Download the dataset : https://media.geeksforgeeks.org/wp- content/uploads/20240319120216/housing.csv

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files housing.csv
• housing.csv(text/csv) - 1423529 bytes, last modified: 3/10/2025 - 100% done
Saving housing.csv to housing.csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats
```

```
import pandas as pd

df = pd.read_csv('housing.csv')
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocear
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	

Next steps: Generate code with df | View recommended plots | New interactive sheet

Perform the describe and info steps

```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_hou
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	2064
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	20685
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	1153
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	149
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	1196
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	1797
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	26472
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	5000

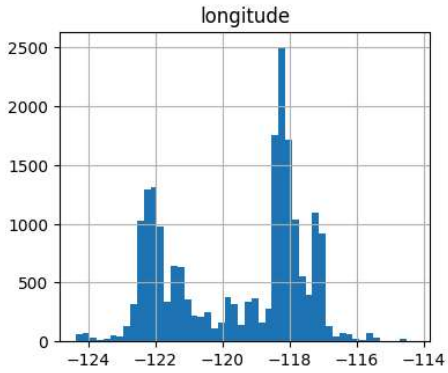
```
df.info()
```

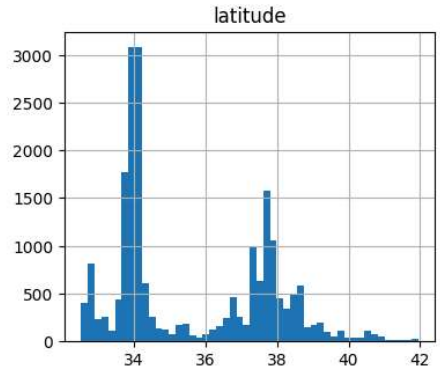
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
```

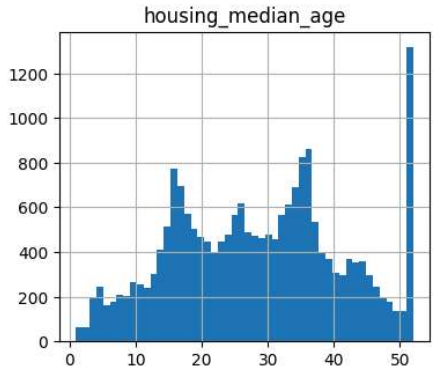
```
 9   ocean_proximity     20640 non-null  object
 10  income_cat          20640 non-null  category
dtypes: category(1), float64(9), object(1)
memory usage: 1.6+ MB
```

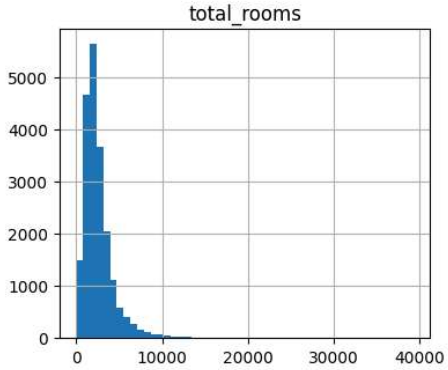
Plot the histogram of each feature(Indicate what does histogram indicate on median_income and house_median_age)

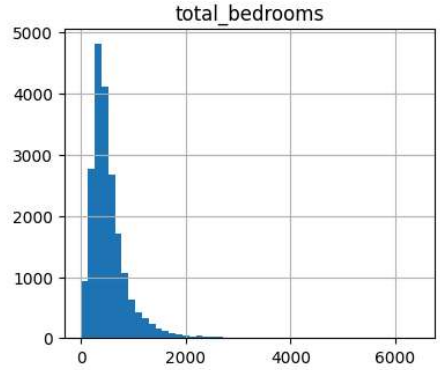
```
import matplotlib.pyplot as plt
df.hist(figsize=(12, 10), bins=50)
plt.tight_layout()

df['median_income'].hist(bins=50, color='skyblue')
plt.title('Distribution of Median Income')
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.show()
df['housing_median_age'].hist(bins=50, color='lightcoral')
plt.title('Distribution of Housing Median Age')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()
df['housing_median_age'].hist(bins=50, color='lightcoral')
plt.title('Distribution of Housing Median Age')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()

df['longitude'].hist(bins=50, color='lightcoral')
plt.title('Distribution of longitude')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()

df['total_rooms'].hist(bins=50, color='lightcoral')
plt.title('Distribution of total rooms')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()

df['total_bedrooms'].hist(bins=50, color='skyblue')
plt.title('Distribution of total bedrooms')
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.show()

df['population'].hist(bins=50, color='lightcoral')
plt.title('Distribution of population')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()

df['households'].hist(bins=50, color='lightcoral')
plt.title('Distribution of household')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()

df['median_house_value'].hist(bins=50, color='skyblue')
plt.title('Distribution of median house value')
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.show()
```

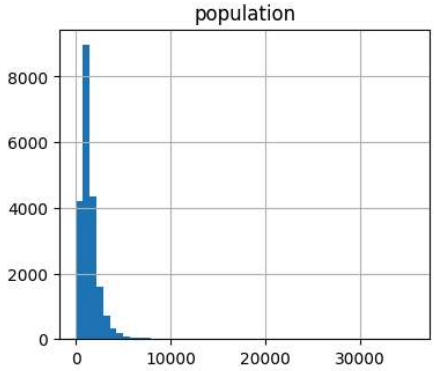

longitude

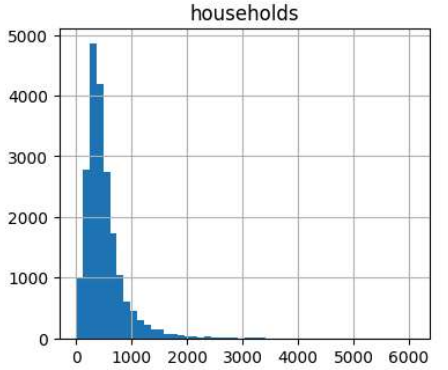

latitude

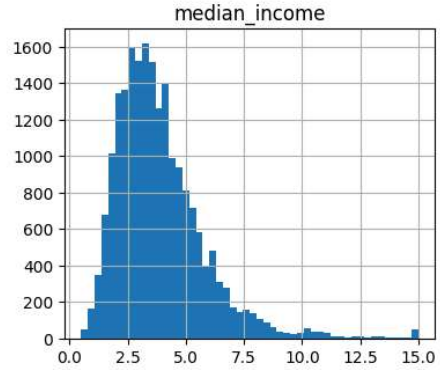

housing_median_age

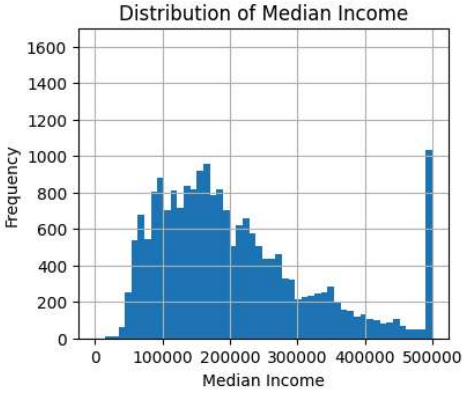

total_rooms

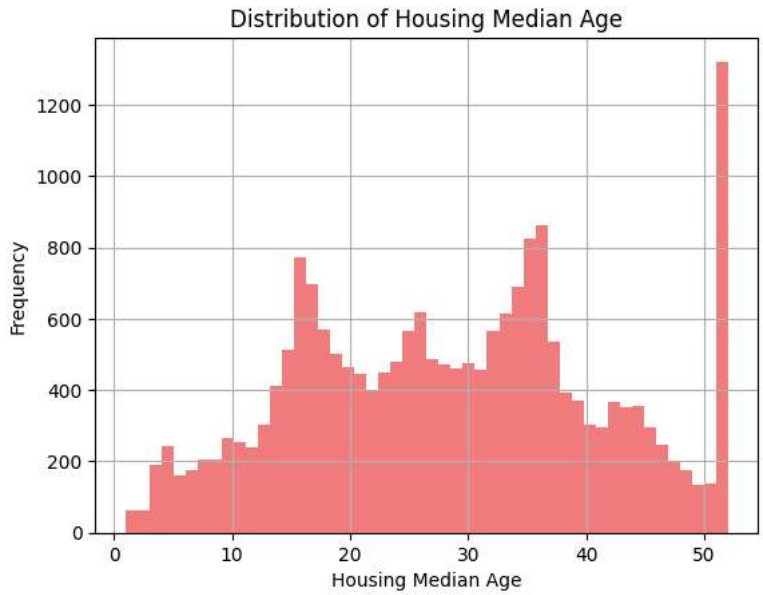

total_bedrooms

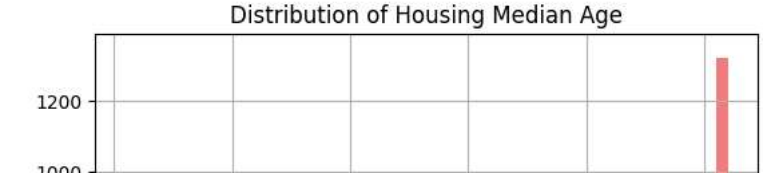

population

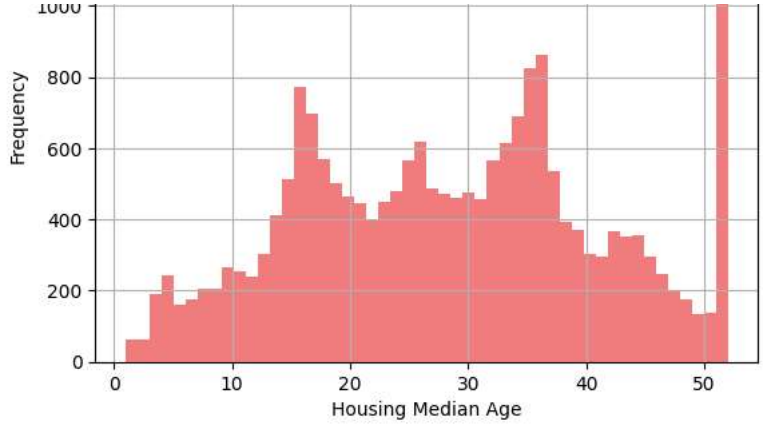

households

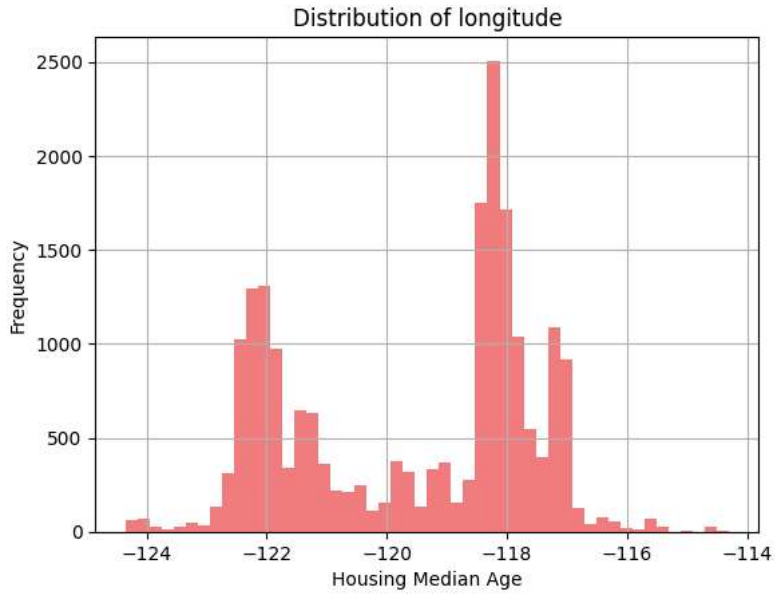

median_income

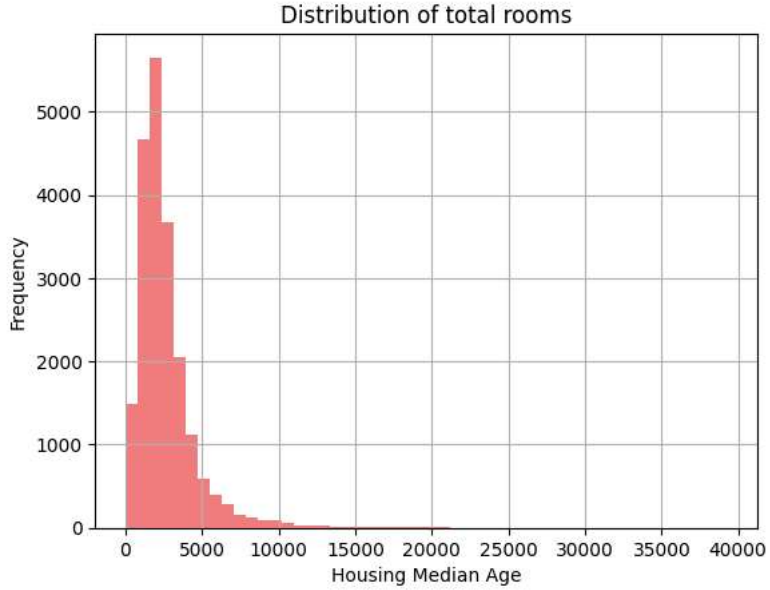

Distribution of Median Income
Frequency
Median Income


Distribution of Housing Median Age
Frequency
Housing Median Age

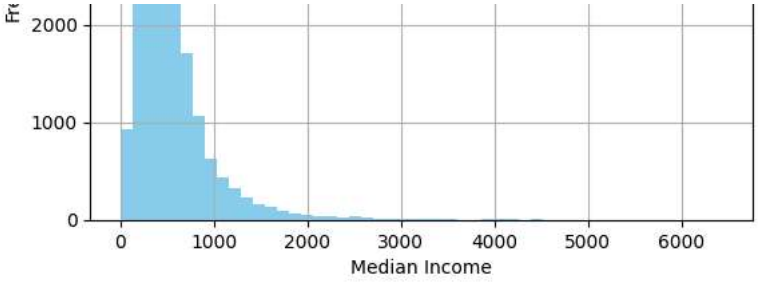

Distribution of Housing Median Age

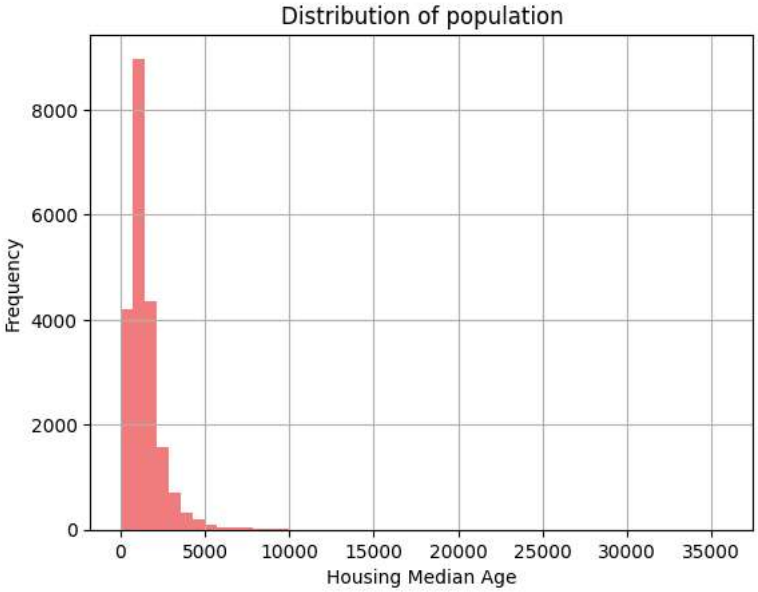

Frequency
Housing Median Age

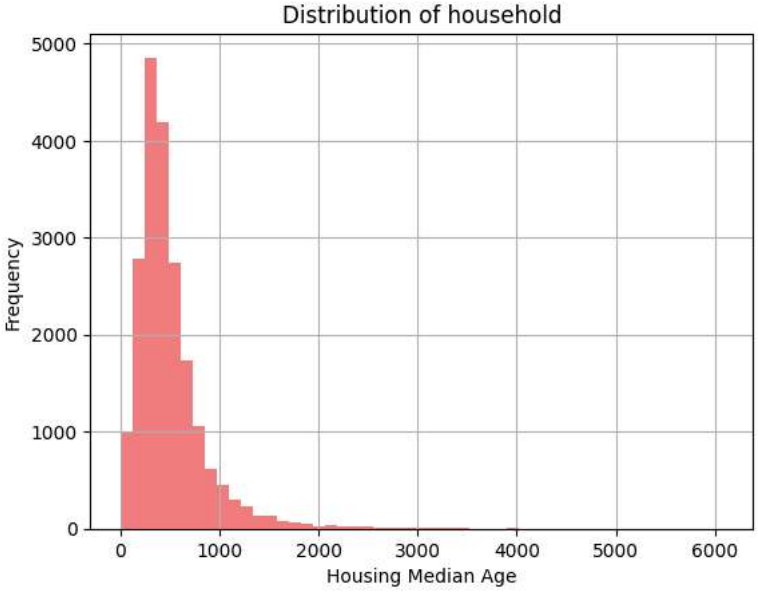

Distribution of longitude
Frequency
Housing Median Age

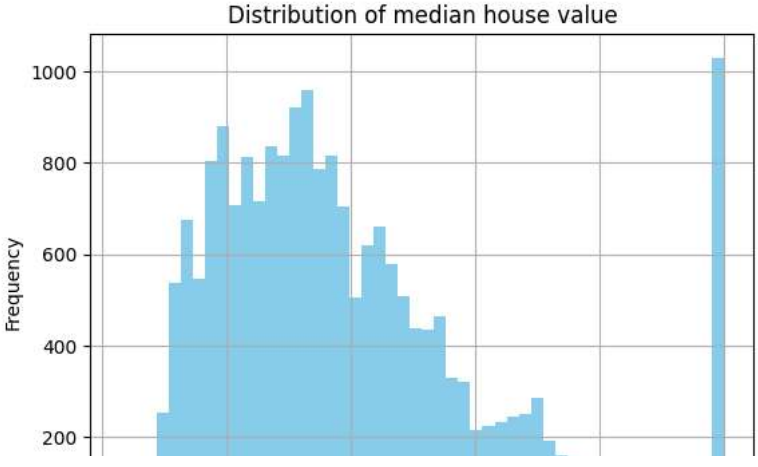

Distribution of total rooms
Frequency
Housing Median Age

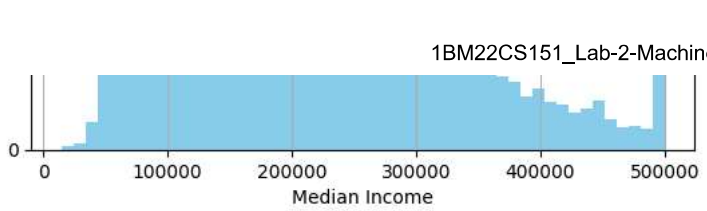

Distribution of total bedrooms


Fr
2000
1000
0
0
1000
2000
3000
4000
5000
6000
Median Income

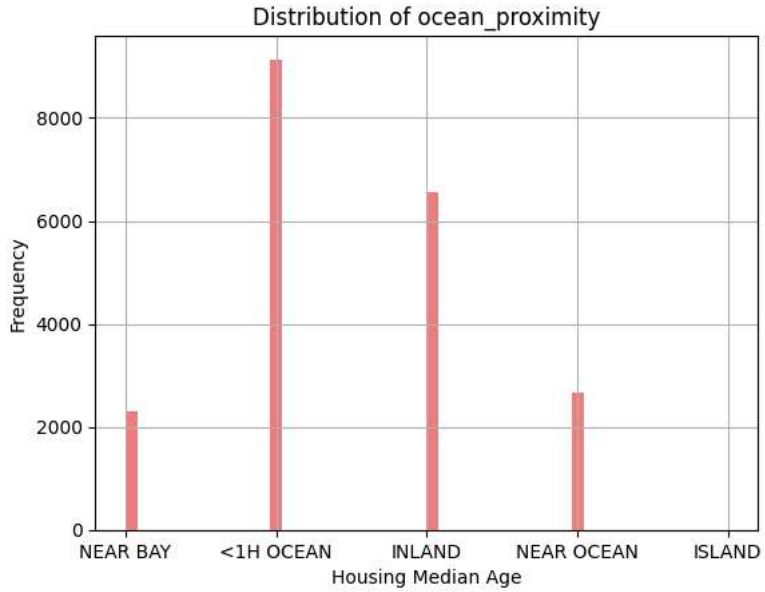

Distribution of population
8000
6000
4000
2000
0
Frequency
0
5000
10000
15000
20000
25000
30000
35000
Housing Median Age


Distribution of household
5000
4000
3000
2000
1000
0
Frequency
0
1000
2000
3000
4000
5000
6000
Housing Median Age

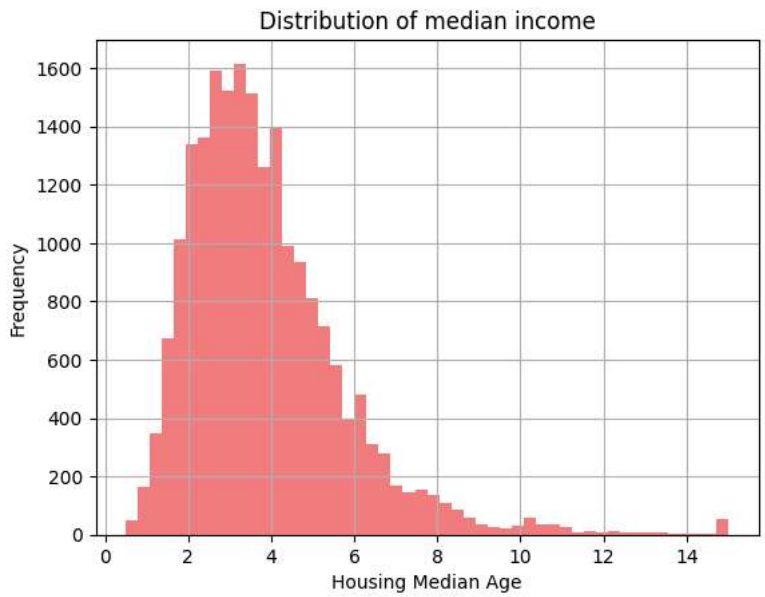

Distribution of median house value
1000
800
600
400
200
Frequency


0
0
100000
200000
300000
400000
500000
Median Income

```
df['ocean_proximity'].hist(bins=50, color='lightcoral')
plt.title('Distribution of ocean_proximity')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()
```




```
df['median_income'].hist(bins=50, color='lightcoral')
plt.title('Distribution of median income')
plt.xlabel('Housing Median Age')
plt.ylabel('Frequency')
plt.show()
```



Demonstrate the process of creating a test set(write the difference between random and stratified test set)

```
train_set_random, test_set_random = train_test_split(df, test_size=0.2, random_state=42)
df["income_cat"] = pd.cut(df["median_income"],
                          bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                          labels=[1, 2, 3, 4, 5])

train_set_strat, test_set_strat = train_test_split(df, test_size=0.2, stratify=df["income_cat"], random_state=42)
for set_ in (train_set_strat, test_set_strat):
    set_.drop("income_cat", axis=1, inplace=True)
df
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	c
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	
...	...	...	...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	

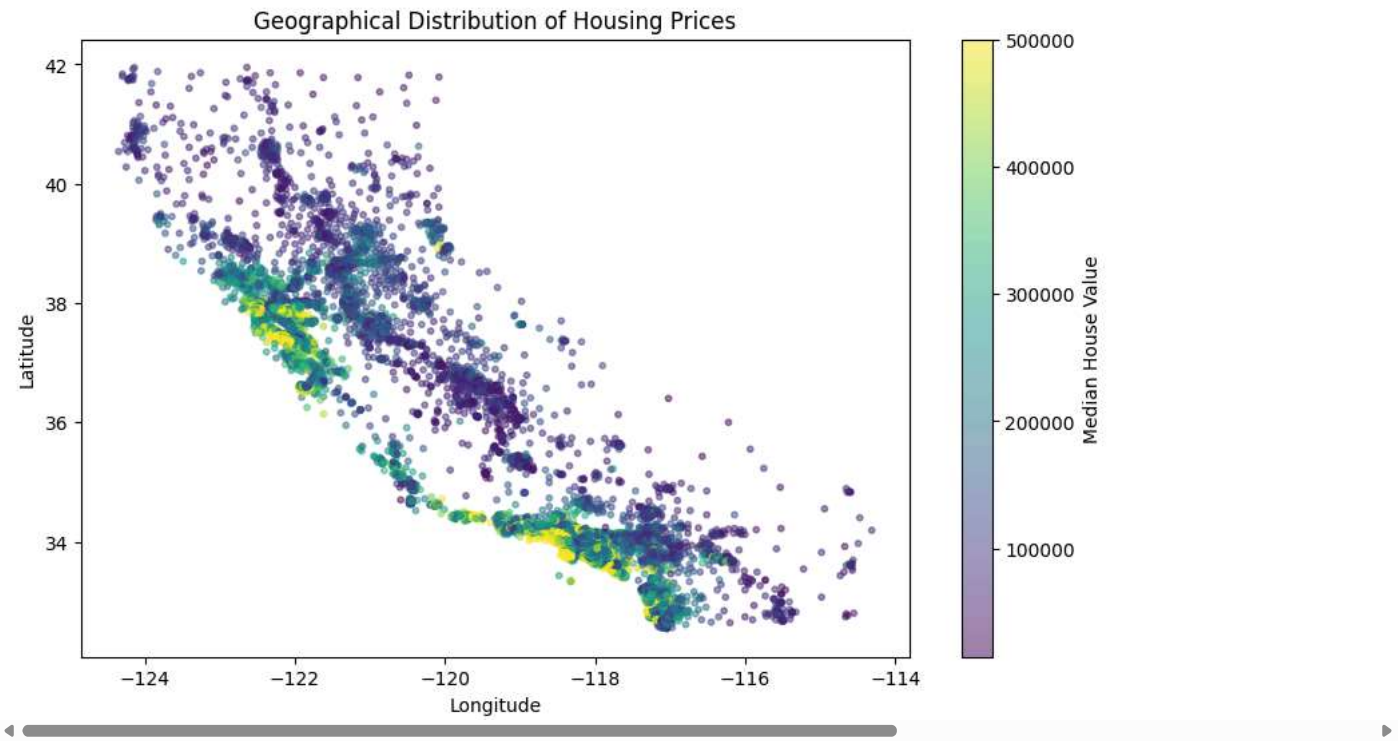
20640 rows × 11 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Difference between random and stratified sampling
# Random sampling:  Samples data points randomly without considering the distribution of a specific feature.
#                  This can lead to an uneven representation of different classes or categories in the test set,
#                  especially if some classes are less frequent.
# Stratified sampling:  Divides the data into homogeneous subgroups (strata) based on a specific feature (e.g., income category),
#                       then samples randomly from each stratum proportionally to its size in the overall dataset.
#                       This helps ensure that the test set has a similar distribution of the chosen feature as the original dataset,
#                       leading to more robust and reliable model evaluation, especially for classification tasks.
```

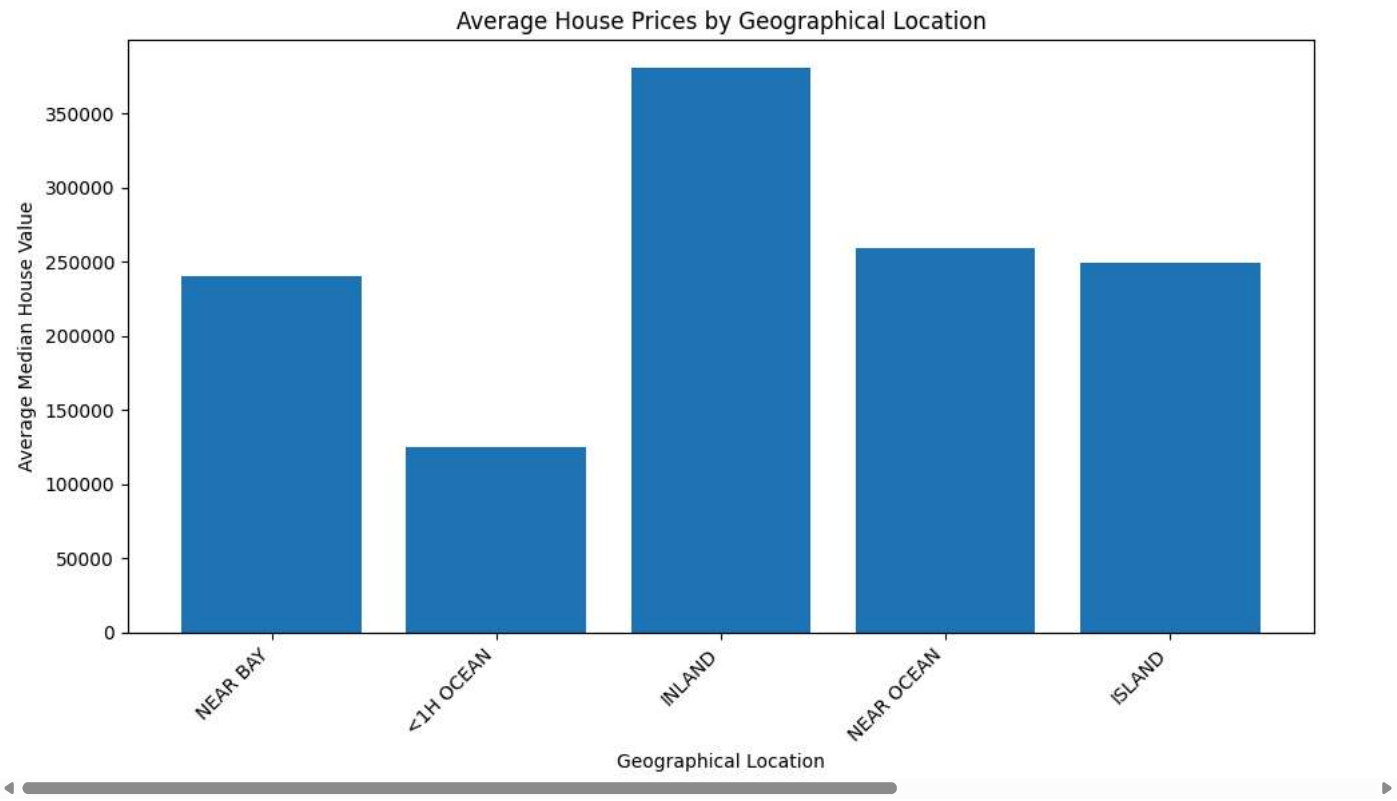
List the geographical features from the dataset and plot a graph to Visualize Geographical Data(what does the graph indicate w.r.t housing prices and location) LATITUDE AND LONGITUDE

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['longitude'], df['latitude'], c=df['median_house_value'], cmap='viridis', alpha=0.5, s=10)
plt.colorbar(scatter, label='Median House Value')
plt.title("Geographical Distribution of Housing Prices")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```



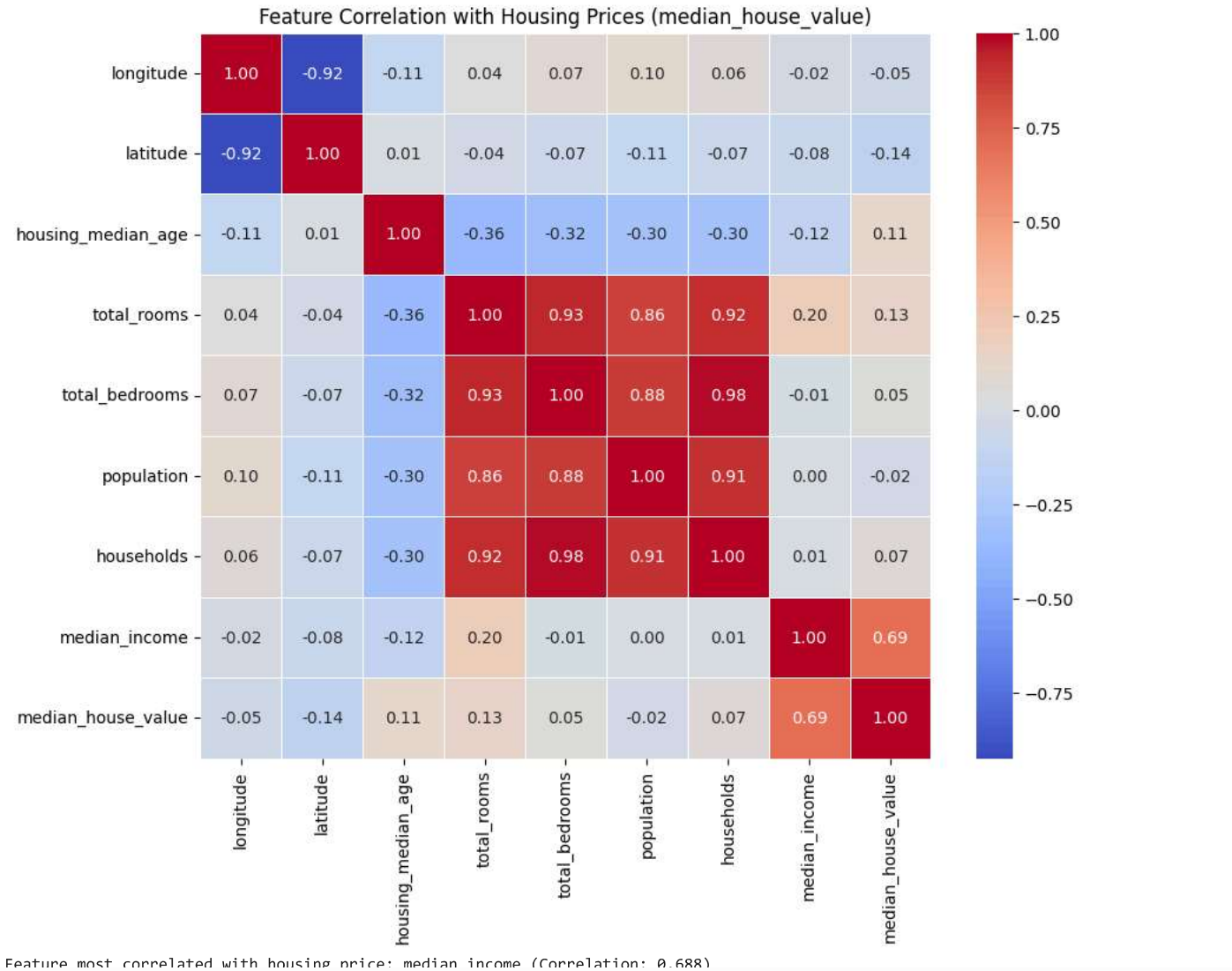
List the geographical features from the dataset and plot a graph to Visualize Geographical Data(what does the graph indicate w.r.t housing prices and location) OCEAN PROXIMITY

```
import matplotlib.pyplot as plt

geographical_features = df['ocean_proximity'].unique()

average_prices = df.groupby('ocean_proximity')['median_house_value'].mean()

plt.figure(figsize=(10, 6))
plt.bar(geographical_features, average_prices)
plt.xlabel("Geographical Location")
plt.ylabel("Average Median House Value")
plt.title("Average House Prices by Geographical Location")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```



Plot a graph to show features correlation with housing price. Which feature corelates to the maximum. Plot the graph for that with housing price and analyze what the graph indicate

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

correlation_matrix = df.corr(numeric_only=True)

# Plot the correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Feature Correlation with Housing Prices (median_house_value)')
plt.show()

# Extract correlation with housing prices (median_house_value) and drop 'median_house_value' itself
correlation_with_price = correlation_matrix['median_house_value'].drop('median_house_value')

# Find the feature with the highest correlation to housing prices
max_correlation_feature = correlation_with_price.idxmax()
max_correlation_value = correlation_with_price.max()

# Print the result
print(f"Feature most correlated with housing price: {max_correlation_feature} (Correlation: {max_correlation_value:.3f})")
```



List the features that could be combined to improve correlation and plot again to see if correlation has improved

```
plt.figure(figsize=(10, 6))
plt.scatter(df['median_income'], df['median_house_value'], alpha=0.5)
plt.xlabel('Median Income')
plt.ylabel('Median House Value')
plt.title('Housing Prices vs. Median Income')
plt.grid(True)
plt.show()
```


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

df['rooms_per_household'] = df['total_rooms'] / df['households']
df['bedrooms_per_household'] = df['total_bedrooms'] / df['households']
df['rooms_per_person'] = df['total_rooms'] / df['population']
df['income_per_person'] = df['median_income'] / df['population']
df['household_income'] = df['median_income'] * df['households']

correlation_matrix = df.corr(numeric_only=True)

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Updated Feature Correlation with Housing Prices (median_house_value)')
plt.show()

correlation_with_price = correlation_matrix['median_house_value'].drop('median_house_value')

max_correlation_feature = correlation_with_price.idxmax()
max_correlation_value = correlation_with_price.max()

print(f"Feature most correlated with housing price after combining: {max_correlation_feature} (Correlation: {max_correlation_value:.3f})")
```