In [12]:
```
import numpy as np
import pandas as pd
import seaborn as sns
df=pd.read_csv("/content/KC_housing_data.csv")
df
```

Out[12]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	co
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	
...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	

4600 rows × 18 columns

In [2]:
```
df.shape
```

Out[2]:
```
(4600, 18)
```

In [3]:
```
df.size
```

Out[3]:
```
82800
```

In [4]:
```
df.head()
```

Out[4]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4

In [5]:
```
df.tail()
```

Out[5]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	c
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	0	
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	0	
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	0	
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	0	
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	0	

In [6]:
```
df.columns
```

Out[6]:
```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

In [7]:
```
df.dtypes
```

Out[7]:
```
date              object
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street            object
city              object
statezip          object
country           object
dtype: object
```

In [8]:
```
df.isna().sum()
```

Out[8]:
```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [16]:
```
# COUNT COLUMNS WITH STRING DATATYPES
#1.COLUMN: date
date=df['date'].value_counts()
print(date)
```

```
2014-06-23 00:00:00    142
2014-06-25 00:00:00    131
2014-06-26 00:00:00    131
2014-07-08 00:00:00    127
2014-07-09 00:00:00    121
                      ...
2014-06-07 00:00:00      4
2014-07-06 00:00:00      3
2014-07-04 00:00:00      2
2014-05-11 00:00:00      2
2014-05-17 00:00:00      1
Name: date, Length: 70, dtype: int64
```

In [17]:
```
# 2.COLUMN: street
street=df['street'].value_counts()
print(street)
```

```
2520 Mulberry Walk NE     4
2500 Mulberry Walk NE     3
9413 34th Ave SW          2
6008 8th Ave NE           2
11034 NE 26th Pl          2
                         ..
1404 Broadmoor Dr E       1
3249 E Ames Lake Dr NE    1
6032 35th Ave NE          1
1006 NE Ravenna Blvd      1
18717 SE 258th St         1
Name: street, Length: 4525, dtype: int64
```

In [18]:
```
# 3.COLUMN: city
city=df['city'].value_counts()
print(city)
```

```
Seattle                1573
Renton                  293
Bellevue                286
Redmond                 235
Issaquah                187
Kirkland                187
Kent                    185
Auburn                  176
Sammamish               175
Federal Way             148
Shoreline               123
Woodinville             115
Maple Valley             96
Mercer Island            86
Burien                   74
Snoqualmie               71
Kenmore                  66
Des Moines               58
North Bend               50
Covington                43
Duvall                   42
Lake Forest Park         36
Bothell                  33
Newcastle                33
SeaTac                   29
Tukwila                  29
Vashon                   29
Enumclaw                 28
Carnation                22
Normandy Park            18
Clyde Hill               11
Medina                   11
Fall City                11
Black Diamond             9
Ravensdale                7
Pacific                   6
Algona                    5
Yarrow Point              4
Skykomish                 3
Preston                   2
Milton                    2
Inglewood-Finn Hill       1
Snoqualmie Pass           1
Beaux Arts Village        1
Name: city, dtype: int64
```

In [ ]:
```
# 4.COLUMN: statezip
state=df['statezip'].value_counts()
print(state)
```

```
WA 98103    148
WA 98052    135
WA 98117    132
WA 98115    130
WA 98006    110
           ...
WA 98047      6
WA 98288      3
WA 98050      2
WA 98354      2
WA 98068      1
Name: statezip, Length: 77, dtype: int64
```

In [ ]:
```
# 5.COLUMN:country
country=df['country'].value_counts()
print(country)
```

```
USA    4600
Name: country, dtype: int64
```

In [19]:
```
# DROP UNWANTED COLUMNS
df1=df.drop(['date','street','city','statezip','country'],axis=1)
df1
```

Out[19]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqf
0	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	
1	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	
2	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	
3	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	
4	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	
...	...	...	...	...	...	...	...	...	...	
4595	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	
4596	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	
4597	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	
4598	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	
4599	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	

4600 rows × 13 columns

In [20]:
```
df1.dtypes
```

Out[20]:
```
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
dtype: object
```

In [21]:
```
df1.isna().sum()
```

Out[21]:
```
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
dtype: int64
```

In [22]:
```
# SEPERATING INPUT X AND OUTPUT Y
x=df1.drop(['condition'],axis=1).values
y=df1['condition'].values
```

In [23]:
```
# SEPERATING TRAIN AND TEST DATA
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,\
                                               test_size=0.20,\
                                               random_state=42)
```

In [24]:
```
# NORMALIZATION
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

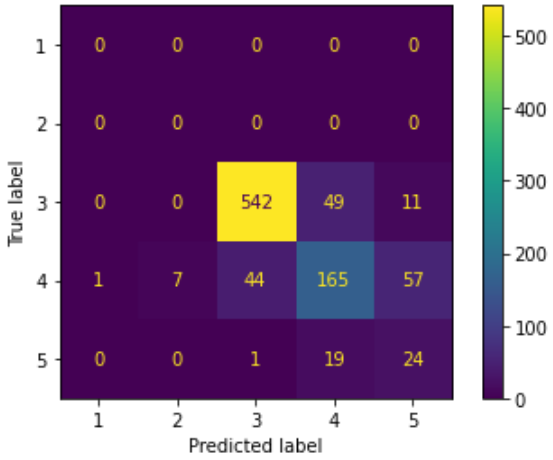
In [50]:
```
# ML MODEL CREATION
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(criterion='entropy')
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```

In [51]:
```
# PERFORMANCE CHECK
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,accurac
y_score,classification_report
mat=confusion_matrix(y_pred,y_test)
mat
```

Out[51]:
```
array([[  0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0],
       [  0,   0, 545,  51,  12],
       [  1,   7,  40, 164,  57],
       [  0,   0,   2,  18,  23]])
```

In [45]:
```
label=['1','2','3','4','5']
cmd=ConfusionMatrixDisplay(mat,display_labels=label)
cmd.plot()
```

Out[45]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fc9e0a
59820>
```



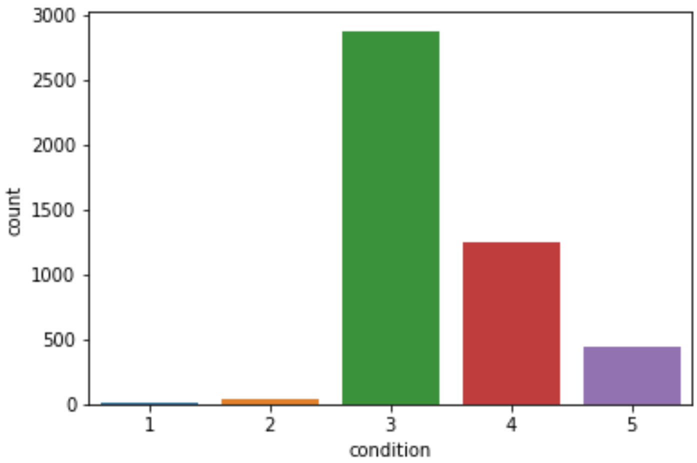
In [52]:
```
score=accuracy_score(y_pred,y_test)
score
```

Out[52]:
```
0.7956521739130434
```

In [54]:
```
import seaborn as sns
sns.countplot('condition',data=df1)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9e2bf9b80>
```



In [53]:
```
report=classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         7
           3       0.90      0.93      0.91       587
           4       0.61      0.70      0.65       233
           5       0.53      0.25      0.34        92

    accuracy                           0.80       920
   macro avg       0.41      0.38      0.38       920
weighted avg       0.78      0.80      0.78       920


/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1
318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1
318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1
318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```