In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
data =pd.read_excel('housing.xlsx') ## import the file into notebook
```

Intro: California Housing Prices

In [ ]:
```python
# in this dataset we can see how the house price varies in the California city with loaction of house , house holds
# proximity of ocean etc.
# here we can see 10 columns of various types:
# Discrete features: 1.latitude 2. housing_median_age 3.total_rooms 4.population 5.households 6. median_house_value
# Continuous features: 1.longitude 2.total_bedrooms 3.median_income
# Categorical (Nominal feature): 1. ocean_proximity
```

In [3]:
```python
data ## to see the quick view of the data
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41	880	129.0	322	126	8.3252	452600	NEAR BAY
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	358500	NEAR BAY
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	352100	NEAR BAY
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	341300	NEAR BAY
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	342200	NEAR BAY
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25	1665	374.0	845	330	1.5603	78100	INLAND
20636	-121.21	39.49	18	697	150.0	356	114	2.5568	77100	INLAND
20637	-121.22	39.43	17	2254	485.0	1007	433	1.7000	92300	INLAND
20638	-121.32	39.43	18	1860	409.0	741	349	1.8672	84700	INLAND
20639	-121.24	39.37	16	2785	616.0	1387	530	2.3886	89400	INLAND

20640 rows × 10 columns

In [4]:
```python
data.isnull().sum() ## to see if there is null values and count them
```

Out[4]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

In [6]:
```python
data.info() ## to see datatypes of columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 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  int64  
 3   total_rooms         20640 non-null  int64  
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  int64  
 6   households          20640 non-null  int64  
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  int64  
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

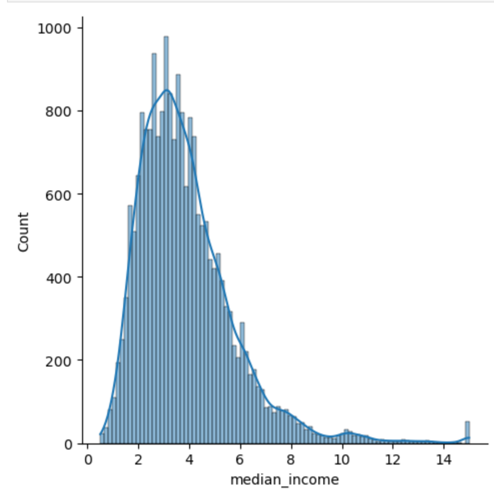
In [7]:
```python
data.describe() ## to see statistical description of the data
```

Out[7]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

Q1.What is the average median income of the data set and check the distribution of data using appropriate plots. Please explain the distribution of the plot.

In [8]:
```python
sns.displot(x='median_income',data = data,kde=True) ## to see the proper distribution kde plot and distribution plot is merged here
plt.show()
```



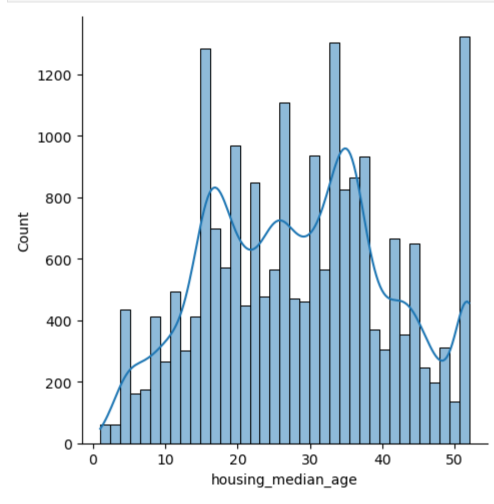
In [9]:
```python
# # from the plot it is concluded that it is a right skewed data.(majority data points are on left)
## Majority values of median income lies between 1.5 to 5.5.
```

In [43]:
```python
data['median_income'].mean() ## to find the median
```

Out[43]:
```
3.8706710029069766
```

Q2.Draw an appropriate plot to see the distribution of housing_median_age and explain your observations.

In [11]:
```python
sns.displot(x='housing_median_age',data = data,kde=True) #to see the proper distribution kde plot and distribution plot is merged here
plt.show()
```



In [ ]:
```python
## the data is nearly type of multimodal data.
```

Q3.Show with the help of visualization, how median_income and median_house_values are related?

In [12]:
```python
a=data[['median_income','median_house_value']] ## create a new dataframe of selecting these 2 columns
```

In [13]:
```python
a ## to show the new dataframe
```

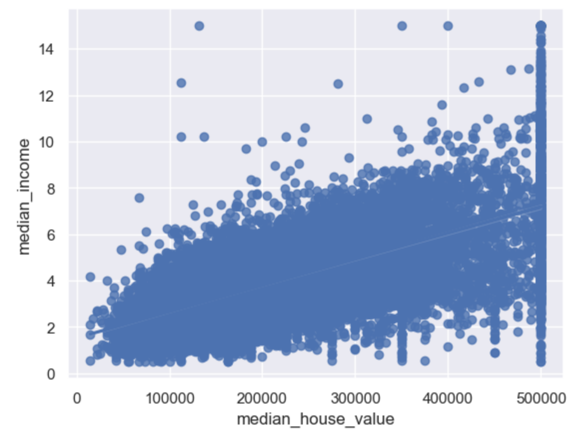
Out[13]:

	median_income	median_house_value
0	8.3252	452600
1	8.3014	358500
2	7.2574	352100
3	5.6431	341300
4	3.8462	342200
...	...	...
20635	1.5603	78100
20636	2.5568	77100
20637	1.7000	92300
20638	1.8672	84700
20639	2.3886	89400

20640 rows × 2 columns

In [14]:
```python
sns.set()
```

In [15]:
```python
sns.regplot(x='median_house_value', y='median_income', data=a) ## Regression plot is used to see the strength and relationship
                                                               ## between these 2 columns.
plt.show()
```



In [ ]:
```python
## the plot indicates approx 68% strength is present in between median income and median house values columns.
## the correlation value indicates average positive relationship between them.
```

In [16]:
```python
a.corr() ## to check the correlation value
```

Out[16]:

	median_income	median_house_value
median_income	1.000000	0.688075
median_house_value	0.688075	1.000000

Q4.Create a data set by deleting the corresponding examples from the data set for which total_bedrooms are not available

In [17]:
```python
data1=data.dropna() ## to delete the rows which even have 1 null value and save it to a new dataset
```

In [32]:
```python
data1 ## to see the new dataset after deleting null values
```

Out[32]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41	880	129.0	322	126	8.3252	452600	NEAR BAY
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	358500	NEAR BAY
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	352100	NEAR BAY
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	341300	NEAR BAY
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	342200	NEAR BAY
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25	1665	374.0	845	330	1.5603	78100	INLAND
20636	-121.21	39.49	18	697	150.0	356	114	2.5568	77100	INLAND
20637	-121.22	39.43	17	2254	485.0	1007	433	1.7000	92300	INLAND
20638	-121.32	39.43	18	1860	409.0	741	349	1.8672	84700	INLAND
20639	-121.24	39.37	16	2785	616.0	1387	530	2.3886	89400	INLAND

20433 rows × 10 columns

Q5.Create a data set by filling the missing data with the mean value of the total_bedrooms in the original data set.

In [19]:
```python
data2=data.fillna(value = data['total_bedrooms'].mean()) ## fillna is filling the missing values with the mean value of the data
                                                          ## and after that saving the result into a new dataset
```

In [20]:
```python
data2 ## to see the new dataset
```

Out[20]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41	880	129.0	322	126	8.3252	452600	NEAR BAY
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	358500	NEAR BAY
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	352100	NEAR BAY
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	341300	NEAR BAY
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	342200	NEAR BAY
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25	1665	374.0	845	330	1.5603	78100	INLAND
20636	-121.21	39.49	18	697	150.0	356	114	2.5568	77100	INLAND
20637	-121.22	39.43	17	2254	485.0	1007	433	1.7000	92300	INLAND
20638	-121.32	39.43	18	1860	409.0	741	349	1.8672	84700	INLAND
20639	-121.24	39.37	16	2785	616.0	1387	530	2.3886	89400	INLAND

20640 rows × 10 columns

In [21]:
```python
data['total_bedrooms'].mean() # to see the mean value of 'total_bedrooms' column
```

Out[21]:
```
537.8705525375618
```

In [48]:
```python
data2.iloc[290:295] ## to verify if null value is replaced by mean value or not
```

Out[48]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	total_bedroom_size
290	-122.16	37.77	47	1256	537.870553	570	218	4.3750	161900	NEAR BAY	medium
291	-122.16	37.77	48	977	194.000000	446	180	4.7708	156300	NEAR BAY	medium
292	-122.16	37.77	45	2324	397.000000	968	384	3.5739	176000	NEAR BAY	medium
293	-122.16	37.77	39	1583	349.000000	857	316	3.0958	145800	NEAR BAY	medium
294	-122.17	37.77	39	1612	342.000000	912	322	3.3958	141900	NEAR BAY	medium

Q6.Write a programming construct (create a user defined function) to calculate the median value of the data set wherever required.

In [57]:
```python
def median(col_name):  ## define a function with col_name parameter
    l1=[]      ## create an empty list
    for a,b in enumerate(data[col_name]): ##using enumerate function to extract column values
        l1.append(b)  ## append the values into the empty list
    l1.sort()          ## after appending sorting the values
    length=len(l1)      ## define a variable to hold the length value
    middle=length//2
    if length%2==0:
        median=(l1[middle-1] + l1[middle])/2      ## used a if else condition
    else:
        median=l1[middle]
    return median

median('population')  ## calling the median function
```

Out[57]:
```
1166.0
```

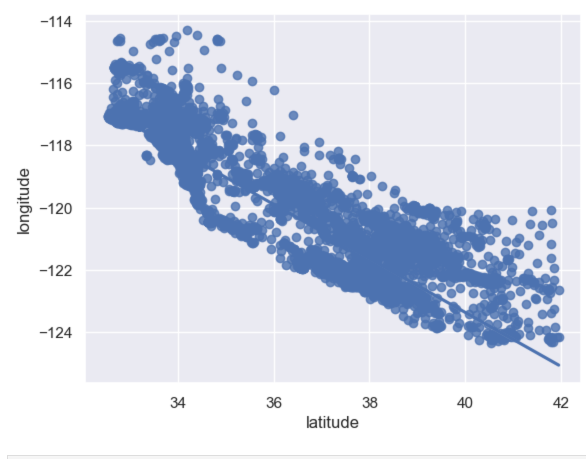
In [54]:
```python
data['population'].median() ## to check and compare the median value
```

Out[54]:
```
1166.0
```

Q7. Plot latitude versus longitude and explain your observations.

In [23]:
```python
sns.set()
```

In [55]:
```python
sns.regplot(x='latitude', y='longitude', data=data) ## Regression plot is used to see the strength and relationship
                                                     ## between these 2 columns.
plt.show()
```



In [56]:
```python
## the plot indicates negative relationship in between median latitude and longitude columns.
```

Q8.Create a data set for which the ocean_proximity is 'Near ocean'.

In [25]:
```python
data3=data[data['ocean_proximity']=='NEAR OCEAN'] ##  applying filter condition and create a new dataset
```

In [26]:
```python
data3 ## to show thw new dataset
```

Out[26]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
1850	-124.17	41.80	16	2739	480.0	1259	436	3.7557	109400	NEAR OCEAN
1851	-124.30	41.80	19	2672	552.0	1298	478	1.9797	85800	NEAR OCEAN
1852	-124.23	41.75	11	3159	616.0	1343	479	2.4805	73200	NEAR OCEAN
1853	-124.21	41.77	17	3461	722.0	1947	647	2.5795	68400	NEAR OCEAN
1854	-124.19	41.78	15	3140	714.0	1645	640	1.6654	74600	NEAR OCEAN
...	...	...	...	...	...	...	...	...	...	...
20380	-118.83	34.14	16	1316	194.0	450	173	10.1597	500001	NEAR OCEAN
20381	-118.83	34.14	16	1956	312.0	671	319	6.4001	321800	NEAR OCEAN
20423	-119.00	34.08	17	1822	438.0	578	291	5.4346	428600	NEAR OCEAN
20424	-118.75	34.18	4	16704	2704.0	6187	2207	6.6122	357600	NEAR OCEAN
20425	-118.75	34.17	18	6217	858.0	2703	834	6.8075	325900	NEAR OCEAN

2658 rows × 10 columns

In [27]:
```python
data['ocean_proximity'].value_counts() ## to count how many rows for each category in that column
```

Out[27]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

Q9.Find the mean and median of the median income for the data set created in question 8.

In [28]:
```python
data3['median_income'].mean() ##to find mean using mean function
```

Out[28]:
```
4.0057848006019565
```

In [30]:
```python
data3['median_income'].median() ##to find median using median function
```

Out[30]:
```
3.64705
```

Q10.Please create a new column named total_bedroom_size. If the total bedrooms is 10 or less, it should be quoted as small. If the total bedrooms is 11 or more but less than 1000, it should be medium, otherwise it should be considered large.

In [ ]:
```python
## The new dataset is used which is created after filling the missing values of total_bedroom column
## with mean value.
```

In [35]:
```python
data2.loc[data2['total_bedrooms']<=10 , 'total_bedroom_size']='small' ## used loc function to select the rows and new column and
                                                                      # according numerical condition put values in new column.
```

In [36]:
```python
data2.loc[(data2['total_bedrooms']>=11) & (data2['total_bedrooms']<1000)  , 'total_bedroom_size']='medium'
```

In [38]:
```python
data2.loc[data2['total_bedrooms']>=1000 , 'total_bedroom_size']='large'
```

In [39]:
```python
data2 ## to see the new dataset
```

Out[39]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	total_bedroom_size
0	-122.23	37.88	41	880	129.0	322	126	8.3252	452600	NEAR BAY	medium
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	358500	NEAR BAY	large
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	352100	NEAR BAY	medium
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	341300	NEAR BAY	medium
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	342200	NEAR BAY	medium
...	...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25	1665	374.0	845	330	1.5603	78100	INLAND	medium
20636	-121.21	39.49	18	697	150.0	356	114	2.5568	77100	INLAND	medium
20637	-121.22	39.43	17	2254	485.0	1007	433	1.7000	92300	INLAND	medium
20638	-121.32	39.43	18	1860	409.0	741	349	1.8672	84700	INLAND	medium
20639	-121.24	39.37	16	2785	616.0	1387	530	2.3886	89400	INLAND	medium

20640 rows × 11 columns