In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [2]:
```
# Load the dataset
data=pd.read_csv("housing.csv")
```

In [3]:
```
# checking the dataset
data
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	1
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	11
2	-122.24	37.85	52.0	1467.0	190.0	496.0	1
3	-122.25	37.85	52.0	1274.0	235.0	558.0	2
4	-122.25	37.85	52.0	1627.0	280.0	565.0	2
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	3
20636	-121.21	39.49	18.0	697.0	150.0	356.0	1
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	4
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	3
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	5

20640 rows × 10 columns

In [4]:
```
# converting string data to numerical data
# in the above dataset we can identify that the column ocean_proximity is having co
data.ocean_proximity.value_counts()
```

Out[4]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

In [5]:
```
# one-hot encoded 'ocean_proximity'
data=pd.get_dummies(data.ocean_proximity,dtype=int)
```

In [6]:
```
data
```

Out[6]:

	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0
...	...	...	...	...	...
20635	0	1	0	0	0
20636	0	1	0	0	0
20637	0	1	0	0	0
20638	0	1	0	0	0
20639	0	1	0	0	0

20640 rows × 5 columns

In [7]:

```
original_data = pd.read_csv('housing.csv') # Load your original data
# Concatenate the one-hot encoded column with the original DataFrame
data = pd.concat([original_data, data], axis=1)
# Drop the original 'ocean_proximity' column if needed
data.drop(['ocean_proximity'], axis=1, inplace=True)
# Handle missing and infinite values in one-hot encoded columns
ocean_proximity_columns = ['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN
data[ocean_proximity_columns] = data[ocean_proximity_columns].fillna(0).astype(int
# remove duplicate values
data = data.loc[:, ~data.columns.duplicated()]
```

In [8]:

```
data
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20640 rows × 14 columns

In [9]:
```
# check for any missing or null data
missing_data = data.isnull().sum()
print("Missing Data:\n", missing_data)
```

```
Missing Data:
 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
<1H OCEAN             0
INLAND                0
ISLAND                0
NEAR BAY              0
NEAR OCEAN            0
dtype: int64
```

In [10]:
```
# return data frame with not null values
data.dropna(inplace=True)
# again check for any null values
missing_data= data.isnull().sum()
print("Missing Data:\n", missing_data)
```

```
Missing Data:
 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
<1H OCEAN           0
INLAND              0
ISLAND              0
NEAR BAY            0
NEAR OCEAN          0
dtype: int64
```

In [11]:
```
data
```

Out[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20433 rows × 14 columns

In [12]:
```
#identifing the target data
x=data.drop(['median_house_value'],axis=1)
y=data['median_house_value']
```

In [13]:
```
# splitting data into train and test data
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2)
```

In [15]:
```
#comparing correlations of x and y
train_data=x_train.join(y_train)
```

In [16]:
```
train_data
```

Out[16]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
14125	-117.08	32.75	16.0	1111.0	328.0	930.0	
12095	-117.26	33.84	12.0	1159.0	209.0	523.0	
6742	-118.08	34.13	28.0	4465.0	985.0	2273.0	
6033	-117.73	34.08	28.0	5173.0	1069.0	3502.0	
7943	-118.13	33.86	45.0	1320.0	256.0	645.0	
...	...	...	...	...	...	...	
369	-122.15	37.75	40.0	1445.0	256.0	849.0	
13431	-117.42	34.10	18.0	3977.0	809.0	2231.0	
19878	-119.27	36.34	7.0	3433.0	626.0	1793.0	
18702	-122.34	40.57	24.0	1610.0	307.0	748.0	
6841	-118.12	34.06	23.0	1190.0	347.0	965.0	

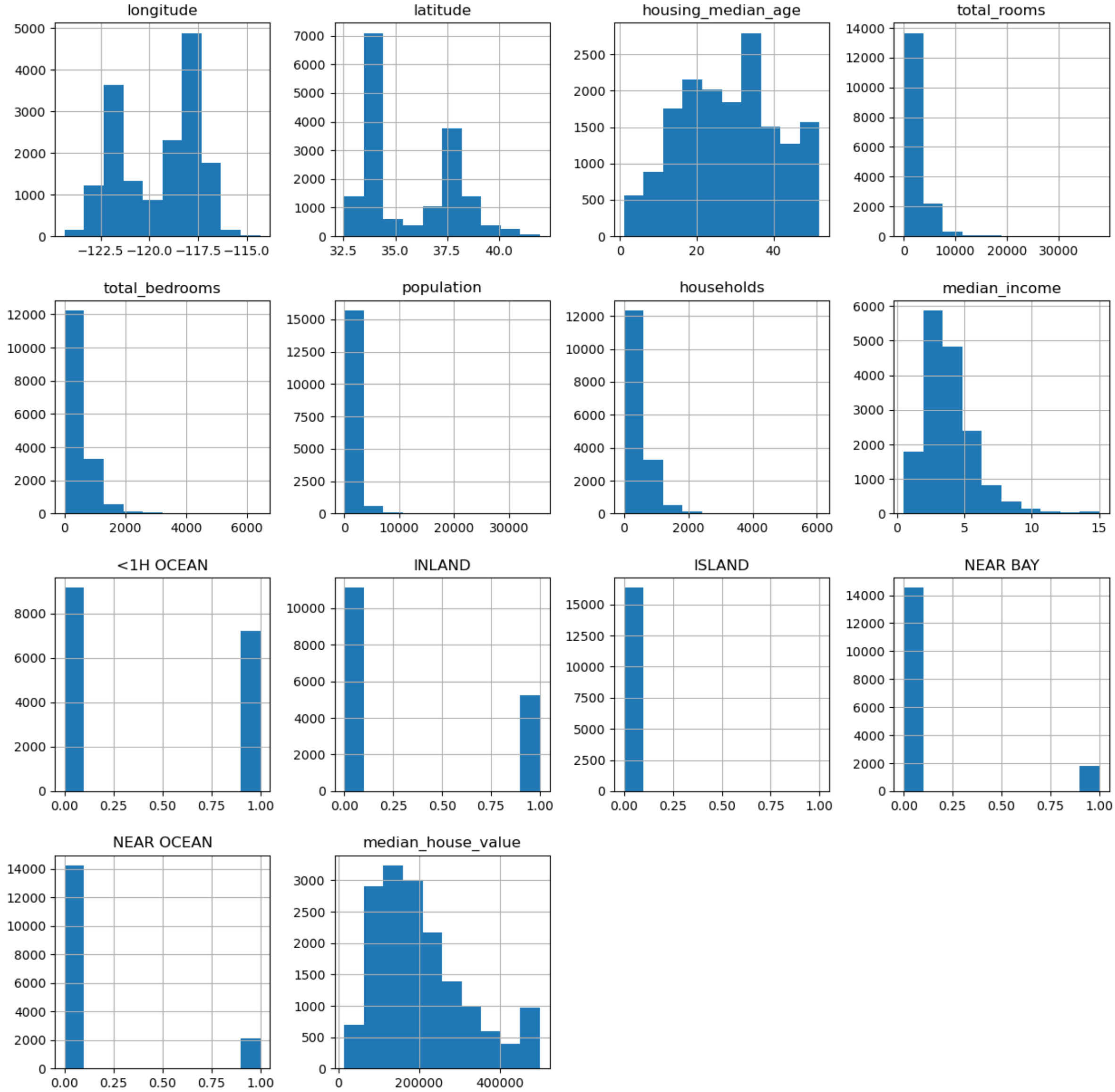
16346 rows × 14 columns

In [17]:

```
train_data.hist(figsize=(15,15))
```

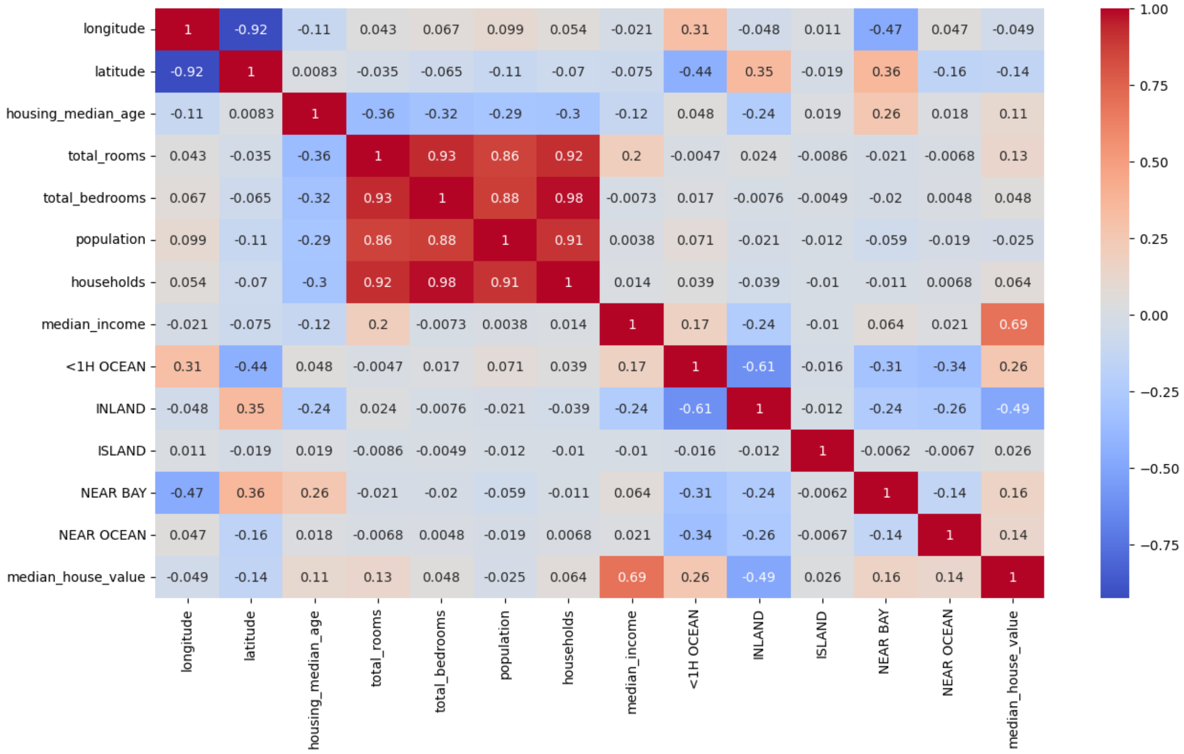
Out[17]:

```
array([[<Axes: title={'center': 'longitude'}>,
        <Axes: title={'center': 'latitude'}>,
        <Axes: title={'center': 'housing_median_age'}>,
        <Axes: title={'center': 'total_rooms'}>],
       [<Axes: title={'center': 'total_bedrooms'}>,
        <Axes: title={'center': 'population'}>,
        <Axes: title={'center': 'households'}>,
        <Axes: title={'center': 'median_income'}>],
       [<Axes: title={'center': '<1H OCEAN'}>,
        <Axes: title={'center': 'INLAND'}>,
        <Axes: title={'center': 'ISLAND'}>,
        <Axes: title={'center': 'NEAR BAY'}>],
       [<Axes: title={'center': 'NEAR OCEAN'}>,
        <Axes: title={'center': 'median_house_value'}>, <Axes: >,
        <Axes: >]], dtype=object)
```



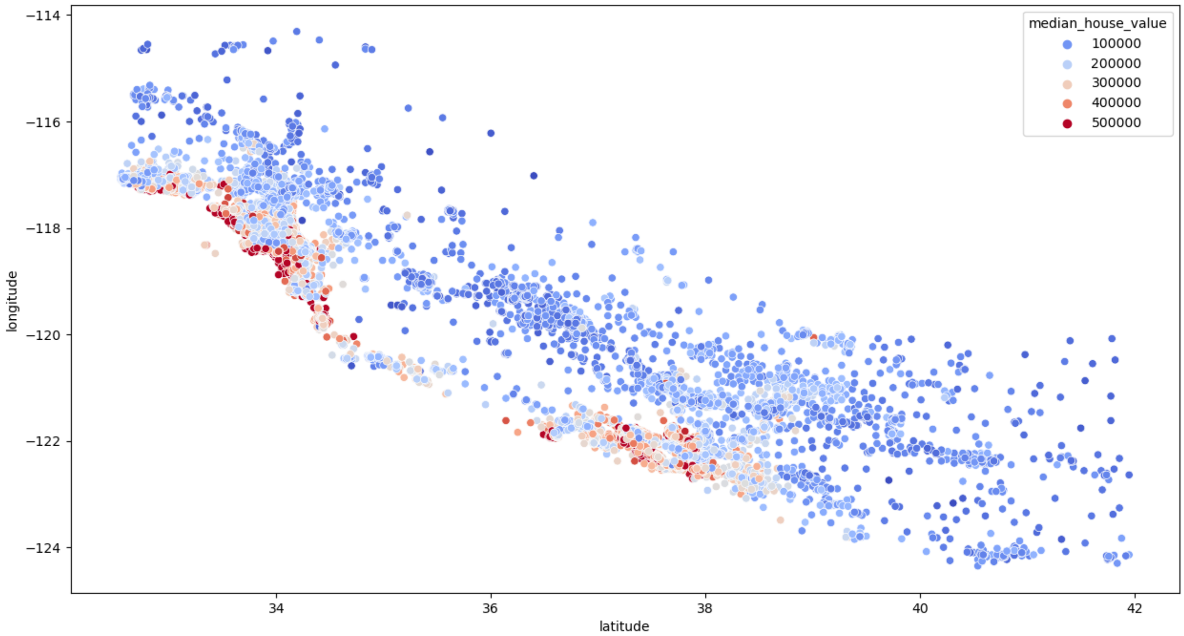

```
In [18]: # to visualize the correlation matrix of train_data
         plt.figure(figsize=(15,8))
         sns.heatmap(train_data.corr(),annot=True,cmap="coolwarm")
```

Out[18]: <Axes: >



In [19]:
```
plt.figure(figsize=(15,8))
sns.scatterplot(x="latitude",y="longitude",data=train_data,hue="median_house_value'
```

Out[19]:
```
<Axes: xlabel='latitude', ylabel='longitude'>
```



In [20]:
```
# training linear regression model using training data
linear_reg=LinearRegression()
linear_reg.fit(x_train,y_train)
```

Out[20]:
```
▾ LinearRegression
LinearRegression()
```

In [21]:
```
y_predict=linear_reg.predict(x_test)
print(y_predict)
print(y_test)
```

```
[210919.76583871 163739.01469614  96084.15941173 ... 152004.8710702
  32766.45012301 308698.46745771]
7440     169300.0
2584      99200.0
19501     75500.0
3526     252100.0
2602      83800.0
           ...
5973     230000.0
607      184900.0
12109    160600.0
9625      56300.0
8722     343000.0
Name: median_house_value, Length: 4087, dtype: float64
```

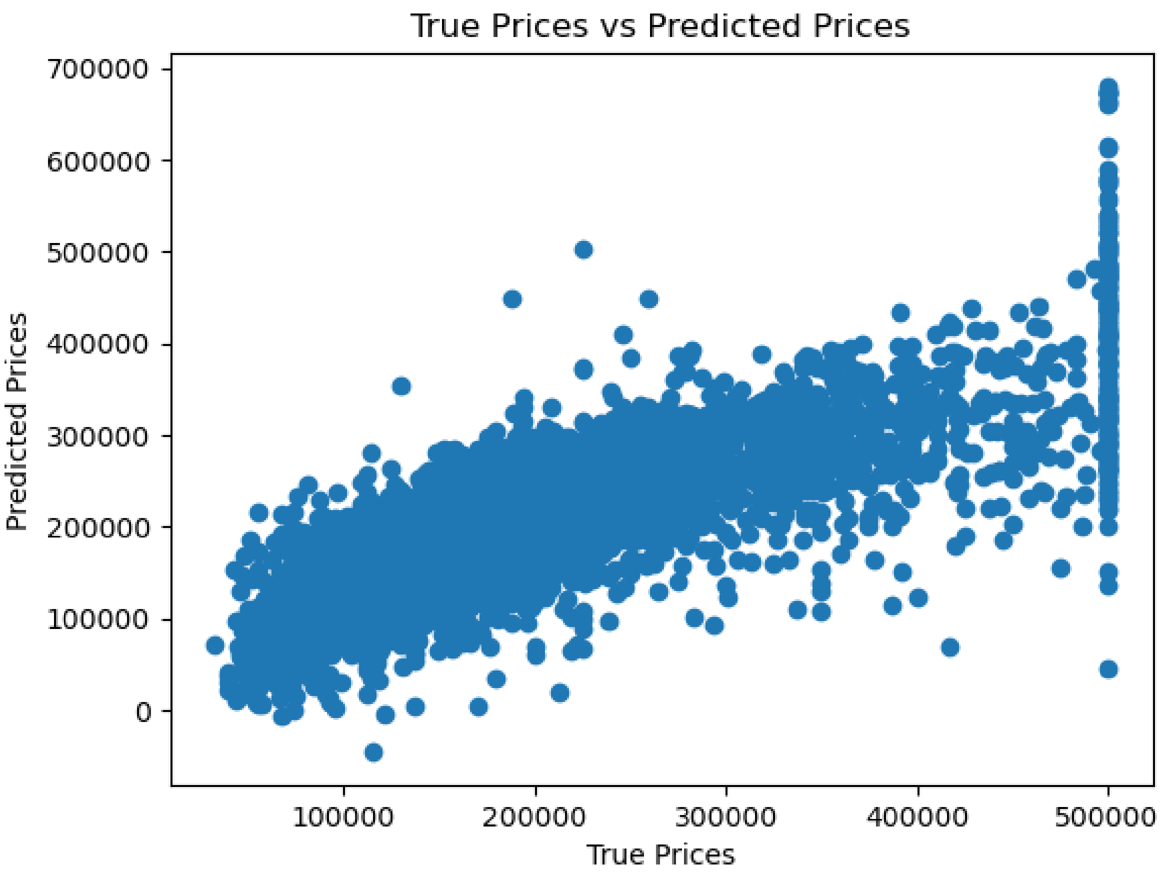
In [22]:
```
# determining the performance of the model using mean squared error metrics
mse = mean_squared_error(y_test, y_predict)
print("Coefficients:", linear_reg.coef_)
print("Intercept:", linear_reg.intercept_)
print("Mean Squared Error:", mse)
```

```
Coefficients: [-2.69815840e+04 -2.57156693e+04  1.06219337e+03 -5.84685048e+00
  9.98958960e+01 -3.66369001e+01  4.58663868e+01  3.90950573e+04
 -2.23820929e+04 -6.24467586e+04  1.30365199e+05 -2.66545374e+04
 -1.88818105e+04]
Intercept: -2258813.524703438
Mean Squared Error: 4679615894.410932
```

In [23]:
```
# determining the performance of the model using r2_score metrics
print('Coefficient of determination : %.5f' % r2_score(y_test,y_predict))
```

```
Coefficient of determination : 0.64900
```

In [24]:
```
# Visualizing the predictions
plt.scatter(y_test, y_predict)
plt.xlabel("True Prices")
plt.ylabel("Predicted Prices")
plt.title("True Prices vs Predicted Prices")
plt.show()
```


True Prices vs Predicted Prices
Predicted Prices
True Prices
700000
600000
500000
400000
300000
200000
100000
0
100000
200000
300000
400000
500000