```
import pandas as pd
import numpy as np
from google.colab import files

uploaded = files.upload()

df = pd.read_csv("Hotel_Dataset.csv")
display(df)
```

Choose Files Hotel_Dataset.csv

Hotel_Dataset.csv(text/csv) - 574 bytes, last modified: 11/19/2025 - 100% done

```
Saving Hotel_Dataset.csv to Hotel_Dataset.csv
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfP
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	

Next steps: Generate code with df | New interactive sheet

```
# returns boolean Series marking duplicated rows (True = duplicate)
dupe_mask = df.duplicated()
display(dupe_mask)
```

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

dtype: bool

```
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: 924.0+ bytes
```

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

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPa
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	-1
8	9	25-30	2	Ibis	Non-Veg	3456	
10	10	30-35	5	RedFox	non-Veg	-6755	

Next steps: Generate code with df | New interactive sheet

```
n_rows = len(df)
print(n_rows)
```

```
10
```

```
df.reset_index(drop=True, inplace=True)
display(df)
```

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

Next steps: Generate code with df New interactive sheet

```
if 'Age_Group.1' in df.columns:
    df.drop(columns=['Age_Group.1'], inplace=True)
display(df)
```

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

Next steps: Generate code with df New interactive sheet

```
import numpy as np

# for each numeric column, set values < 0 to NaN using .loc
if 'CustomerID' in df.columns:
    df.loc[df['CustomerID'] < 0, 'CustomerID'] = np.nan

if 'Bill' in df.columns:
    df.loc[df['Bill'] < 0, 'Bill'] = np.nan

if 'EstimatedSalary' in df.columns:
    df.loc[df['EstimatedSalary'] < 0, 'EstimatedSalary'] = np.nan

display(df)
```

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

Next steps: Generate code with df New interactive sheet

```
if 'NoOfPax' in df.columns:
    df.loc[(df['NoOfPax'] < 1) | (df['NoOfPax'] > 20), 'NoOfPax'] = np.nan

display(df)
```

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

Next steps: Generate code with df New interactive sheet

```
if 'Age_Group' in df.columns:
    unique_ages = df['Age_Group'].unique()
    print(unique_ages)
```

```
['20-25' '30-35' '25-30' '35+']
```

```
if 'Hotel' in df.columns:
    unique_hotels = df['Hotel'].unique()
    print(unique_hotels)
```

```
['Ibis' 'LemonTree' 'RedFox' 'Ibys']
```

```
if 'Hotel' in df.columns:
    df['Hotel'].replace({'Ibys': 'Ibis'}, inplace=True)

display(df['Hotel'].unique())
```

```
/tmp/ipython-input-231423752.py:2: FutureWarning: A value is trying to be set
The behavior will change in pandas 3.0. This inplace method will never work b

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


  df['Hotel'].replace({'Ibys': 'Ibis'}, inplace=True)
array(['Ibis', 'LemonTree', 'RedFox'], dtype=object)
```

```
if 'FoodPreference' in df.columns:
    df['FoodPreference'].replace({'Vegetarian': 'Veg', 'veg': 'Veg', 'non
    # ensure capitalization consistency, e.g., 'Non-Veg'
display(df['FoodPreference'].value_counts())
```

```
/tmp/ipython-input-1545286410.py:2: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b

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


  df['FoodPreference'].replace({'Vegetarian': 'Veg', 'veg': 'Veg', 'non-Veg':
```

FoodPreference	count
Veg	6
Non-Veg	4

dtype: int64

```
if 'EstimatedSalary' in df.columns:
    df['EstimatedSalary'].fillna(round(df['EstimatedSalary'].mean()), inp

# Bill (continuous) -> mean
if 'Bill' in df.columns:
```

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

# NoOfPax (discrete) -> median
if 'NoOfPax' in df.columns:
    df['NoOfPax'].fillna(round(df['NoOfPax'].median()), inplace=True)

# Rating(1-5) (discrete) -> median
if 'Rating(1-5)' in df.columns:
    df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=T

# categorical columns -> mode
for col in ['Hotel','FoodPreference','Age_Group']:
    if col in df.columns:
        mode_val = df[col].mode(dropna=True)
        if len(mode_val) > 0:
            df[col].fillna(mode_val[0], inplace=True)

display(df)
```

```
/tmp/ipython-input-1120200388.py:2: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b

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


  df['EstimatedSalary'].fillna(round(df['EstimatedSalary'].mean()), inplace=T
/tmp/ipython-input-1120200388.py:6: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b
```