

# Data Pre-Processing

In this document, we will try to adapt the data we have to the machine learning model by using various data sets.

In [3]:

```
import pandas as pd
import seaborn as sns

pd.set_option('display.max_columns', None)
pd.set_option('display.width', 500)
df = sns.load_dataset("titanic")
df.head()
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	NaN
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	NaN
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

◀ ▶

## Overview of the Titanic Dataset

In [4]:

```
from EDA import check_df

check_df(df)
```

```
#####
# Shape #####
(891, 15)
#####
# Types #####
survived      int64
pclass        int64
sex           object
age           float64
sibsp         int64
parch         int64
fare           float64
embarked      object
class          category
who            object
adult_male     bool
deck           category
embark_town    object
alive          object
alone          bool
dtype: object
#####
# Head #####
#####
# Tail #####
```

```

survived pclass      sex   age  sibsp  parch    fare embarked  class  who ad
ult_male deck embark_town alive alone
886       0        2 male  27.0     0     0  13.00      S Second man
True   NaN Southampton no  True
887       1        1 female 19.0     0     0  30.00      S First woman
False   B Southampton yes True
888       0        3 female  NaN     1     2  23.45      S Third woman
False   NaN Southampton no False
889       1        1 male  26.0     0     0  30.00      C First man
True   C Cherbourg yes True
890       0        3 male  32.0     0     0   7.75      Q Third man
True   Nan Queenstown no True
#####
survived          0
pclass            0
sex              0
age             177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck             688
embark_town      2
alive            0
alone            0
dtype: int64
#####
Quantiles #####
count      mean       std    min    0%    5%    50%    95%
99% 100% max
survived 891.0  0.383838  0.486592  0.00  0.00  0.000  0.0000  1.00000  1.00
000 1.0000 1.0000
pclass   891.0  2.308642  0.836071  1.00  1.00  1.000  3.0000  3.00000  3.00
000 3.0000 3.0000
age     714.0  29.699118  14.526497  0.42  0.42  4.000  28.0000  56.00000  65.87
000 80.0000 80.0000
sibsp   891.0  0.523008  1.102743  0.00  0.00  0.000  0.0000  3.00000  5.00
000 8.0000 8.0000
parch   891.0  0.381594  0.806057  0.00  0.00  0.000  0.0000  2.00000  4.00
000 6.0000 6.0000
fare    891.0  32.204208  49.693429  0.00  0.00  7.225  14.4542  112.07915  249.00
622 512.3292 512.3292

```

## Analysis of Categorical Variables

In [5]: `df["survived"].value_counts()`

Out[5]:

0	549
1	342
Name: survived, dtype: int64	

In [6]: `df["sex"].unique()`

Out[6]:

array(['male', 'female'], dtype=object)
---

```
In [7]: df["class"].nunique()
```

```
Out[7]: 3
```

```
In [8]: from EDA import *
cat_summary(df, 'sex')
```

	sex	Ratio
male	577	64.758698
female	314	35.241302
		#####

### Let's find out what are categorical variables

```
In [9]: cat_cols, num_cols, cat_but_car = grab_col_names(df)
cat_cols
```

```
Observations: 891
```

```
Variables: 15
```

```
cat_cols: 9
```

```
num_cols: 2
```

```
cat_but_car: 0
```

```
num_but_cat: 0
```

```
Out[9]: ['sex',
          'embarked',
          'class',
          'who',
          'adult_male',
          'deck',
          'embark_town',
          'alive',
          'alone']
```

```
In [10]: # Numerical columns
```

```
num_cols
```

```
Out[10]: ['age', 'fare']
```

```
In [11]: # Categorical but High cardinality columns
```

```
cat_but_car
```

```
Out[11]: []
```

```
In [12]: # Summary for categorical variables
from EDA import cat_summary
```

```
for col in cat_cols:
    cat_summary(df, col)
```

```
sex      Ratio
```

```

male      577  64.758698
female    314  35.241302
#####
    embarked      Ratio
S         644  72.278339
C         168  18.855219
Q          77   8.641975
#####
    class      Ratio
Third     491  55.106622
First     216  24.242424
Second    184  20.650954
#####
    who      Ratio
man       537  60.269360
woman     271  30.415264
child      83   9.315376
#####
    adult_male      Ratio
1         537  60.26936
0         354  39.73064
#####
    deck      Ratio
C         59   6.621773
B         47   5.274972
D         33   3.703704
E         32   3.591470
A         15   1.683502
F         13   1.459035
G          4   0.448934
#####
    embark_town      Ratio
Southampton        644  72.278339
Cherbourg          168  18.855219
Queenstown         77   8.641975
#####
    alive      Ratio
no        549  61.616162
yes      342  38.383838
#####
    alone      Ratio
1         537  60.26936
0         354  39.73064
#####

```

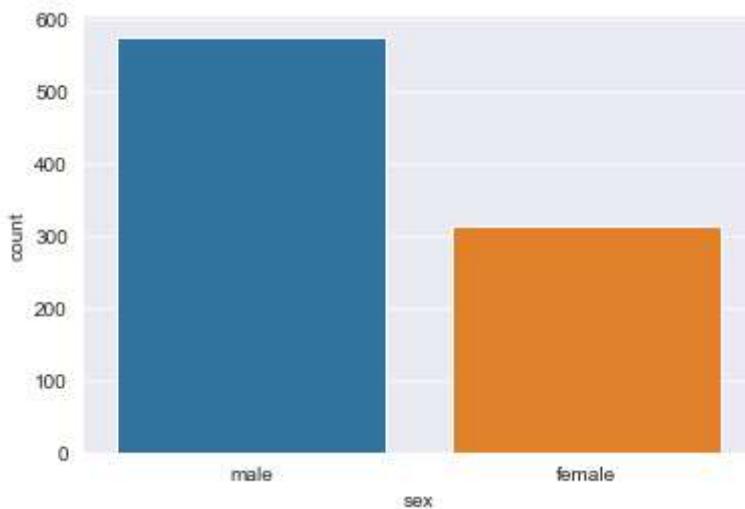
In [13]:

```
cat_summary(df, "sex", plot=True)
```

```

sex      Ratio
male    577  64.758698
female  314  35.241302
#####

```



## Analysis of Numerical Variables

In [15]:

```
num_summary(df, "age")
```

```
count    714.000000
mean     29.699118
std      14.526497
min      0.420000
5%       4.000000
10%      14.000000
20%      19.000000
30%      22.000000
40%      25.000000
50%      28.000000
60%      31.800000
70%      36.000000
80%      41.000000
90%      50.000000
95%      56.000000
99%      65.870000
max      80.000000
Name: age, dtype: float64
```

**Summary information for all numeric variables**

In [16]:

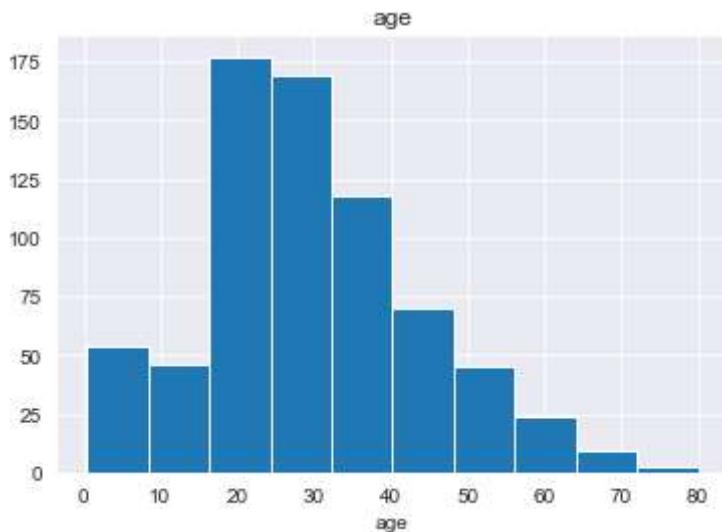
```
for col in num_cols:
    num_summary(df, col)
```

```
count    714.000000
mean     29.699118
std      14.526497
min      0.420000
5%       4.000000
10%      14.000000
20%      19.000000
30%      22.000000
40%      25.000000
50%      28.000000
60%      31.800000
```

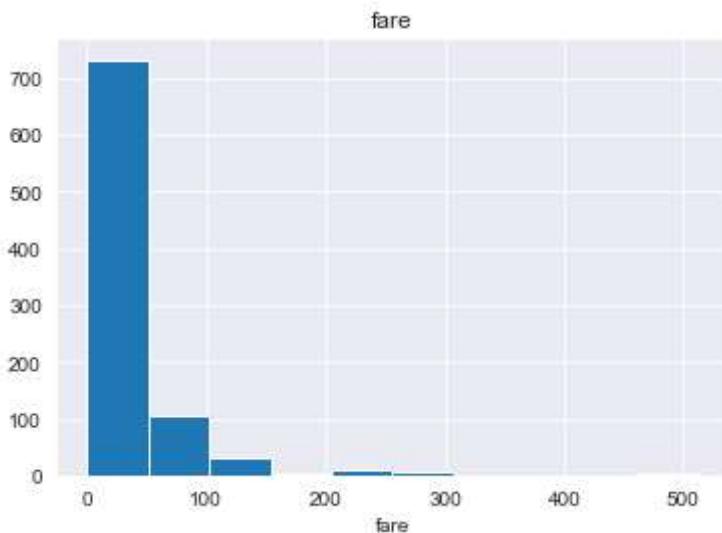
```
70%      36.000000
80%      41.000000
90%      50.000000
95%      56.000000
99%      65.870000
max      80.000000
Name: age, dtype: float64
count    891.000000
mean     32.204208
std      49.693429
min      0.000000
5%       7.225000
10%      7.550000
20%      7.854200
30%      8.050000
40%      10.500000
50%      14.454200
60%      21.679200
70%      27.000000
80%      39.687500
90%      77.958300
95%      112.079150
99%      249.006220
max      512.329200
Name: fare, dtype: float64
```

```
In [17]: for col in num_cols:
    num_summary(df, col, plot=True)
```

```
count    714.000000
mean     29.699118
std      14.526497
min      0.420000
5%       4.000000
10%      14.000000
20%      19.000000
30%      22.000000
40%      25.000000
50%      28.000000
60%      31.800000
70%      36.000000
80%      41.000000
90%      50.000000
95%      56.000000
99%      65.870000
max      80.000000
Name: age, dtype: float64
```



```
count      891.000000
mean       32.204208
std        49.693429
min        0.000000
5%         7.225000
10%        7.550000
20%        7.854200
30%        8.050000
40%        10.500000
50%        14.454200
60%        21.679200
70%        27.000000
80%        39.687500
90%        77.958300
95%        112.079150
99%        249.006220
max        512.329200
Name: fare, dtype: float64
```



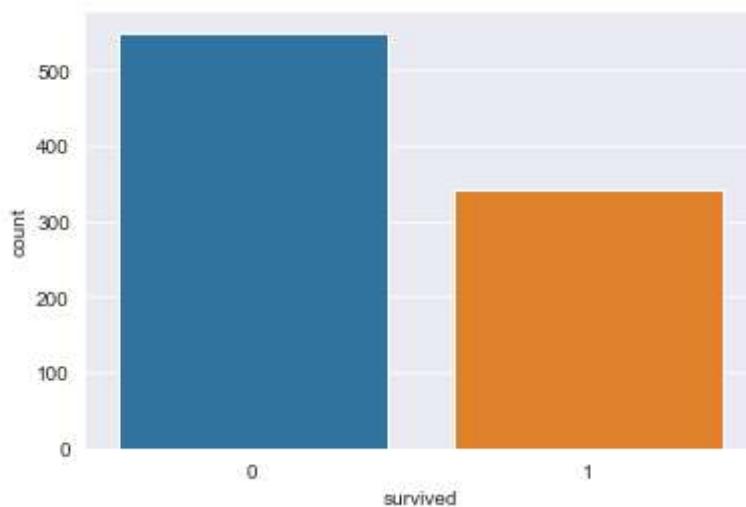
## Analysis of Target Variable

```
In [18]: df["survived"].value_counts()
```

```
Out[18]: 0    549  
1    342  
Name: survived, dtype: int64
```

```
In [19]: cat_summary(df, "survived", plot=True)
```

```
survived      Ratio  
0        549  61.616162  
1        342  38.383838  
#####
```



## Analysis of the Target Variable with Categorical Variables

```
In [20]: from EDA import target_summary_with_cat
```

```
target_summary_with_cat(df, "survived", "pclass")
```

```
TARGET_MEAN  
pclass  
1        0.629630  
2        0.472826  
3        0.242363
```

```
In [21]: for col in cat_cols:  
    target_summary_with_cat(df, "survived", col)
```

```
TARGET_MEAN  
sex  
female    0.742038  
male      0.188908
```

```
TARGET_MEAN  
embarked  
C        0.553571  
Q        0.389610
```

S 0.336957

TARGET\_MEAN  
class  
First 0.629630  
Second 0.472826  
Third 0.242363

TARGET\_MEAN  
who  
child 0.590361  
man 0.163873  
woman 0.756458

TARGET\_MEAN  
adult\_male  
0 0.717514  
1 0.163873

TARGET\_MEAN  
deck  
A 0.466667  
B 0.744681  
C 0.593220  
D 0.757576  
E 0.750000  
F 0.615385  
G 0.500000

TARGET\_MEAN  
embark\_town  
Cherbourg 0.553571  
Queenstown 0.389610  
Southampton 0.336957

TARGET\_MEAN  
alive  
no 0.0  
yes 1.0

TARGET\_MEAN  
alone  
0 0.505650  
1 0.303538

## Analysis of Target Variable with Numerical Variables

```
In [22]: from EDA import target_summary_with_num
```

```
for col in num_cols:  
    target_summary_with_num(df, "survived", col)
```

```
age  
survived  
0      30.626179  
1      28.343690
```

```
fare  
survived  
0      22.117887  
1      48.395408
```

## Analysis of Correlation

```
In [23]: from EDA import high_corr_cols
```

```
df = pd.read_csv("breast_cancer.csv")  
df = df.iloc[:, 1:-1]  
df.head()
```

Out[23]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactr
0	M	17.99	10.38	122.80	1001.0	0.11840	
1	M	20.57	17.77	132.90	1326.0	0.08474	
2	M	19.69	21.25	130.00	1203.0	0.10960	
3	M	11.42	20.38	77.58	386.1	0.14250	
4	M	20.29	14.34	135.10	1297.0	0.10030	

◀ ▶

In [24]:

```
cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

```
Observations: 569  
Variables: 31  
cat_cols: 1  
num_cols: 30  
cat_but_car: 0  
num_but_cat: 0
```

In [25]:

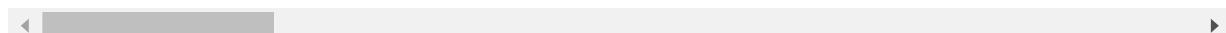
```
corr = df[num_cols].corr()  
corr
```

Out[25]:

```
radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean
```

.....

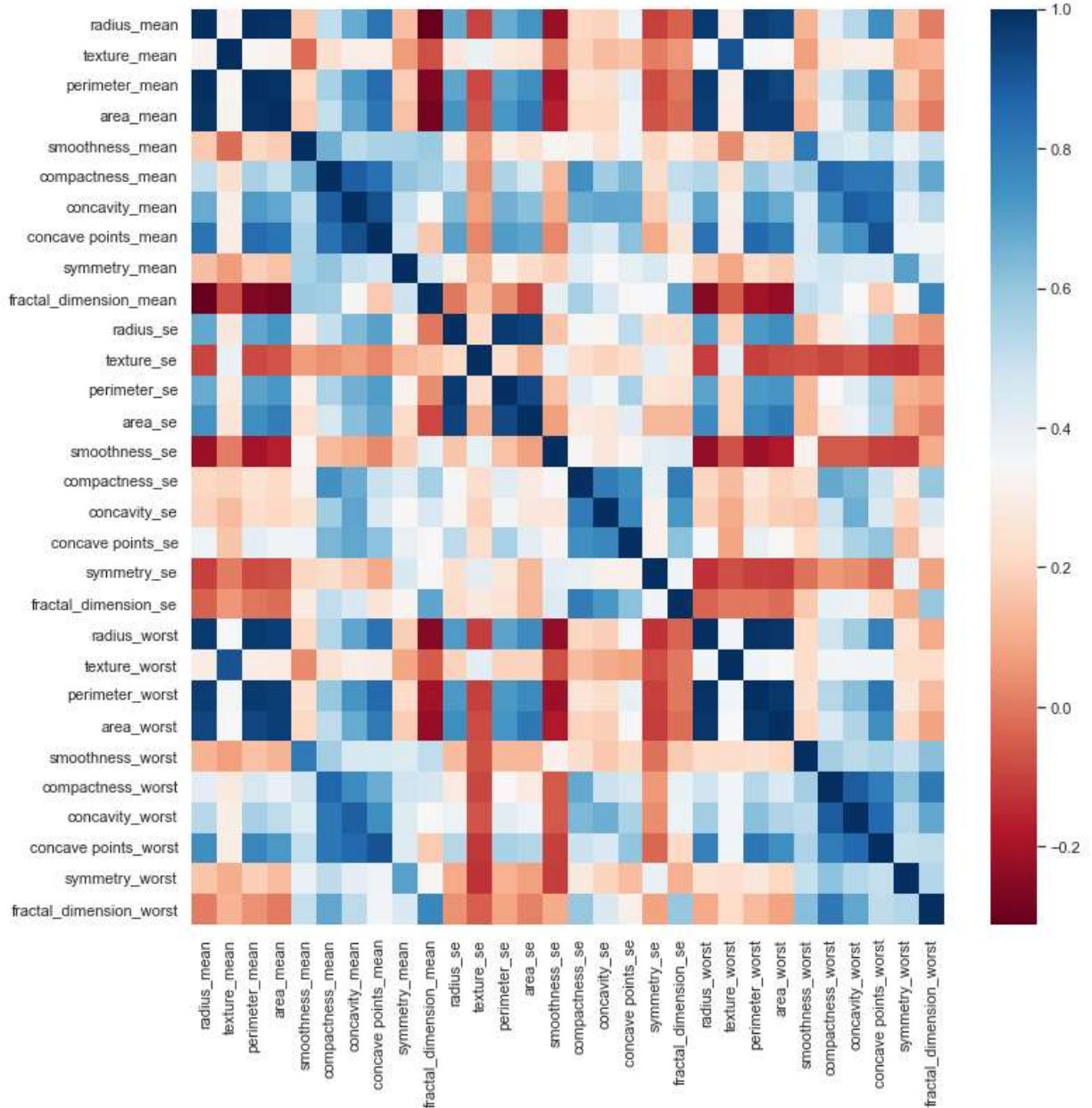
	<b>radius_mean</b>	<b>texture_mean</b>	<b>perimeter_mean</b>	<b>area_mean</b>	<b>smoothness_mean</b>
<b>radius_mean</b>	1.000000	0.323782	0.997855	0.987357	0.170581
<b>texture_mean</b>	0.323782	1.000000	0.329533	0.321086	-0.023389
<b>perimeter_mean</b>	0.997855	0.329533	1.000000	0.986507	0.207278
<b>area_mean</b>	0.987357	0.321086	0.986507	1.000000	0.177028
<b>smoothness_mean</b>	0.170581	-0.023389	0.207278	0.177028	1.000000
<b>compactness_mean</b>	0.506124	0.236702	0.556936	0.498502	0.659123
<b>concavity_mean</b>	0.676764	0.302418	0.716136	0.685983	0.521984
<b>concave points_mean</b>	0.822529	0.293464	0.850977	0.823269	0.553695
<b>symmetry_mean</b>	0.147741	0.071401	0.183027	0.151293	0.557775
<b>fractal_dimension_mean</b>	-0.311631	-0.076437	-0.261477	-0.283110	0.584792
<b>radius_se</b>	0.679090	0.275869	0.691765	0.732562	0.301467
<b>texture_se</b>	-0.097317	0.386358	-0.086761	-0.066280	0.068406
<b>perimeter_se</b>	0.674172	0.281673	0.693135	0.726628	0.296092
<b>area_se</b>	0.735864	0.259845	0.744983	0.800086	0.246552
<b>smoothness_se</b>	-0.222600	0.006614	-0.202694	-0.166777	0.332375
<b>compactness_se</b>	0.206000	0.191975	0.250744	0.212583	0.318943
<b>concavity_se</b>	0.194204	0.143293	0.228082	0.207660	0.248396
<b>concave points_se</b>	0.376169	0.163851	0.407217	0.372320	0.380676
<b>symmetry_se</b>	-0.104321	0.009127	-0.081629	-0.072497	0.200774
<b>fractal_dimension_se</b>	-0.042641	0.054458	-0.005523	-0.019887	0.283607
<b>radius_worst</b>	0.969539	0.352573	0.969476	0.962746	0.213120
<b>texture_worst</b>	0.297008	0.912045	0.303038	0.287489	0.036072
<b>perimeter_worst</b>	0.965137	0.358040	0.970387	0.959120	0.238853
<b>area_worst</b>	0.941082	0.343546	0.941550	0.959213	0.206718
<b>smoothness_worst</b>	0.119616	0.077503	0.150549	0.123523	0.805324
<b>compactness_worst</b>	0.413463	0.277830	0.455774	0.390410	0.472468
<b>concavity_worst</b>	0.526911	0.301025	0.563879	0.512606	0.434926
<b>concave points_worst</b>	0.744214	0.295316	0.771241	0.722017	0.503053
<b>symmetry_worst</b>	0.163953	0.105008	0.189115	0.143570	0.394309
<b>fractal_dimension_worst</b>	0.007066	0.119205	0.051019	0.003738	0.499316



In [26]:

```
sns.set(rc={'figure.figsize': (12, 12)})
sns.heatmap(corr, cmap="RdBu")
```

```
plt.show()
```

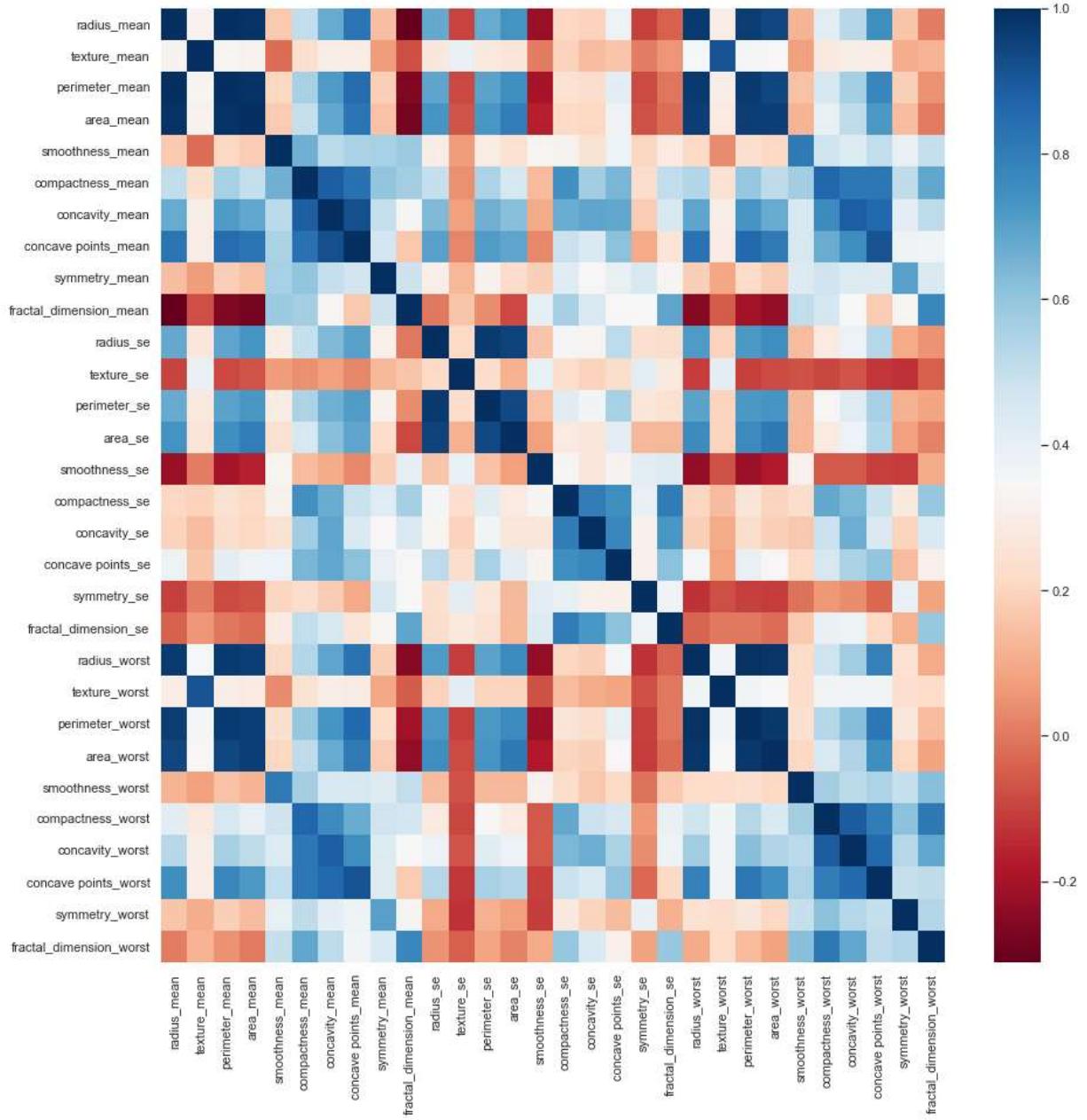


## Deleting Highly Correlation Variables

```
In [28]: high_corr_cols(df)
```

```
Out[28]: ['perimeter_mean',
 'area_mean',
 'concave points_mean',
 'perimeter_se',
 'area_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'concave points_worst']
```

```
In [29]: drop_list = high_corr_cols(df, plot=True)
```



```
In [30]: df.drop(drop_list, axis=1)
```

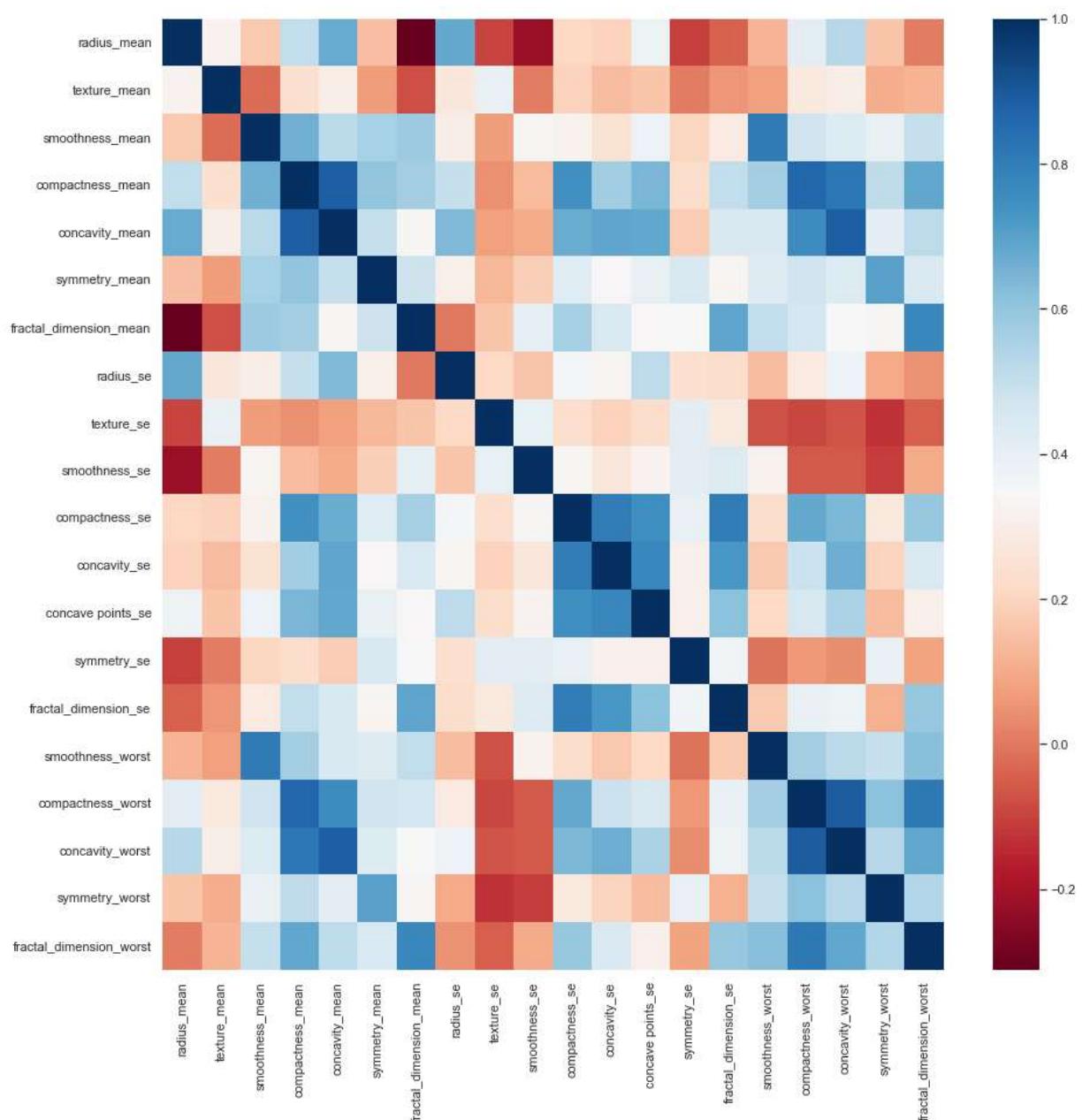
```
Out[30]:
```

	diagnosis	radius_mean	texture_mean	smoothness_mean	compactness_mean	concavity_mean
<b>0</b>	M	17.99	10.38	0.11840	0.27760	0.30010
<b>1</b>	M	20.57	17.77	0.08474	0.07864	0.08690
<b>2</b>	M	19.69	21.25	0.10960	0.15990	0.19740
<b>3</b>	M	11.42	20.38	0.14250	0.28390	0.24140
<b>4</b>	M	20.29	14.34	0.10030	0.13280	0.19800
...	...	...	...	...	...	...
<b>564</b>	M	21.56	22.39	0.11100	0.11590	0.24390

	diagnosis	radius_mean	texture_mean	smoothness_mean	compactness_mean	concavity_mean
565	M	20.13	28.25	0.09780	0.10340	0.14400
566	M	16.60	28.08	0.08455	0.10230	0.09251
567	M	20.60	29.33	0.11780	0.27700	0.35140
568	B	7.76	24.54	0.05263	0.04362	0.00000

569 rows × 21 columns

```
In [31]: high_corr_cols(df.drop(drop_list, axis=1), plot=True)
```



```
Out[31]: []
```

# How to Catch Outliers?

```
In [32]: from PreProcessing import outlier_thresholds  
  
df = sns.load_dataset('titanic')
```

```
In [33]: outlier_thresholds(df, "age")
```

```
Out[33]: (-6.6875, 64.8125)
```

```
In [34]: outlier_thresholds(df, "fare")
```

```
Out[34]: (-26.724, 65.6344)
```

```
In [35]: low, up = outlier_thresholds(df, "fare")  
low, up
```

```
Out[35]: (-26.724, 65.6344)
```

## Is there any outlier value?

```
In [37]: from PreProcessing import check_outlier  
  
check_outlier(df, "age")
```

```
Out[37]: True
```

```
In [38]: check_outlier(df, "fare")
```

```
Out[38]: True
```

```
In [39]: df = pd.read_csv('application_train.csv')  
  
cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

```
Observations: 307511  
Variables: 122  
cat_cols: 19  
num_cols: 61  
cat_but_car: 1  
num_but_cat: 4
```

```
In [40]: num_cols = [col for col in num_cols if col not in "SK_ID_CURR"]  
num_cols
```

```
Out[40]: ['AMT_INCOME_TOTAL',
```

'AMT\_CREDIT',  
'AMT\_ANNUITY',  
'AMT\_GOODS\_PRICE',  
'REGION\_POPULATION\_RELATIVE',  
'DAYS\_REGISTRATION',  
'OWN\_CAR\_AGE',  
'CNT\_FAM\_MEMBERS',  
'EXT\_SOURCE\_1',  
'EXT\_SOURCE\_2',  
'EXT\_SOURCE\_3',  
'APARTMENTS\_AVG',  
'BASEMENTAREA\_AVG',  
'YEARS\_BEGINEXPLUATATION\_AVG',  
'YEARS\_BUILD\_AVG',  
'COMMONAREA\_AVG',  
'ELEVATORS\_AVG',  
'ENTRANCES\_AVG',  
'FLOORSMAX\_AVG',  
'FLOORSMIN\_AVG',  
'LANDAREA\_AVG',  
'LIVINGAPARTMENTS\_AVG',  
'LIVINGAREA\_AVG',  
'NONLIVINGAPARTMENTS\_AVG',  
'NONLIVINGAREA\_AVG',  
'APARTMENTS\_MODE',  
'BASEMENTAREA\_MODE',  
'YEARS\_BEGINEXPLUATATION\_MODE',  
'YEARS\_BUILD\_MODE',  
'COMMONAREA\_MODE',  
'ELEVATORS\_MODE',  
'ENTRANCES\_MODE',  
'FLOORSMAX\_MODE',  
'FLOORSMIN\_MODE',  
'LANDAREA\_MODE',  
'LIVINGAPARTMENTS\_MODE',  
'LIVINGAREA\_MODE',  
'NONLIVINGAPARTMENTS\_MODE',  
'NONLIVINGAREA\_MODE',  
'APARTMENTS\_MEDI',  
'BASEMENTAREA\_MEDI',  
'YEARS\_BEGINEXPLUATATION\_MEDI',  
'YEARS\_BUILD\_MEDI',  
'COMMONAREA\_MEDI',  
'ELEVATORS\_MEDI',  
'ENTRANCES\_MEDI',  
'FLOORSMAX\_MEDI',  
'FLOORSMIN\_MEDI',  
'LANDAREA\_MEDI',  
'LIVINGAPARTMENTS\_MEDI',  
'LIVINGAREA\_MEDI',  
'NONLIVINGAPARTMENTS\_MEDI',  
'NONLIVINGAREA\_MEDI',  
'TOTALAREA\_MODE',  
'OBS\_30\_CNT\_SOCIAL\_CIRCLE',  
'DEF\_30\_CNT\_SOCIAL\_CIRCLE',  
'OBS\_60\_CNT\_SOCIAL\_CIRCLE',  
'DAYS\_LAST\_PHONE\_CHANGE',  
'AMT\_REQ\_CREDIT\_BUREAU\_MON',  
'AMT\_REQ\_CREDIT\_BUREAU\_QRT',  
'AMT\_REQ\_CREDIT\_BUREAU\_YEAR']

In [41]:

```
for col in num_cols:  
    print(col, check_outlier(df, col))  
  
AMT_INCOME_TOTAL True  
AMT_CREDIT True  
AMT_ANNUITY True  
AMT_GOODS_PRICE True  
REGION_POPULATION_RELATIVE True  
DAYS_REGISTRATION True  
OWN_CAR_AGE True  
CNT_FAM_MEMBERS True  
EXT_SOURCE_1 False  
EXT_SOURCE_2 False  
EXT_SOURCE_3 False  
APARTMENTS_AVG True  
BASEMENTAREA_AVG True  
YEARS_BEGINEXPLUATATION_AVG True  
YEARS_BUILD_AVG True  
COMMONAREA_AVG True  
ELEVATORS_AVG True  
ENTRANCES_AVG True  
FLOORSMAX_AVG True  
FLOORSMIN_AVG True  
LANDAREA_AVG True  
LIVINGAPARTMENTS_AVG True  
LIVINGAREA_AVG True  
NONLIVINGAPARTMENTS_AVG True  
NONLIVINGAREA_AVG True  
APARTMENTS_MODE True  
BASEMENTAREA_MODE True  
YEARS_BEGINEXPLUATATION_MODE True  
YEARS_BUILD_MODE True  
COMMONAREA_MODE True  
ELEVATORS_MODE True  
ENTRANCES_MODE True  
FLOORSMAX_MODE True  
FLOORSMIN_MODE True  
LANDAREA_MODE True  
LIVINGAPARTMENTS_MODE True  
LIVINGAREA_MODE True  
NONLIVINGAPARTMENTS_MODE True  
NONLIVINGAREA_MODE True  
APARTMENTS_MEDI True  
BASEMENTAREA_MEDI True  
YEARS_BEGINEXPLUATATION_MEDI True  
YEARS_BUILD_MEDI True  
COMMONAREA_MEDI True  
ELEVATORS_MEDI True  
ENTRANCES_MEDI True  
FLOORSMAX_MEDI True  
FLOORSMIN_MEDI True  
LANDAREA_MEDI True  
LIVINGAPARTMENTS_MEDI True  
LIVINGAREA_MEDI True  
NONLIVINGAPARTMENTS_MEDI True  
NONLIVINGAREA_MEDI True  
TOTALAREA_MODE True  
OBS_30_CNT_SOCIAL_CIRCLE True
```

```
DEF_30_CNT_SOCIAL_CIRCLE True
OBS_60_CNT_SOCIAL_CIRCLE True
DAYS_LAST_PHONE_CHANGE True
AMT_REQ_CREDIT_BUREAU_MON True
AMT_REQ_CREDIT_BUREAU_QRT True
AMT_REQ_CREDIT_BUREAU_YEAR True
```

## Accessing Outliers

```
In [42]: from PreProcessing import grab_outliers

df = sns.load_dataset('titanic')
grab_outliers(df, 'age')
```

```
survived pclass sex age sibsp parch fare embarked class who adult
t_male deck embark_town alive alone
33 0 2 male 66.0 0 0 10.5000 S Second man
True NaN Southampton no True
54 0 1 male 65.0 0 1 61.9792 C First man
True B Cherbourg no False
96 0 1 male 71.0 0 0 34.6542 C First man
True A Cherbourg no True
116 0 3 male 70.5 0 0 7.7500 Q Third man
True NaN Queenstown no True
280 0 3 male 65.0 0 0 7.7500 Q Third man
True NaN Queenstown no True
```

```
In [43]: grab_outliers(df, "age", True)
```

```
survived pclass sex age sibsp parch fare embarked class who adult
t_male deck embark_town alive alone
33 0 2 male 66.0 0 0 10.5000 S Second man
True NaN Southampton no True
54 0 1 male 65.0 0 1 61.9792 C First man
True B Cherbourg no False
96 0 1 male 71.0 0 0 34.6542 C First man
True A Cherbourg no True
116 0 3 male 70.5 0 0 7.7500 Q Third man
True NaN Queenstown no True
280 0 3 male 65.0 0 0 7.7500 Q Third man
True NaN Queenstown no True
```

```
Out[43]: Int64Index([33, 54, 96, 116, 280, 456, 493, 630, 672, 745, 851], dtype='int64')
```

```
In [44]: age_index = grab_outliers(df, "age", True)
```

```
survived pclass sex age sibsp parch fare embarked class who adult
t_male deck embark_town alive alone
33 0 2 male 66.0 0 0 10.5000 S Second man
True NaN Southampton no True
54 0 1 male 65.0 0 1 61.9792 C First man
True B Cherbourg no False
96 0 1 male 71.0 0 0 34.6542 C First man
True A Cherbourg no True
116 0 3 male 70.5 0 0 7.7500 Q Third man
True NaN Queenstown no True
```

```
280      0      3 male  65.0      0      0    7.7500      Q   Third man
True   NaN  Queenstown    no   True
```

```
In [45]: outlier_thresholds(df, "age")
```

```
Out[45]: (-6.6875, 64.8125)
```

```
In [47]: check_outlier(df, "age")
```

```
Out[47]: True
```

```
In [48]: grab_outliers(df, "age", True)
```

```
survived  pclass  sex  age  sibsp  parch     fare embarked  class  who  adul
t_male deck embark_town alive alone
33          0        2 male  66.0      0      0  10.5000      S  Second man
True   NaN  Southampton    no   True
54          0        1 male  65.0      0      1  61.9792      C  First  man
True     B  Cherbourg    no  False
96          0        1 male  71.0      0      0  34.6542      C  First  man
True     A  Cherbourg    no   True
116         0        3 male  70.5      0      0    7.7500      Q  Third man
True   NaN  Queenstown    no   True
280         0        3 male  65.0      0      0    7.7500      Q  Third man
True   NaN  Queenstown    no   True
Out[48]: Int64Index([33, 54, 96, 116, 280, 456, 493, 630, 672, 745, 851], dtype='int64')
```

## Solving the Outlier Problem

### 1) Deletion

```
In [49]: cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

```
Observations: 891
Variables: 15
cat_cols: 9
num_cols: 2
cat_but_car: 0
num_but_cat: 0
```

```
In [50]: num_cols = [col for col in num_cols if col not in "PassengerId"]
```

```
In [51]: df.shape
```

```
Out[51]: (891, 15)
```

```
In [52]: from PreProcessing import remove_outlier
```

```
for col in num_cols:  
    new_df = remove_outlier(df, col)
```

In [53]:  
df.shape[0] - new\_df.shape[0]

Out[53]: 116

## 2) re-assignment with thresholds

In [54]:  
low, up = outlier\_thresholds(df, "fare")  
low, up

Out[54]: (-26.724, 65.6344)

In [55]:  
df = sns.load\_dataset('titanic')  
df.shape

Out[55]: (891, 15)

In [56]:  
cat\_cols, num\_cols, cat\_but\_car = grab\_col\_names(df)

Observations: 891  
Variables: 15  
cat\_cols: 9  
num\_cols: 2  
cat\_but\_car: 0  
num\_but\_cat: 0

In [57]:  
num\_cols = [col for col in num\_cols if col not in "PassengerId"]

In [60]:  
from PreProcessing import replace\_with\_thresholds

```
for col in num_cols:  
    print(col, check_outlier(df, col))
```

age True  
fare True

In [61]:  
for col in num\_cols:  
 replace\_with\_thresholds(df, col)

In [62]:  
for col in num\_cols:  
 print(col, check\_outlier(df, col))

age False  
fare False

# Solving the Missing Value Problem

```
In [63]: df = sns.load_dataset('titanic')
df.shape
```

```
Out[63]: (891, 15)
```

## 1) Deletion

```
In [64]: df.dropna().shape
```

```
Out[64]: (182, 15)
```

## 2) Filling with Simple Assignment Methods

```
In [65]: df = sns.load_dataset('titanic')
df.isnull().sum()
```

```
Out[65]: survived      0
pclass          0
sex            0
age           177
sibsp          0
parch          0
fare            0
embarked       2
class          0
who            0
adult_male     0
deck          688
embark_town    2
alive          0
alone          0
dtype: int64
```

```
In [67]: df['age'] = df["age"].fillna(df["age"].mean())
df.isnull().sum()
```

```
Out[67]: survived      0
pclass          0
sex            0
age            0
sibsp          0
parch          0
fare            0
embarked       2
class          0
who            0
adult_male     0
deck          688
embark_town    2
alive          0
alone          0
dtype: int64
```

```
In [68]:
```

```
df = sns.load_dataset('titanic')
df.isnull().sum()
```

```
Out[68]: survived      0
         pclass        0
         sex          0
         age         177
         sibsp        0
         parch        0
         fare          0
         embarked      2
         class         0
         who          0
         adult_male    0
         deck        688
         embark_town    2
         alive         0
         alone         0
         dtype: int64
```

```
df["age"].fillna(df["age"].median())
```

```
Out[69]: 0      22.0
         1      38.0
         2      26.0
         3      35.0
         4      35.0
         ...
         886     27.0
         887     19.0
         888     28.0
         889     26.0
         890     32.0
Name: age, Length: 891, dtype: float64
```

```
df = sns.load_dataset('titanic')

cat_cols, num_cols, cat_but_car = grab_col_names(df)
num_cols = [col for col in num_cols if col not in "PassengerId"]
```

```
Observations: 891
Variables: 15
cat_cols: 9
num_cols: 2
cat_but_car: 0
num_but_cat: 0
```

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

```
Out[71]: survived      0
         pclass        0
         sex          0
         age         177
         sibsp        0
         parch        0
         fare          0
         embarked      2
         class         0
```

```
who          0
adult_male   0
deck        688
embark_town  2
alive         0
alone         0
dtype: int64
```

```
In [72]: df.loc[:, num_cols] = df[num_cols].apply(lambda x: x.fillna(x.mean()), axis=0)
df.isnull().sum()
```

```
Out[72]: survived      0
pclass        0
sex           0
age           0
sibsp         0
parch         0
fare           0
embarked      2
class          0
who            0
adult_male    0
deck        688
embark_town   2
alive          0
alone          0
dtype: int64
```

```
In [73]: df["embarked"].isnull().sum()
```

```
Out[73]: 2
```

```
In [74]: df['embarked'] = df["embarked"].fillna(df["embarked"].mode()[0])
```

```
In [75]: df["embarked"].isnull().sum()
```

```
Out[75]: 0
```

## Examining the Relationship of Missing Values with the Dependent Variable

```
In [76]: from PreProcessing import missing_values_table

df = sns.load_dataset('titanic')

missing_values_table(df, True)
```

	n_miss	ratio
deck	688	77.22
age	177	19.87

```
embarked          2    0.22
embark_town      2    0.22
Out[76]: ['age', 'embarked', 'deck', 'embark_town']
```

```
In [77]: na_cols = missing_values_table(df, True)
```

	n_miss	ratio
deck	688	77.22
age	177	19.87
embarked	2	0.22
embark_town	2	0.22

```
In [78]: from PreProcessing import missing_vs_target

missing_vs_target(df, "survived", na_cols)
```

	TARGET_MEAN	Count
age_NA_FLAG		
0	0.406162	714
1	0.293785	177

	TARGET_MEAN	Count
embarked_NA_FLAG		
0	0.382452	889
1	1.000000	2

	TARGET_MEAN	Count
deck_NA_FLAG		
0	0.669951	203
1	0.299419	688

	TARGET_MEAN	Count
embark_town_NA_FLAG		
0	0.382452	889
1	1.000000	2

## Label Encoding & Binary Encoding

```
In [79]: df = sns.load_dataset('titanic')

df["sex"].head()
```

```
Out[79]: 0     male
1   female
2   female
3   female
4     male
Name: sex, dtype: object
```

```
In [80]:
```

```

from PreProcessing import label_encoder

binary_cols = [col for col in df.columns if df[col].dtype not in [int, float]
              and df[col].nunique() == 2]

for col in binary_cols:
    label_encoder(df, col)

df.sex.head()

```

Out[80]:

0	1
1	0
2	0
3	0
4	1

Name: sex, dtype: int32

## One-Hot Encoding

In [81]:

```

from PreProcessing import one_hot_encoder

df = sns.load_dataset('titanic')
df["embarked"].value_counts()

```

Out[81]:

S	644
C	168
Q	77

Name: embarked, dtype: int64

In [82]:

```

ohe_cols = [col for col in df.columns if 10 >= df[col].nunique() > 2]
ohe_cols

```

Out[82]:

'pclass'	'sibsp'	'parch'	'embarked'	'class'	'who'	'deck'	'embark_town'
----------	---------	---------	------------	---------	-------	--------	---------------

In [83]:

```
one_hot_encoder(df, ohe_cols).head()
```

Out[83]:

	survived	sex	age	fare	adult_male	alive	alone	pclass_2	pclass_3	sibsp_1	sibsp_2	si
0	0	male	22.0	7.2500	True	no	False	0	1	1	0	
1	1	female	38.0	71.2833	False	yes	False	0	0	1	0	
2	1	female	26.0	7.9250	False	yes	True	0	1	0	0	
3	1	female	35.0	53.1000	False	yes	False	0	0	1	0	
4	0	male	35.0	8.0500	True	no	True	0	1	0	0	

## Rare Encoding

In [84]:

```
from PreProcessing import rare_encoder, rare_analyser
```

```
df = pd.read_csv('application_train.csv')
df["NAME_EDUCATION_TYPE"].value_counts()
```

```
Out[84]: Secondary / secondary special    218391
Higher education                      74863
Incomplete higher                       10277
Lower secondary                           3816
Academic degree                            164
Name: NAME_EDUCATION_TYPE, dtype: int64
```

```
In [85]: cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

```
Observations: 307511
Variables: 122
cat_cols: 19
num_cols: 61
cat_but_car: 1
num_but_cat: 4
```

```
In [86]: for col in cat_cols:
    cat_summary(df, col)
```

	NAME_CONTRACT_TYPE	Ratio
Cash loans	278232	90.478715
Revolving loans	29279	9.521285
##### CODE_GENDER	Ratio	
F	202448	65.834393
M	105059	34.164306
XNA	4	0.001301
##### FLAG_OWN_CAR	Ratio	
N	202924	65.989184
Y	104587	34.010816
##### FLAG_OWN_REALTY	Ratio	
Y	213312	69.367275
N	94199	30.632725
##### NAME_TYPE_SUITE	Ratio	
Unaccompanied	248526	80.818572
Family	40149	13.056118
Spouse, partner	11370	3.697429
Children	3267	1.062401
Other_B	1770	0.575589
Other_A	866	0.281616
Group of people	271	0.088127
##### NAME_INCOME_TYPE	Ratio	
Working	158774	51.631974
Commercial associate	71617	23.289248
Pensioner	55362	18.003258
State servant	21703	7.057634
Unemployed	22	0.007154
Student	18	0.005853
Businessman	10	0.003252
Maternity leave	5	0.001626

#		
	NAME_EDUCATION_TYPE	Ratio
Secondary / secondary special	218391	71.018923
Higher education	74863	24.344820
Incomplete higher	10277	3.341994
Lower secondary	3816	1.240931
Academic degree	164	0.053331
#		
	NAME_FAMILY_STATUS	Ratio
Married	196432	63.878040
Single / not married	45444	14.778008
Civil marriage	29775	9.682580
Separated	19770	6.429038
Widow	16088	5.231683
Unknown	2	0.000650
#		
	NAME_HOUSING_TYPE	Ratio
House / apartment	272868	88.734387
With parents	14840	4.825844
Municipal apartment	11183	3.636618
Rented apartment	4881	1.587260
Office apartment	2617	0.851026
Co-op apartment	1122	0.364865
#		
	OCCUPATION_TYPE	Ratio
Laborers	55186	17.946025
Sales staff	32102	10.439301
Core staff	27570	8.965533
Managers	21371	6.949670
Drivers	18603	6.049540
High skill tech staff	11380	3.700681
Accountants	9813	3.191105
Medicine staff	8537	2.776161
Security staff	6721	2.185613
Cooking staff	5946	1.933589
Cleaning staff	4653	1.513117
Private service staff	2652	0.862408
Low-skill Laborers	2093	0.680626
Waiters/barmen staff	1348	0.438358
Secretaries	1305	0.424375
Realty agents	751	0.244219
HR staff	563	0.183083
IT staff	526	0.171051
#		
	WEEKDAY_APPR_PROCESS_START	Ratio
TUESDAY	53901	17.528153
WEDNESDAY	51934	16.888502
MONDAY	50714	16.491768
THURSDAY	50591	16.451769
FRIDAY	50338	16.369496
SATURDAY	33852	11.008387
SUNDAY	16181	5.261926
#		
	FONDKAPREMONT_MODE	Ratio
reg oper account	73830	24.008897
reg oper spec account	12080	3.928315
not specified	5687	1.849365
org spec account	5619	1.827252
#		
	HOUSETYPE_MODE	Ratio

block of flats	150503	48.942314
specific housing	1499	0.487462
terraced house	1212	0.394132
#####	#####	#####
	WALLSMATERIAL_MODE	Ratio
Panel	66040	21.475655
Stone, brick	64815	21.077295
Block	9253	3.008998
Wooden	5362	1.743677
Mixed	2296	0.746640
Monolithic	1779	0.578516
Others	1625	0.528436
#####	#####	#####
	EMERGENCYSTATE_MODE	Ratio
No	159428	51.844649
Yes	2328	0.757046
#####	#####	#####
	DEF_60_CNT_SOCIAL_CIRCLE	Ratio
0.0	280721	91.288117
1.0	21841	7.102510
2.0	3170	1.030857
3.0	598	0.194465
4.0	135	0.043901
5.0	20	0.006504
6.0	3	0.000976
7.0	1	0.000325
24.0	1	0.000325
#####	#####	#####
	AMT_REQ_CREDIT_BUREAU_HOUR	Ratio
0.0	264366	85.969608
1.0	1560	0.507299
2.0	56	0.018211
3.0	9	0.002927
4.0	1	0.000325
#####	#####	#####
	AMT_REQ_CREDIT_BUREAU_DAY	Ratio
0.0	264503	86.014159
1.0	1292	0.420148
2.0	106	0.034470
3.0	45	0.014634
4.0	26	0.008455
5.0	9	0.002927
6.0	8	0.002602
9.0	2	0.000650
8.0	1	0.000325
#####	#####	#####
	AMT_REQ_CREDIT_BUREAU_WEEK	Ratio
0.0	257456	83.722534
1.0	8208	2.669173
2.0	199	0.064713
3.0	58	0.018861
4.0	34	0.011057
6.0	20	0.006504
5.0	10	0.003252
8.0	5	0.001626
7.0	2	0.000650
#####	#####	#####

In [87]: `df[ "NAME_INCOME_TYPE" ].value_counts()`

```
Out[87]: Working          158774
          Commercial associate   71617
          Pensioner            55362
          State servant         21703
          Unemployed             22
          Student                18
          Businessman             10
          Maternity leave           5
          Name: NAME_INCOME_TYPE, dtype: int64
```

```
In [88]: df.groupby("NAME_INCOME_TYPE")["TARGET"].mean()
```

```
Out[88]: NAME_INCOME_TYPE
          Businessman        0.000000
          Commercial associate 0.074843
          Maternity leave      0.400000
          Pensioner            0.053864
          State servant         0.057550
          Student                0.000000
          Unemployed            0.363636
          Working                0.095885
          Name: TARGET, dtype: float64
```

```
In [89]: rare_analyser(df, "TARGET", cat_cols)
```

```
NAME_CONTRACT_TYPE : 2
                    COUNT      RATIO  TARGET_MEAN
Cash loans       278232  0.904787  0.083459
Revolving loans  29279   0.095213  0.054783
```

```
CODE_GENDER : 3
              COUNT      RATIO  TARGET_MEAN
F      202448  0.658344  0.069993
M      105059  0.341643  0.101419
XNA        4   0.000013  0.000000
```

```
FLAG_OWN_CAR : 2
                COUNT      RATIO  TARGET_MEAN
N     202924  0.659892  0.085002
Y     104587  0.340108  0.072437
```

```
FLAG_OWN_REALTY : 2
                  COUNT      RATIO  TARGET_MEAN
N     94199   0.306327  0.083249
Y    213312   0.693673  0.079616
```

```
NAME_TYPE_SUITE : 7
                  COUNT      RATIO  TARGET_MEAN
Children        3267   0.010624  0.073768
Family          40149   0.130561  0.074946
Group of people  271    0.000881  0.084871
Other_A          866    0.002816  0.087760
Other_B          1770   0.005756  0.098305
Spouse, partner 11370   0.036974  0.078716
```

Unaccompanied 248526 0.808186 0.081830

NAME\_INCOME\_TYPE : 8

	COUNT	RATIO	TARGET_MEAN
Businessman	10	0.000033	0.000000
Commercial associate	71617	0.232892	0.074843
Maternity leave	5	0.000016	0.400000
Pensioner	55362	0.180033	0.053864
State servant	21703	0.070576	0.057550
Student	18	0.000059	0.000000
Unemployed	22	0.000072	0.363636
Working	158774	0.516320	0.095885

NAME\_EDUCATION\_TYPE : 5

	COUNT	RATIO	TARGET_MEAN
Academic degree	164	0.000533	0.018293
Higher education	74863	0.243448	0.053551
Incomplete higher	10277	0.033420	0.084850
Lower secondary	3816	0.012409	0.109277
Secondary / secondary special	218391	0.710189	0.089399

NAME\_FAMILY\_STATUS : 6

	COUNT	RATIO	TARGET_MEAN
Civil marriage	29775	0.096826	0.099446
Married	196432	0.638780	0.075599
Separated	19770	0.064290	0.081942
Single / not married	45444	0.147780	0.098077
Unknown	2	0.000007	0.000000
Widow	16088	0.052317	0.058242

NAME\_HOUSING\_TYPE : 6

	COUNT	RATIO	TARGET_MEAN
Co-op apartment	1122	0.003649	0.079323
House / apartment	272868	0.887344	0.077957
Municipal apartment	11183	0.036366	0.085397
Office apartment	2617	0.008510	0.065724
Rented apartment	4881	0.015873	0.123131
With parents	14840	0.048258	0.116981

OCCUPATION\_TYPE : 18

	COUNT	RATIO	TARGET_MEAN
Accountants	9813	0.031911	0.048303
Cleaning staff	4653	0.015131	0.096067
Cooking staff	5946	0.019336	0.104440
Core staff	27570	0.089655	0.063040
Drivers	18603	0.060495	0.113261
HR staff	563	0.001831	0.063943
High skill tech staff	11380	0.037007	0.061599
IT staff	526	0.001711	0.064639
Laborers	55186	0.179460	0.105788
Low-skill Laborers	2093	0.006806	0.171524
Managers	21371	0.069497	0.062140
Medicine staff	8537	0.027762	0.067002
Private service staff	2652	0.008624	0.065988
Realty agents	751	0.002442	0.078562

Sales staff	32102	0.104393	0.096318
Secretaries	1305	0.004244	0.070498
Security staff	6721	0.021856	0.107424
Waiters/barmen staff	1348	0.004384	0.112760

WEEKDAY\_APPR\_PROCESS\_START : 7

	COUNT	RATIO	TARGET_MEAN
FRIDAY	50338	0.163695	0.081469
MONDAY	50714	0.164918	0.077572
SATURDAY	33852	0.110084	0.078873
SUNDAY	16181	0.052619	0.079291
THURSDAY	50591	0.164518	0.081003
TUESDAY	53901	0.175282	0.083505
WEDNESDAY	51934	0.168885	0.081604

FONDKAPREMONT\_MODE : 4

	COUNT	RATIO	TARGET_MEAN
not specified	5687	0.018494	0.075435
org spec account	5619	0.018273	0.058195
reg oper account	73830	0.240089	0.069782
reg oper spec account	12080	0.039283	0.065563

HOUSETYPE\_MODE : 3

	COUNT	RATIO	TARGET_MEAN
block of flats	150503	0.489423	0.069434
specific housing	1499	0.004875	0.101401
terraced house	1212	0.003941	0.084983

WALLSMATERIAL\_MODE : 7

	COUNT	RATIO	TARGET_MEAN
Block	9253	0.030090	0.070247
Mixed	2296	0.007466	0.075348
Monolithic	1779	0.005785	0.047218
Others	1625	0.005284	0.083077
Panel	66040	0.214757	0.063477
Stone, brick	64815	0.210773	0.074057
Wooden	5362	0.017437	0.096979

EMERGENCYSTATE\_MODE : 2

	COUNT	RATIO	TARGET_MEAN
No	159428	0.518446	0.069649
Yes	2328	0.007570	0.095790

DEF\_60\_CNT\_SOCIAL\_CIRCLE : 9

	COUNT	RATIO	TARGET_MEAN
0.0	280721	0.912881	0.078348
1.0	21841	0.071025	0.105169
2.0	3170	0.010309	0.121451
3.0	598	0.001945	0.158863
4.0	135	0.000439	0.111111
5.0	20	0.000065	0.150000
6.0	3	0.000010	0.000000
7.0	1	0.000003	0.000000
24.0	1	0.000003	0.000000

```
AMT_REQ_CREDIT_BUREAU_HOUR : 5
    COUNT      RATIO  TARGET_MEAN
0.0  264366  0.859696   0.077173
1.0   1560   0.005073   0.080128
2.0     56   0.000182   0.107143
3.0     9   0.000029   0.000000
4.0     1   0.000003   0.000000
```

```
AMT_REQ_CREDIT_BUREAU_DAY : 9
    COUNT      RATIO  TARGET_MEAN
0.0  264503  0.860142   0.077096
1.0   1292   0.004201   0.096749
2.0    106   0.000345   0.103774
3.0     45   0.000146   0.044444
4.0     26   0.000085   0.115385
5.0     9   0.000029   0.000000
6.0     8   0.000026   0.000000
8.0     1   0.000003   0.000000
9.0     2   0.000007   0.000000
```

```
AMT_REQ_CREDIT_BUREAU_WEEK : 9
    COUNT      RATIO  TARGET_MEAN
0.0  257456  0.837225   0.077159
1.0   8208   0.026692   0.077729
2.0    199   0.000647   0.100503
3.0     58   0.000189   0.068966
4.0     34   0.000111   0.117647
5.0     10   0.000033   0.100000
6.0     20   0.000065   0.050000
7.0     2   0.000007   0.000000
8.0     5   0.000016   0.000000
```

```
In [90]: new_df = rare_encoder(df, 0.01)
```

```
In [91]: rare_analyser(new_df, "TARGET", cat_cols )
```

```
NAME_CONTRACT_TYPE : 2
    COUNT      RATIO  TARGET_MEAN
Cash loans       278232  0.904787   0.083459
Revolving loans 29279   0.095213   0.054783
```

```
CODE_GENDER : 3
    COUNT      RATIO  TARGET_MEAN
F      202448  0.658344   0.069993
M      105059  0.341643   0.101419
Rare        4   0.000013   0.000000
```

```
FLAG_OWN_CAR : 2
    COUNT      RATIO  TARGET_MEAN
N      202924  0.659892   0.085002
```

Y 104587 0.340108 0.072437

FLAG\_OWN\_REALTY : 2  
COUNT RATIO TARGET\_MEAN  
N 94199 0.306327 0.083249  
Y 213312 0.693673 0.079616

NAME\_TYPE\_SUITE : 5  
COUNT RATIO TARGET\_MEAN  
Children 3267 0.010624 0.073768  
Family 40149 0.130561 0.074946  
Rare 2907 0.009453 0.093911  
Spouse, partner 11370 0.036974 0.078716  
Unaccompanied 248526 0.808186 0.081830

NAME\_INCOME\_TYPE : 5  
COUNT RATIO TARGET\_MEAN  
Commercial associate 71617 0.232892 0.074843  
Pensioner 55362 0.180033 0.053864  
Rare 55 0.000179 0.181818  
State servant 21703 0.070576 0.057550  
Working 158774 0.516320 0.095885

NAME\_EDUCATION\_TYPE : 5  
COUNT RATIO TARGET\_MEAN  
Higher education 74863 0.243448 0.053551  
Incomplete higher 10277 0.033420 0.084850  
Lower secondary 3816 0.012409 0.109277  
Rare 164 0.000533 0.018293  
Secondary / secondary special 218391 0.710189 0.089399

NAME\_FAMILY\_STATUS : 6  
COUNT RATIO TARGET\_MEAN  
Civil marriage 29775 0.096826 0.099446  
Married 196432 0.638780 0.075599  
Rare 2 0.000007 0.000000  
Separated 19770 0.064290 0.081942  
Single / not married 45444 0.147780 0.098077  
Widow 16088 0.052317 0.058242

NAME\_HOUSING\_TYPE : 5  
COUNT RATIO TARGET\_MEAN  
House / apartment 272868 0.887344 0.077957  
Municipal apartment 11183 0.036366 0.085397  
Rare 3739 0.012159 0.069805  
Rented apartment 4881 0.015873 0.123131  
With parents 14840 0.048258 0.116981

OCCUPATION\_TYPE : 12  
COUNT RATIO TARGET\_MEAN  
Accountants 9813 0.031911 0.048303  
Cleaning staff 4653 0.015131 0.096067  
Cooking staff 5946 0.019336 0.104440

Core staff	27570	0.089655	0.063040
Drivers	18603	0.060495	0.113261
High skill tech staff	11380	0.037007	0.061599
Laborers	55186	0.179460	0.105788
Managers	21371	0.069497	0.062140
Medicine staff	8537	0.027762	0.067002
Rare	9238	0.030041	0.098181
Sales staff	32102	0.104393	0.096318
Security staff	6721	0.021856	0.107424

WEEKDAY\_APPR\_PROCESS\_START : 7

	COUNT	RATIO	TARGET_MEAN
FRIDAY	50338	0.163695	0.081469
MONDAY	50714	0.164918	0.077572
SATURDAY	33852	0.110084	0.078873
SUNDAY	16181	0.052619	0.079291
THURSDAY	50591	0.164518	0.081003
TUESDAY	53901	0.175282	0.083505
WEDNESDAY	51934	0.168885	0.081604

FONDKAPREMONT\_MODE : 4

	COUNT	RATIO	TARGET_MEAN
not specified	5687	0.018494	0.075435
org spec account	5619	0.018273	0.058195
reg oper account	73830	0.240089	0.069782
reg oper spec account	12080	0.039283	0.065563

HOUSETYPE\_MODE : 2

	COUNT	RATIO	TARGET_MEAN
Rare	2711	0.008816	0.094061
block of flats	150503	0.489423	0.069434

WALLSMATERIAL\_MODE : 5

	COUNT	RATIO	TARGET_MEAN
Block	9253	0.030090	0.070247
Panel	66040	0.214757	0.063477
Rare	5700	0.018536	0.068772
Stone, brick	64815	0.210773	0.074057
Wooden	5362	0.017437	0.096979

EMERGENCYSTATE\_MODE : 2

	COUNT	RATIO	TARGET_MEAN
No	159428	0.518446	0.069649
Rare	2328	0.007570	0.095790

DEF\_60\_CNT\_SOCIAL\_CIRCLE : 9

	COUNT	RATIO	TARGET_MEAN
0.0	280721	0.912881	0.078348
1.0	21841	0.071025	0.105169
2.0	3170	0.010309	0.121451
3.0	598	0.001945	0.158863
4.0	135	0.000439	0.111111
5.0	20	0.000065	0.150000
6.0	3	0.000010	0.000000

7.0	1	0.000003	0.000000
24.0	1	0.000003	0.000000

AMT\_REQ\_CREDIT\_BUREAU\_HOUR : 5

	COUNT	RATIO	TARGET_MEAN
0.0	264366	0.859696	0.077173
1.0	1560	0.005073	0.080128
2.0	56	0.000182	0.107143
3.0	9	0.000029	0.000000
4.0	1	0.000003	0.000000

AMT\_REQ\_CREDIT\_BUREAU\_DAY : 9

	COUNT	RATIO	TARGET_MEAN
0.0	264503	0.860142	0.077096
1.0	1292	0.004201	0.096749
2.0	106	0.000345	0.103774
3.0	45	0.000146	0.044444
4.0	26	0.000085	0.115385
5.0	9	0.000029	0.000000
6.0	8	0.000026	0.000000
8.0	1	0.000003	0.000000
9.0	2	0.000007	0.000000

AMT\_REQ\_CREDIT\_BUREAU\_WEEK : 9

	COUNT	RATIO	TARGET_MEAN
0.0	257456	0.837225	0.077159
1.0	8208	0.026692	0.077729
2.0	199	0.000647	0.100503
3.0	58	0.000189	0.068966
4.0	34	0.000111	0.117647
5.0	10	0.000033	0.100000
6.0	20	0.000065	0.050000
7.0	2	0.000007	0.000000
8.0	5	0.000016	0.000000