In [82]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [83]:

```
df = pd.read_csv('yds_data.csv', index_col = 0)
```

In [84]:

```
df.head()
```

Out[84]:

	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout_match	game_season	remaining_sec	dista
0	10.0	167.0	72.0	10.0	1.0	0.0	2000-01	27.0	
1	12.0	-157.0	0.0	10.0	1.0	0.0	2000-01	22.0	
2	35.0	-101.0	135.0	7.0	1.0	0.0	2000-01	45.0	
3	43.0	138.0	175.0	6.0	1.0	0.0	2000-01	52.0	
4	155.0	0.0	0.0	NaN	2.0	0.0	2000-01	19.0	

In [85]:

```
df.shape
```

Out[85]:

```
(10066, 27)
```

In [86]:

```
for col in df.columns:
  print(str(col) + " " + " " + str(df[col].nunique()))
```

```
match_event_id  570
location_x  390
location_y  320
remaining_min  12
power_of_shot  6
knockout_match  1
game_season  7
remaining_sec  60
distance_of_shot  63
is_goal  2
area_of_shot  6
shot_basics  7
range_of_shot  5
team_name  1
date_of_game  460
home/away  68
```

```
shot_id_number  9531
lat/lng  35
type_of_shot  57
type_of_combined_shot  5
match_id  460
team_id  1
remaining_min.1  249
power_of_shot.1  175
knockout_match.1  344
remaining_sec.1  314
distance_of_shot.1  254
```

From the first look, we can see that lots of variables can be removed since it contains only one unique value of un-necessary variables.

In [87]:

```
col_drop = ['team_name', 'team_id', 'knockout_match.1', 'power_of_shot.1', 'remaining_min
.1', 'remaining_sec.1', 'distance_of_shot.1', 'match_event_id']
df.drop(col_drop, axis = 1, inplace = True)
```

In [88]:

```
df.head()
```

Out[88]:

	location_x	location_y	remaining_min	power_of_shot	knockout_match	game_season	remaining_sec	distance_of_shot	is_g
0	167.0	72.0	10.0	1.0	0.0	2000-01	27.0	38.0	N
1	-157.0	0.0	10.0	1.0	0.0	2000-01	22.0	35.0	
2	-101.0	135.0	7.0	1.0	0.0	2000-01	45.0	36.0	
3	138.0	175.0	6.0	1.0	0.0	2000-01	52.0	42.0	
4	0.0	0.0	NaN	2.0	0.0	2000-01	19.0	20.0	

Some of the extra columns is also removed which does not fit for the data.

In [89]:

```
col_drop2 = ['area_of_shot', 'date_of_game', 'game_season', 'shot_basics', 'match_id', 's
hot_id_number']
df.drop(col_drop2, axis = 1, inplace=True)
```

In [90]:

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

Out[90]:

```
location_x                 452
location_y                 481
remaining_min              488
power_of_shot              475
knockout_match             515
remaining_sec              540
distance_of_shot           511
is_goal                   2066
```

```
range_of_shot              533
home/away                  507
lat/lng                    512
type_of_shot              5043
type_of_combined_shot     5024
dtype: int64
```

In [91]:

```
df.dtypes
```

Out[91]:

```
location_x                float64
location_y                float64
remaining_min             float64
power_of_shot             float64
knockout_match            float64
remaining_sec             float64
distance_of_shot          float64
is_goal                   float64
range_of_shot              object
home/away                  object
lat/lng                    object
type_of_shot               object
type_of_combined_shot      object
dtype: object
```

In [92]:

```
df['range_of_shot'].value_counts()
```

Out[92]:

```
Less Than 8 ft.     3049
16-24 ft.           2781
8-16 ft.            1954
24+ ft.             1715
Back Court Shot       34
Name: range_of_shot, dtype: int64
```

Taking care of null values

Replacing all the null values in float64 datatypes by mean.

In [93]:

```
columns = ['location_x', 'location_y', 'remaining_min', 'power_of_shot', 'knockout_match
', 'remaining_sec', 'distance_of_shot']

for col in columns:
  df[col].fillna(df[col].mean(), inplace = True)
```

In [94]:

```
df.dtypes
```

Out[94]:

```
location_x                float64
location_y                float64
remaining_min             float64
power_of_shot             float64
knockout_match            float64
remaining_sec             float64
distance_of_shot          float64
is_goal                   float64
range_of_shot              object
home/away                  object
lat/lng                    object
type_of_shot               object
type_of_shot               object
type_of_combined_shot      object
dtype: object
```

In [95]:

```
df['lat/lng'].value_counts().index[0]
```

Out[95]:

```
'42.982923, -71.446094'
```

Latitude and Longitude are two separate entities. It has to be splitted accordingly.

In [96]:

```
df[['Lat','Long']] = df['lat/lng'].astype(str).str.split(',', expand=True).astype('float
64')
df.drop('lat/lng', axis = 1, inplace=True)
```

In [97]:

```
df['type_of_combined_shot'].value_counts()
```

Out[97]:

```
shot - 3    3783
shot - 4     955
shot - 1     252
shot - 5      34
shot - 2      18
Name: type_of_combined_shot, dtype: int64
```

Replacing 'type of combined shot' and 'type of shot' NaN varible to shot-NaN meaning no shot taken.

In [98]:

```
columns = ['type_of_shot', 'type_of_combined_shot']

for col in columns:
  df[col].replace(np.NaN, "shot-NaN", inplace=True)
```

In [99]:

```
df.head()
```

Out[99]:

	location_x	location_y	remaining_min	power_of_shot	knockout_match	remaining_sec	distance_of_shot	is_goal	range_of_
0	167.0	72.0	10.000000	1.0	0.0	27.0	38.0	NaN	16-2
1	-157.0	0.0	10.000000	1.0	0.0	22.0	35.0	0.0	8-1
2	-101.0	135.0	7.000000	1.0	0.0	45.0	36.0	1.0	16-2
3	138.0	175.0	6.000000	1.0	0.0	52.0	42.0	0.0	16-2
4	0.0	0.0	4.966277	2.0	0.0	19.0	20.0	1.0	Less Th

In [100]:

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

Out[100]:

```
location_x                    0
```

```
location_y                 0
remaining_min              0
power_of_shot              0
knockout_match             0
remaining_sec              0
distance_of_shot           0
is_goal                 2066
range_of_shot            533
home/away                507
type_of_shot               0
type_of_combined_shot      0
Lat                      512
Long                     512
dtype: int64
```

Replacing lat and long by mean

In [101]:

```
columns = ['Lat', 'Long']

for col in columns:
  df[col].fillna(df[col].mean(), inplace = True)
```

In [102]:

```
df['range_of_shot'].value_counts()
```

Out[102]:

```
Less Than 8 ft.    3049
16-24 ft.          2781
8-16 ft.           1954
24+ ft.            1715
Back Court Shot      34
Name: range_of_shot, dtype: int64
```

In [103]:

```
df[df['range_of_shot'].isnull()]
```

Out[103]:

	location_x	location_y	remaining_min	power_of_shot	knockout_match	remaining_sec	distance_of_shot	is_goal	rang
17	-117.0	226.000000	8.0	2.0	0.0	50.0	45.000000	1.0	
36	1.0	4.000000	4.0	1.0	0.0	9.0	20.000000	NaN	
37	-117.0	116.000000	5.0	2.0	0.0	33.0	36.000000	NaN	
46	-4.0	84.864267	2.0	3.0	0.0	55.0	33.070434	0.0	
49	-176.0	30.000000	3.0	4.0	0.0	19.0	37.000000	NaN	
...	...	...	...	...	...	...	...	...	
9969	0.0	0.000000	11.0	3.0	0.0	19.0	20.000000	1.0	
10010	-72.0	77.000000	6.0	4.0	0.0	0.0	30.000000	NaN	
10015	146.0	84.864267	6.0	1.0	0.0	35.0	39.000000	0.0	
10020	125.0	-13.000000	1.0	1.0	0.0	35.0	32.000000	NaN	
10042	212.0	135.000000	2.0	1.0	0.0	54.0	45.000000	0.0	

	location_x	location_y	remaining_min	power_of_shot	knockout_match	remaining_sec	distance_of_shot	is_goal	rang

533 rows × 14 columns

In [104]:

```
df = pd.concat([df, pd.get_dummies(df['range_of_shot'])], axis=1)
```

"get_dummies" creates a categorical variable for each of the values in the respected column. This pre-processing will be helpful for ML .

In [105]:

```
df.drop('range_of_shot', axis = 1)
```

Out[105]:

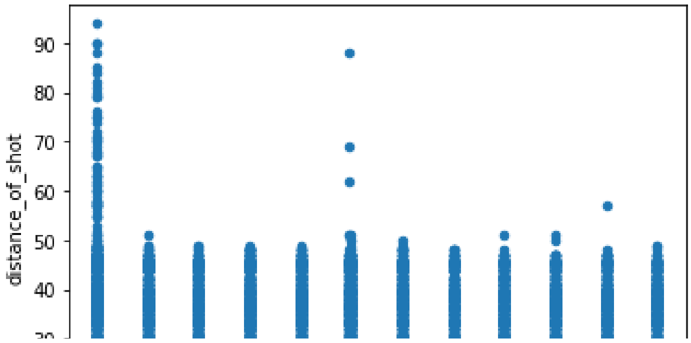
	location_x	location_y	remaining_min	power_of_shot	knockout_match	remaining_sec	distance_of_shot	is_goal	home
0	167.0	72.0	10.000000	1.0	0.0	27.0	38.0	NaN	MA
1	-157.0	0.0	10.000000	1.0	0.0	22.0	35.0	0.0	MA
2	-101.0	135.0	7.000000	1.0	0.0	45.0	36.0	1.0	
3	138.0	175.0	6.000000	1.0	0.0	52.0	42.0	0.0	MA
4	0.0	0.0	4.966277	2.0	0.0	19.0	20.0	1.0	MA
...	...	...	...	...	...	...	...	...	
10061	-79.0	141.0	8.000000	2.0	0.0	25.0	36.0	1.0	MA
10062	167.0	10.0	7.000000	2.0	0.0	54.0	36.0	0.0	MA
10063	167.0	194.0	5.000000	2.0	0.0	1.0	45.0	1.0	MA
10064	-29.0	166.0	3.000000	2.0	0.0	45.0	36.0	1.0	MA
10065	144.0	125.0	3.000000	2.0	0.0	26.0	39.0	NaN	MA

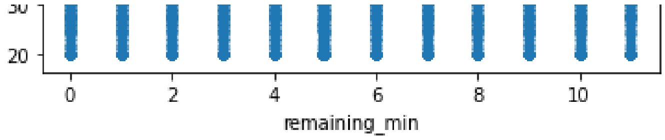
10066 rows × 18 columns

Relation between Remaining minutes and distance of shot.

In [109]:

```
df.plot.scatter('remaining_min', 'distance_of_shot')
plt.show()
```





The distance of shot is more in 0th remaining_min.

Conclusion

1. **The unnecessary columns have been deleted.**
2. **Few of the column null values have been addressed.**