.47470665, 39936.28907748,  8453.03671416,
         143.08142864,  8874.82383918, 15117.85425873,  9557.08594807,
       35275.59070316,  7270.62037452, 16826.50981439,  9572.8088055 ,
        8159.95902395,  2952.65859719, 32706.51413703, 31283.9896012 ,
       39216.89699401,  5362.49911669,  9675.40479836,  3778.85297694,
        7946.39718647,  8585.02883773, 31341.17050506, 29551.7714624 ,
       29853.91861524,  9151.88904567, 32625.66390263,  3229.01239018,
        3529.93652932, 11054.17156002, 13442.38216447, 12761.80223436,
```

```
 5363.70249634, 15875.56674406, 15252.72853146,  2382.17016287,
 -120.56014234, 10834.07802124,  7372.12214193, 31759.88622234,
12314.86913452,  2548.30390645,  6284.28252705,  8170.0107525 ,
 4285.24015268,  2331.14818812, 11414.21888159, 12551.18010753,
 7208.95663304, 16615.95420641, 11792.56220606, 13920.69808423,
 3134.30793579,  7262.13973297, 22758.38813544,  7596.99822972,
 5401.65993492,  5339.75438707,  6641.09944767,  5142.27041   ,
 9983.03913716,  5526.89132472,  5628.18992827,  6975.95618531,
 3673.17907317,  5521.32735633, 37913.25218948,  1337.01243212,
12636.06438156,  8935.78276524, 13661.56267036,  5572.770716  ,
 5181.38538205, 36214.23931831,  4207.49996636,  1896.75580314,
15163.16594007, 12674.02182014, 34823.20434979,  5093.20670396,
 5580.90282376, 31320.99694717,  5982.46375195,  1940.59597738,
 8389.18364163, 10016.84576515,  8238.45168712,  5687.97489766,
13133.993244  , 38538.79843345, 13749.62605459, 28607.07797491,
 6685.39503417, 35610.2777963 ,  3716.13611211, 12131.97274228,
 9356.80352592,  6339.94803517, 11268.82158683, 14519.98276598,
 5175.63113265,  4233.99203814,  7768.5658748 ,  1150.93881984,
 7861.51707835,  4401.34365822, 13351.6815701 ,  4312.6173173 ,
10007.10646576,  7274.06437597,  9167.81641515,  2307.24223958,
13115.48905979, 16739.15126554, 15287.81532153, 10516.20091624,
 5706.41694356,  2453.74757235,  2105.41152789, 13376.27751148,
```