

```
import numpy as np
import pandas as pd
df=pd.read_csv("Hotel_Dataset.csv")
print(df)
```

```
↗
  CustomerID  Age_Group  Rating(1-5)  Hotel  FoodPreference  Bill \
0           1         20-25          4    Ibis             veg   1300
1           2         30-35          5  LemonTree        Non-Veg   2000
2           3         25-30          6   RedFox           Veg    1322
3           4         20-25         -1  LemonTree           Veg   1234
4           5          35+          3    Ibis      Vegetarian    989
5           6          35+          3   Ibys        Non-Veg   1909
6           7          35+          4   RedFox      Vegetarian  1000
7           8         20-25          7  LemonTree           Veg   2999
8           9         25-30          2    Ibis        Non-Veg   3456
9           9         25-30          2    Ibis        Non-Veg   3456
10          10         30-35          5   RedFox        non-Veg -6755
```

```
      NoOfPax  EstimatedSalary  Age_Group.1
0           2           40000      20-25
1           3           59000      30-35
2           2           30000      25-30
3           2          120000      20-25
4           2           45000          35+
5           2          122220          35+
6          -1           21122          35+
7          -10          345673      20-25
8           3          -99999      25-30
9           3          -99999      25-30
10          4           87777      30-35
```

```
df.duplicated()
```

```
↗
  0
0  False
1  False
2  False
3  False
4  False
5  False
6  False
7  False
8  False
9   True
10 False
```

```
df.info()
```

```
↗
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   CustomerID            11 non-null    int64
1   Age_Group             11 non-null    object
2   Rating(1-5)           11 non-null    int64
3   Hotel                 11 non-null    object
4   FoodPreference        11 non-null    object
5   Bill                 11 non-null    int64
6   NoOfPax              11 non-null    int64
7   EstimatedSalary       11 non-null    int64
8   Age_Group.1           11 non-null    object
dtypes: int64(5), object(4)
memory usage: 920.0+ bytes
```

```
df.drop_duplicates(inplace=True)
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1	
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25	
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35	
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30	
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25	
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+	
5	6	35+	3	lbys	Non-Veg	1909	2	122220	35+	
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+	
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25	
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30	
10	10	30-35	5	RedFox	non-Vea	-6755	4	87777	30-35	

Next steps:

[Generate code with df](#)[View recommended plots](#)[New interactive sheet](#)

len(df)

10

index=np.array(list(range(0,len(df))))

df.set\_index(index,inplace=True)

index

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

print(df.columns)

```
Index(['CustomerID', 'Rating(1-5)', 'Hotel', 'FoodPreference', 'Bill',
      'NoOfPax', 'EstimatedSalary'],
      dtype='object')
```

import numpy as np

df.loc[df.CustomerID &lt; 0, 'CustomerID'] = np.nan

df.loc[df.Bill &lt; 0, 'Bill'] = np.nan

df.loc[df.EstimatedSalary &lt; 0, 'EstimatedSalary'] = np.nan


df

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
0	1.0	20-25	4	Ibis	veg	1300.0	2	40000.0	
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3	59000.0	
2	3.0	25-30	6	RedFox	Veg	1322.0	2	30000.0	
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2	120000.0	
4	5.0	35+	3	Ibis	Vegetarian	989.0	2	45000.0	
5	6.0	35+	3	lbys	Non-Veg	1909.0	2	122220.0	
6	7.0	35+	4	RedFox	Vegetarian	1000.0	-1	21122.0	
7	8.0	20-25	7	LemonTree	Veg	2999.0	-10	345673.0	
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3	NaN	
9	10.0	30-35	5	RedFox	non-Vea	NaN	4	87777.0	

Next steps:

[Generate code with df](#)[View recommended plots](#)[New interactive sheet](#)

```
df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
df
```

 <ipython-input-13-55d18ab59348>:1: FutureWarning: ChainedAssignmentError: behaviour will change in pandas 3.0!  
You are setting values through chained assignment. Currently this works in certain cases, but when using Copy-on-Write (which will be the default in pandas 3.0) this can fail to propagate the change to the underlying data. A typical example is when you are setting values in a column of a DataFrame, like:

```
df["col"][row_indexer] = value
```

Use `df.loc[row\_indexer, "col"] = values` instead, to perform the assignment in a single step and ensure this keeps updating the original DataFrame.

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)




```
df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
```

<ipython-input-13-55d18ab59348>:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
0	1.0	20-25	4	Ibis	veg	1300.0	2.0	40000.0	
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0	
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0	
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0	
4	5.0	35+	3	Ibis	Vegetarian	989.0	2.0	45000.0	
5	6.0	35+	3	Ibys	Non-Veg	1909.0	2.0	122220.0	
6	7.0	35+	4	RedFox	Vegetarian	1000.0	NaN	21122.0	
7	8.0	20-25	7	LemonTree	Veg	2999.0	NaN	345673.0	
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	NaN	
9	10.0	30-35	5	RedFox	non-Veg	NaN	4.0	87777.0	


Next steps:

[Generate code with df](#)


[View recommended plots](#)

[New interactive sheet](#)

```
df.Age_Group.unique()
```


 array(['20-25', '30-35', '25-30', '35+'], dtype=object)

```
df.Hotel.unique()
```

 array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'], dtype=object)

```
df.Hotel.replace(['Ibys'],'Ibis',inplace=True)
```

```
print(df.FoodPreference.unique)
```

 <bound method Series.unique of 0 Veg  
1 Non-Veg  
2 Veg  
3 Veg  
4 Veg  
5 Non-Veg  
6 Veg  
7 Veg  
8 Non-Veg  
9 Non-Veg  
Name: FoodPreference, dtype: object>


```
df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
```

```
df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)
```

```
df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)
```

```
df.Bill.fillna(round(df.Bill.mean()),inplace=True)
```

```
df
```

 <ipython-input-19-9197dedb9332>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
```

<ipython-input-19-9197dedb9332>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)
```

<ipython-input-19-9197dedb9332>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting



For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)
```

<ipython-input-19-9197dedb9332>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
df.Bill.fillna(round(df.Bill.mean()),inplace=True)
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
0	1.0	20-25	4	Ibis	Veg	1300.0	2.0	40000.0	
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0	
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0	
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0	
4	5.0	35+	3	Ibis	Veg	989.0	2.0	45000.0	
5	6.0	35+	3	Ibis	Non-Veg	1909.0	2.0	122220.0	
6	7.0	35+	4	RedFox	Veg	1000.0	2.0	21122.0	
7	8.0	20-25	7	LemonTree	Veg	2999.0	2.0	345673.0	
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	96755.0	
9	10.0	30-35	5	RedFox	Non-Veg	1801.0	4.0	87777.0	

Next steps:

[Generate code with df](#)
[View recommended plots](#)
[New interactive sheet](#)

```
df.FoodPreference.replace(['Vegetarian', 'veg'], 'Veg', inplace=True)
df.FoodPreference.replace(['non-Veg'], 'Non-Veg', inplace=True)
```