Project Title: Housing Data Analysis

Introduction:

This project aims to analyze the California housing dataset, which contains information about housing characteristics in different regions of California. We'll explore and visualize various features of the dataset and gain insights into the California housing market.

```
#importing required packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading data to data frame

```
url = "/content/drive/MyDrive/DATA_sets/housing+(1).xlsx"
df = pd.read_excel(url)   # convert to pandas dataframe
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41	880	129.0	322	126	8.3252	452600
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	358500
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	352100
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	341300
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	342200

```
df.columns  # columns attribute returns column labels
```

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

```
df.shape#dataframe contains 20640 rows and 10 columns
```

```
(20640, 10)
```

```
df.info() #checking columns and their data types
```

```
<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
```

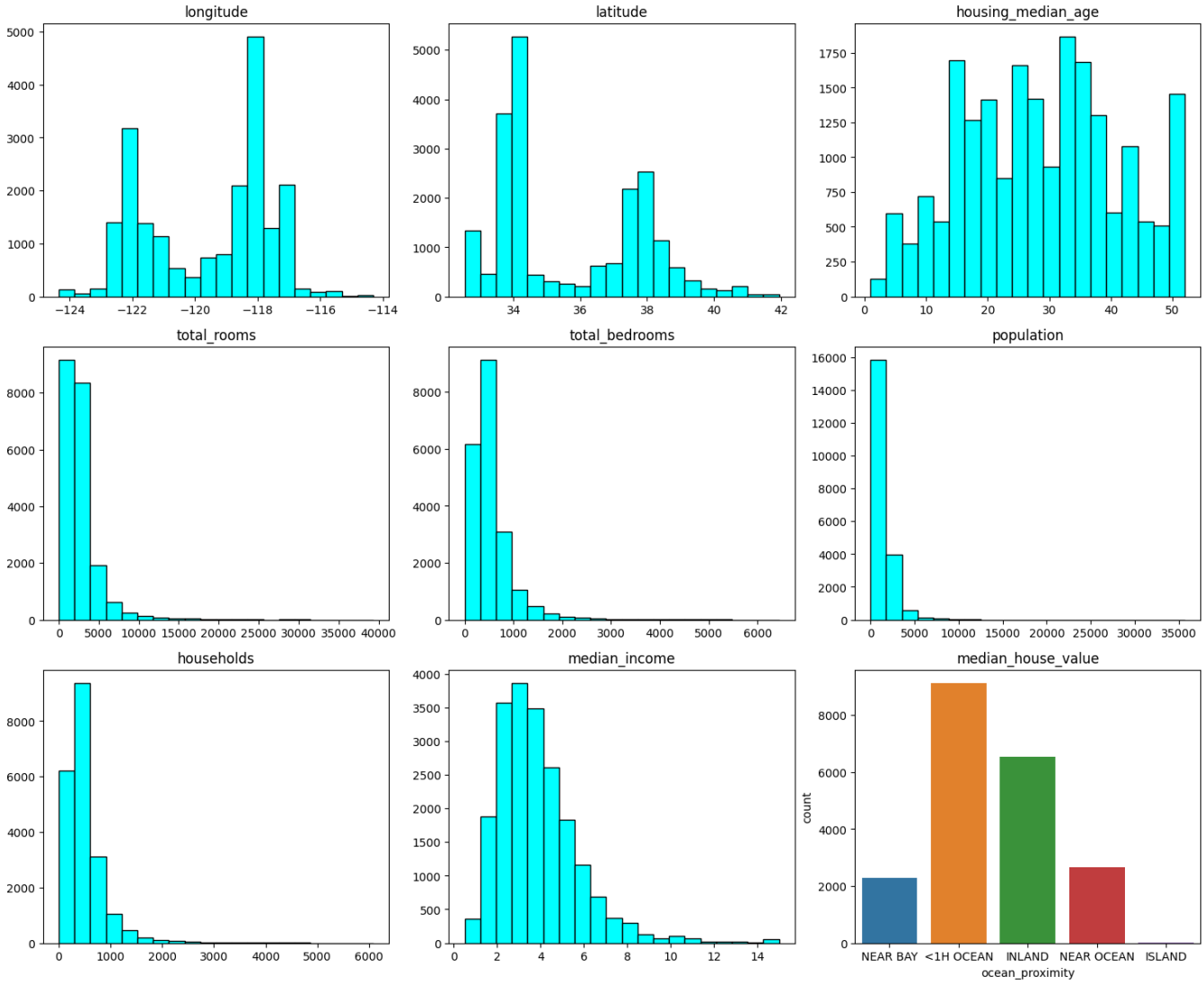
```
df.describe() #calculating basic statistics
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	2

```
#sns.pairplot(df)
```

```
df[['longitude', 'latitude', 'housing_median_age', 'total_rooms',
'total_bedrooms', 'population', 'households', 'median_income',
           'median_house_value']].hist(bins=20, figsize=(15, 12),facecolor="cyan" ,edgecolor='black',grid = False)
plt.tight_layout()  # To ensure proper spacing
#plt.show()
sns.countplot(x = df["ocean_proximity"])

plt.show()
```

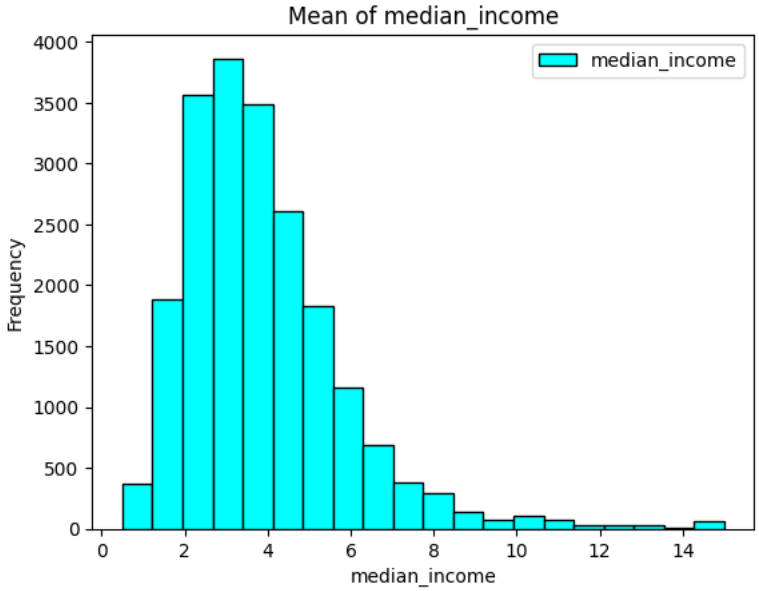


1. What is the average median income of the data set and check the distribution of data using appropriate plots. Please explain the distribution of the plot.

```
df["median_income"].mean()    # mean() to calculate mean/average
```

```
3.8706710029069766
```

```
plt.hist(df["median_income"],bins = 20,facecolor="cyan" ,edgecolor='black',label="median_income") #histogram plot to check the distribut
plt.xlabel("median_income")
plt.ylabel("Frequency")
plt.title("Mean of median_income") #title to the plot
plt.legend(loc="upper right")
plt.show()
```



The distribution of the "median_income" plot

Double-click (or enter) to edit

Shape: The distribution of median income is skewed to Right.

peakedness : unimodal- distribution has one peak.

Center/median: The data seem to be centered around 3.8.

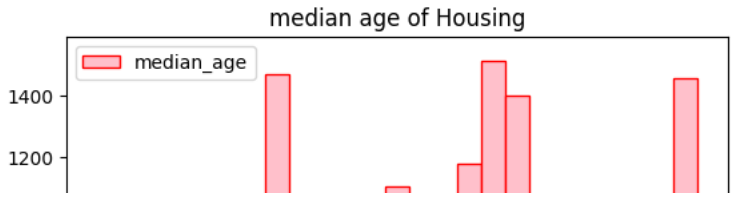
Spread: The data range from minimum 0 to maximum 15.

Outliers: There is one probable outlier to the far right near 14

If we look at the median income histogram more closely most of the median income values are clustered around 2 to 5 The median income is a distribution with a long tail. It means that the salary of people is more or less normally distributed but there is some people getting a high salary

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

```
plt.hist(df["housing_median_age"],facecolor="pink",edgecolor="red",bins=25,label="median_age")
plt.xlabel("housing_median_age")   # nameing of x-axis
plt.title("median age of Housing")  #title
plt.legend(loc="upper left")  #labelling on upper left location
plt.show()
```



The distribution of housing_median_age:

Shape: the distribution is more or less uniform ,we can call it is aproximately symmetric , as in the histogram below, the distribution forms an approximate mirror image with respect to the center of the distribution.

Center: The data seem to be centered around 28.63.

Spread: The data range from minimum 1 to maximum 52, so the approximate range equals 51

Outliers: There is no probable outlier

peakedness (modality) — The target distribution has long tail as well.

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

```
#df["median_income"].corr(df["median_house_value"])
```

```
plt.scatter(df["median_income"],df["median_house_value"],alpha =0.3)#scatter plot for relation between income and house value
plt.xlabel('median_income')#name of x axis
plt.ylabel('median_house_value')#name of y axis
plt.title("median_income vs median_house_values")#title

plt.show()    #show the plot
```

A scatter plot will show the relationship between median_income and median_house_values.

Observation: we can say that relation between median_income and the median_house_value is positive linear relationship

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

```
df.isnull().sum() #Handle missing values :207 null values in Column total_bedrooms
```

```
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
```

```
df1 = df
df1.columns
df2=df1.dropna(subset= ["total_bedrooms"])   #dropna() method removes the rows that contains NULL value
df2.isnull().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

```
df2["total_bedrooms"].size
```

```
20433
```

```
df["total_bedrooms"].size
```

```
20640
```

```
df2["total_bedrooms"].mean()
```

```
537.8705525375618
```

```
df["total_bedrooms"].mean()
```

```
537.8705525375618
```

```
df2.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41	880	129.0	322
1	-122.22	37.86	21	7099	1106.0	2401
2	-122.24	37.85	52	1467	190.0	496
3	-122.25	37.85	52	1274	235.0	558
4	-122.25	37.85	52	1627	280.0	565

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

```
bed_mean = df["total_bedrooms"].mean()#calculate and storing the total_bedrooms mean to a variable "bed_mean"
bed_mean
```

```
537.8705525375618
```

```
df["total_bedrooms"].fillna(bed_mean,inplace=True)#filling null values with mean of totalbedrooms inplace=true means changes are applie

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

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

```
df["total_bedrooms"].mean()
```

```
537.8705525375617
```

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

```
def Median(a, n):
  a.sort()
  if n % 2 == 0:
    m1 = a[n//2]
    m2 = a[n//2 - 1]
    return (m1 + m2)/2
  else:
    return a[n//2]
```

```
a=df['median_income'].tolist() #median income column to list and passing it to function
n=len(a)
Median(a,n)
```

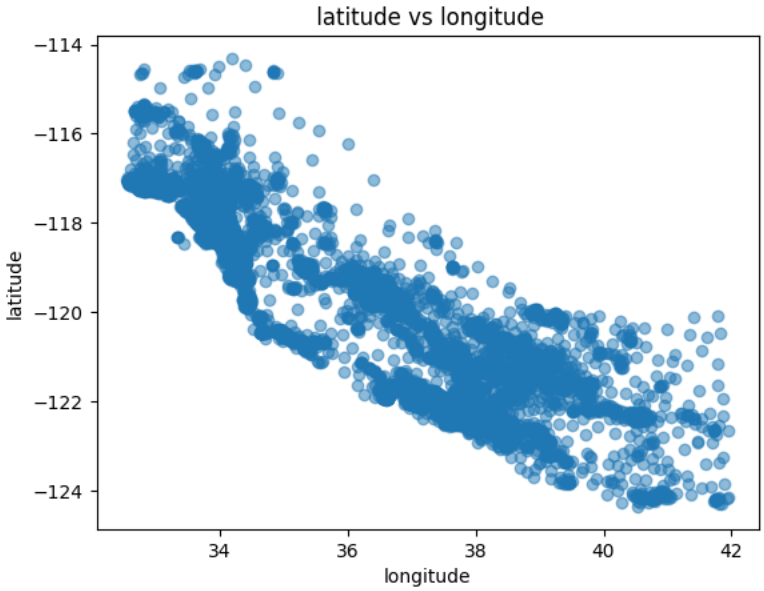
```
3.5347999999999997
```

7. Plot latitude versus longitude and explain your observations.

```
df["latitude"].corr(df["longitude"])
```

```
-0.924664433915041
```

```
plt.scatter(df["latitude"],df["longitude"] ,alpha =0.5) #selected scatter plot to draw the relationship between latitude and longitude
plt.xlabel("longitude")     #name of x axis
plt.ylabel("latitude")   #name of y axis
plt.title(" latitude vs longitude") #title
plt.show()  #show the plot
```



Observation

we can see the relation between latitude vs longitude is strong negative and linear relationship

8. Create a data set for which the ocean_proximity is ‘Near ocean’.

```
da=df.loc[df['ocean_proximity']=='NEAR OCEAN']  # creating a new data set where  ocean_proximity  is ‘Near ocean’
da
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
1850	-124.17	41.80	16	2739	480.0	12
1851	-124.30	41.80	19	2672	552.0	12
1852	-124.23	41.75	11	3159	616.0	13
1853	-124.21	41.77	17	3461	722.0	19

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

```
da["median_income"].agg(['mean','median']) #using aggregation function to calculate both mean and median
```

```
mean      4.005785
median    3.647050
Name: median_income, dtype: float64
```

we can observe that the mean and median income of people who houses are near the ocean .

2658 rows × 10 columns

10. 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.

```
da["total_bedroom_size"] = 0    #creating a new column to the data frame
```

```
<ipython-input-67-86fb70528045>:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
```

```
da.loc[da["total_bedrooms"] <= 10, "total_bedroom_size"] = "small"#using loc method to select the rows that meet the condition and assi
da.loc[(da["total_bedrooms"] >= 11) & (da["total_bedrooms"] <= 1000), "total_bedroom_size"] = "medium"
da.loc[da["total_bedrooms"] > 1000, "total_bedroom_size"] = "large"
```

```
da.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
1850	-124.17	41.80	16	2739	480.0	125
1851	-124.30	41.80	19	2672	552.0	129
1852	-124.23	41.75	11	3159	616.0	134
1853	-124.21	41.77	17	3461	722.0	194
1854	-124.19	41.78	15	3140	714.0	164

```
da["total_bedroom_size"].value_counts()
```

```
medium    2443
large      208
small        7
Name: total_bedroom_size, dtype: int64
```

```
sns.countplot(da,x="total_bedroom_size")
```

```
<Axes: xlabel='total_bedroom_size', ylabel='count'>
```