```
from google.colab import drive

# mount the drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
```

```
# read file from the drive
import pandas as pd

datasetPath = '/content/drive/MyDrive/matches_1930_2022.csv'

df = pd.read_csv(datasetPath)
```

```
df
```

	home_team	away_team	home_score	home_xg	home_penalty	away_score	awa
0	Argentina	France	3	3.3	4.0	3	
1	Croatia	Morocco	2	0.7	NaN	1	
2	France	Morocco	2	2.0	NaN	0	
3	Argentina	Croatia	3	2.3	NaN	0	
4	Morocco	Portugal	1	1.4	NaN	0	
...	...	...	...	...	...	...	
959	Argentina	France	1	NaN	NaN	0	
960	Yugoslavia	Brazil	2	NaN	NaN	1	
961	Romania	Peru	3	NaN	NaN	1	
962	United States	Belgium	3	NaN	NaN	0	
963	France	Mexico	4	NaN	NaN	1	

964 rows × 44 columns

```
df.columns
```

```
Index(['home_team', 'away_team', 'home_score', 'home_xg', 'home_penalty',
       'away_score', 'away_xg', 'away_penalty', 'home_manager',
'home_captain',
       'away_manager', 'away_captain', 'Attendance', 'Venue', 'Officials',
       'Round', 'Date', 'Score', 'Referee', 'Notes', 'Host', 'Year',
       'home_goal', 'away_goal', 'home_goal_long', 'away_goal_long',
       'home_own_goal', 'away_own_goal', 'home_penalty_goal',
       'away_penalty_goal', 'home_penalty_miss_long',
'away_penalty_miss_long',
       'home_penalty_shootout_goal_long',
'away_penalty_shootout_goal_long',
       'home_penalty_shootout_miss_long',
'away_penalty_shootout_miss_long',
       'home_red_card', 'away_red_card', 'home_yellow_red_card',
       'away_yellow_red_card', 'home_yellow_card_long',
       'away_yellow_card_long', 'home_substitute_in_long',
       'away_substitute_in_long'],
      dtype='object')
```

```
df['home_team'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 964 entries, 0 to 963
Series name: home_team
Non-Null Count  Dtype
--------------  -----
964 non-null    object
dtypes: object(1)
memory usage: 7.7+ KB
```

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

Show hidden output

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

Show hidden output

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

Show hidden output

```
ScoredAboveZero = df['Score'] > '0-0'
```

```
df[ScoredAboveZero]
```

```
(929, 44)
```

```
ScoredByBrazil = df['home_team'] == 'Brazil'
```

```
df[ScoredAboveZero & ScoredByBrazil]
```

```
brazildf = df[ScoredAboveZero & ScoredByBrazil].sort_values(by = 'home_score',a
```

```
brazildf
```

Show hidden output

```
brazildf.info()
```

Show hidden output

```
%matplotlib inline
```

```
from matplotlib import pyplot as plt
```

```
brazildf.plot(x='home_score',y='away_score',kind='scatter')
plt.xlabel('Home Score')
plt.ylabel('Away Score')
plt.legend('Takeaway')
plt.title('Brazil Matches')
plt.show()
```

Show hidden output

```
brazildf.plot(x='Year',y='home_score',kind='hexbin')
plt.xlabel('Year')
plt.ylabel('Home Score')
plt.legend('Takeaway')
plt.title('Brazil Matches')
plt.show()
```

Show hidden output

```
df
```

Show hidden output

```
def AnalyzeFrance():
  matchesPlayed = df['home_team'].value_counts()['France']
  print(f'Total Number of matches played by France is : {matchesPlayed}')
```

```
AnalyzeFrance()
```

```
Total Number of matches played by France is : 38
```

```
def AnalyzeTowns():
  TownData = df['home_team'].value_counts()
  for _ in TownData.index:
    print(f'Total Number of matches played by {_} is : {TownData[_]}')

AnalyzeTowns()
```

Show hidden output

```
%matplotlib inline
```

```
Town30 = df['home_team'].value_counts().values > 30
```

```
Town30
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False])
```

```
townName = []
matchesPlayed = []
TownData = df['home_team'].value_counts()
for _ in TownData.index:
  if TownData[_] > 30 :
    townName.append(_)
    matchesPlayed.append(TownData[_])
```

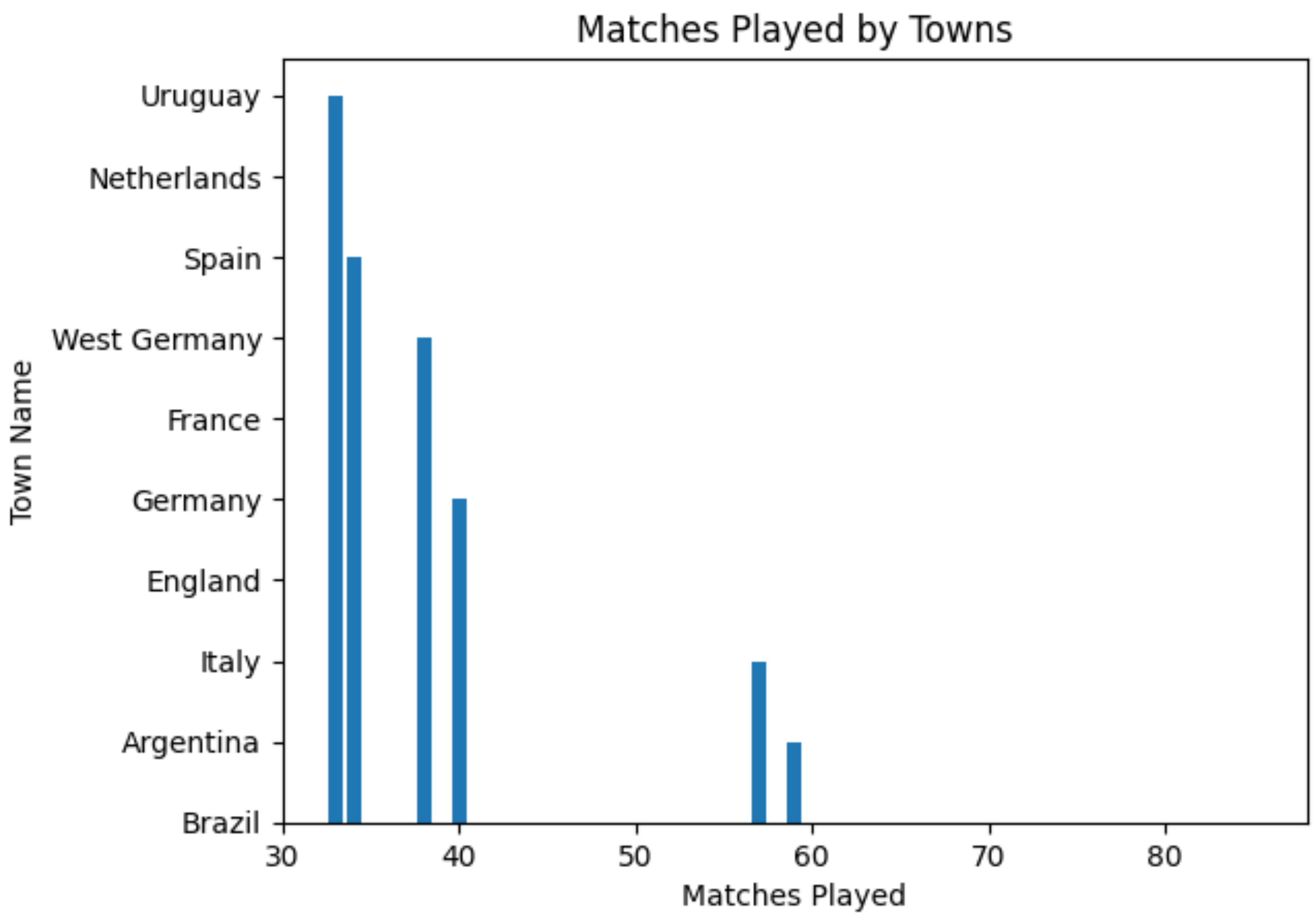
```
from matplotlib import pyplot as plt
```

```
plt.bar(matchesPlayed,townName)
plt.ylabel('Town Name')
plt.xlabel('Matches Played')
plt.title('Matches Played by Towns')
plt.show()
```

Show hidden output

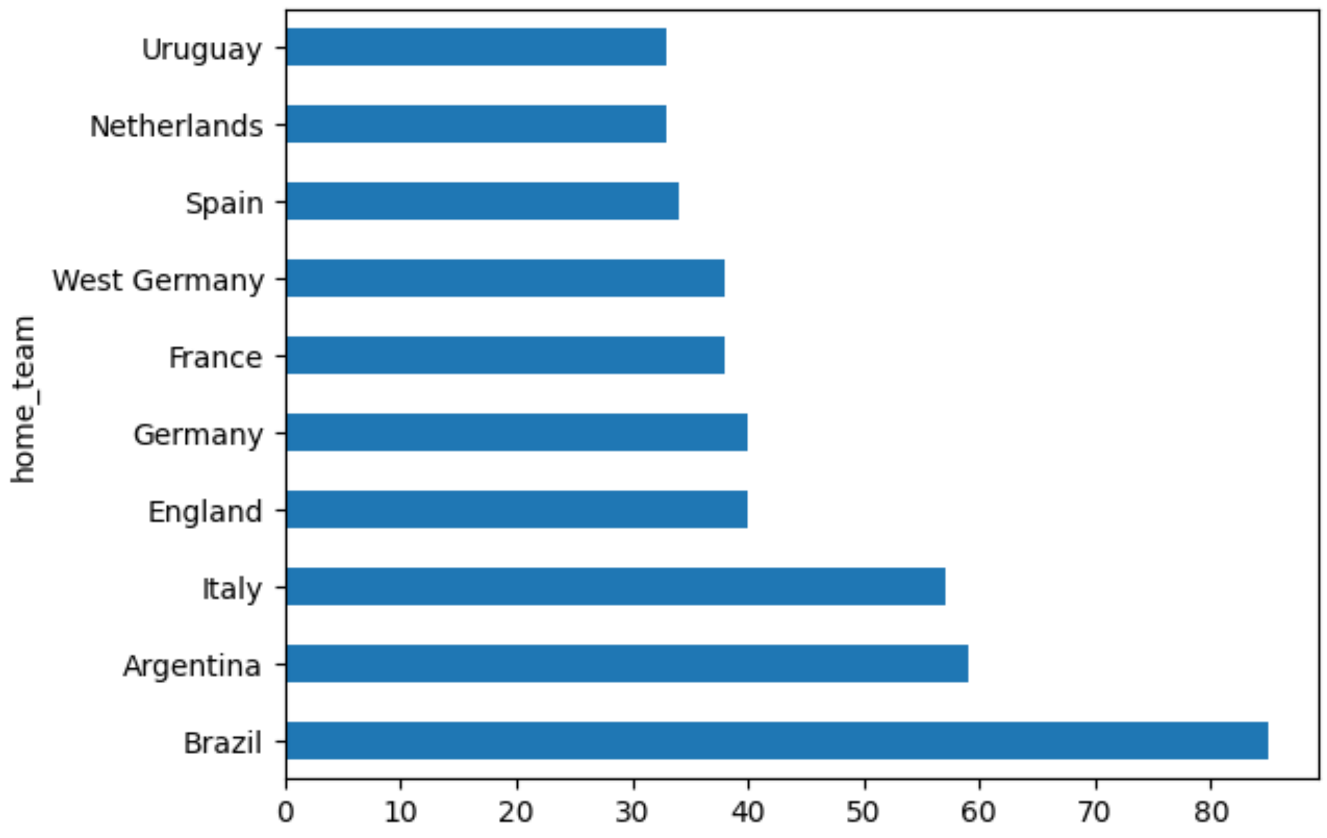
```
def MatchesPlayedByTown():
  townName = []
  matchesPlayed = []
  TownData = df['home_team'].value_counts()
  for _ in TownData.index:
    if TownData[_] > 30 :
      townName.append(_)
      matchesPlayed.append(TownData[_])
  plt.bar(matchesPlayed,townName)
  plt.ylabel('Town Name')
  plt.xlabel('Matches Played')
  plt.title('Matches Played by Towns')
  plt.show()

MatchesPlayedByTown()
```



```
def MatchesPlayedByTown2():
  Above30 = df['home_team'].value_counts()
  filteredData =  Above30[Above30> 30]
  filteredData.plot(kind = 'barh')

MatchesPlayedByTown2()
```



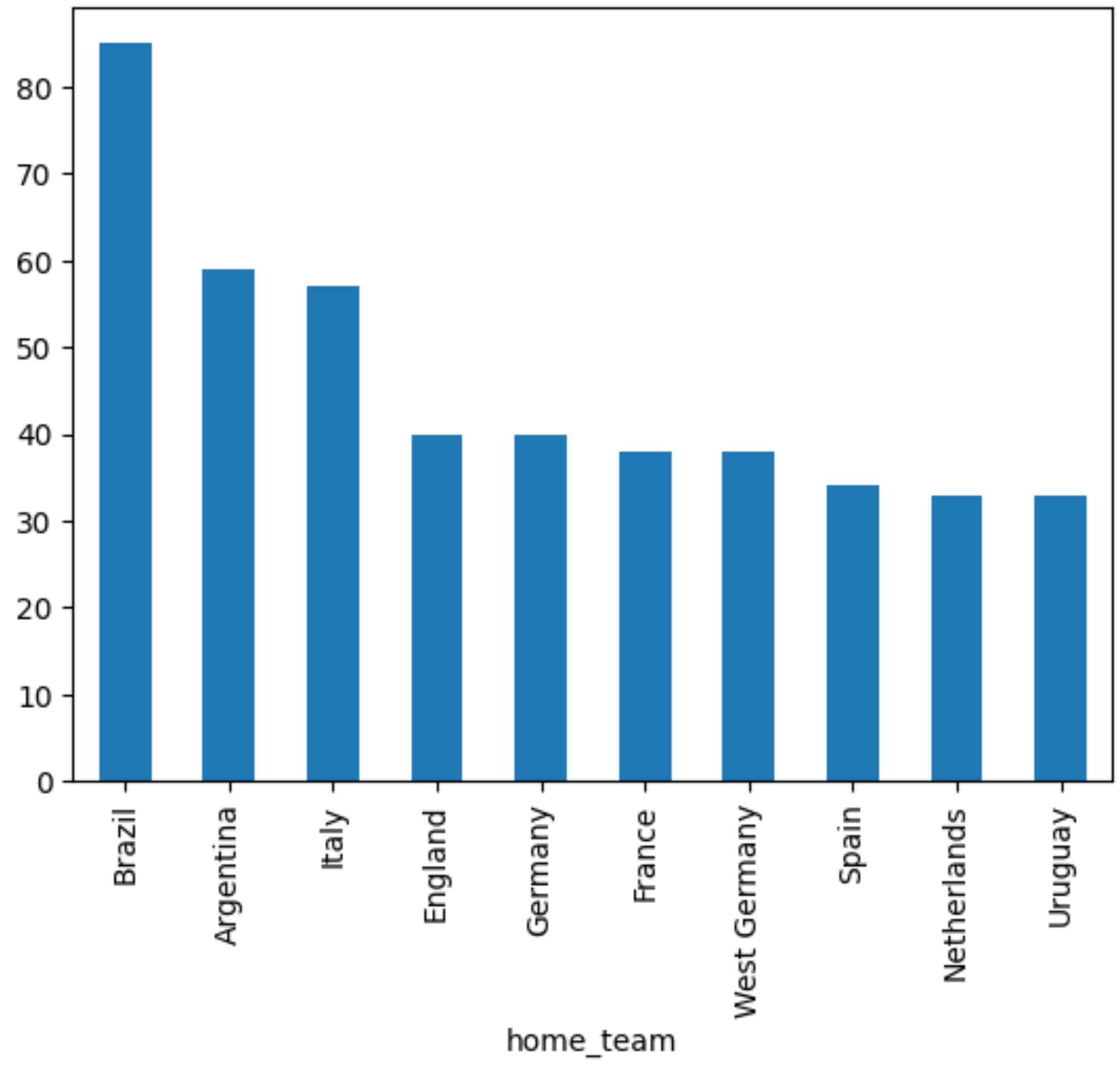
Start coding or generate with AI.

```
Above30 = df['home_team'].value_counts()
```

```
filteredData =  Above30[Above30> 30]
```

```
filteredData.plot(kind = 'bar')
```

```
<Axes: xlabel='home_team'>
```



```
df
```

	home_team	away_team	home_score	home_xg	home_penalty	away_score	awa
0	Argentina	France	3	3.3	4.0	3	
1	Croatia	Morocco	2	0.7	NaN	1	
2	France	Morocco	2	2.0	NaN	0	
3	Argentina	Croatia	3	2.3	NaN	0	
4	Morocco	Portugal	1	1.4	NaN	0	
...	...	...	...	...	...	...	
959	Argentina	France	1	NaN	NaN	0	
960	Yugoslavia	Brazil	2	NaN	NaN	1	
961	Romania	Peru	3	NaN	NaN	1	
962	United States	Belgium	3	NaN	NaN	0	
963	France	Mexico	4	NaN	NaN	1	

964 rows × 44 columns

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

home_team	count
Brazil	85
Argentina	59
Italy	57
England	40
Germany	40
...	...
Jamaica	1
FR Yugoslavia	1
Angola	1
Trinidad and Tobago	1
Bosnia and Herzegovina	1

82 rows × 1 columns

dtype: int64

```
filteredData = df['home_team'].value_counts()
```

```
f2 = filteredData[filteredData >= 40].index
```

```
f2
```

```
Index(['Brazil', 'Argentina', 'Italy', 'England', 'Germany'],
dtype='object', name='home_team')
```

```
df2 = df[df["home_team"].isin(f2)]
```

```
df2
```

Show hidden output

```
df2["home_xg"].value_counts()
```

Show hidden output

```
%matplotlib inline
```

```
df2.plot(x = 'home_team',y = 'home_xg',kind = 'bar')
```

Show hidden output

```
f3 = df2["home_xg"] > 2.5
```

```
df3 = df2[f3]
```

```
df3
```

	home_team	away_team	home_score	home_xg	home_penalty	away_score	awa
0	Argentina	France	3	3.3	4.0	3	
11	Brazil	Korea Republic	4	3.6	NaN	1	
53	Germany	Japan	1	3.1	NaN	2	
71	Brazil	Belgium	1	2.8	NaN	2	
74	Brazil	Mexico	2	2.7	NaN	0	
96	England	Panama	6	2.8	NaN	1	
102	Brazil	Costa Rica	2	2.6	NaN	0	

7 rows × 44 columns

Start coding or generate with AI.

```
from matplotlib import pyplot as plt
```

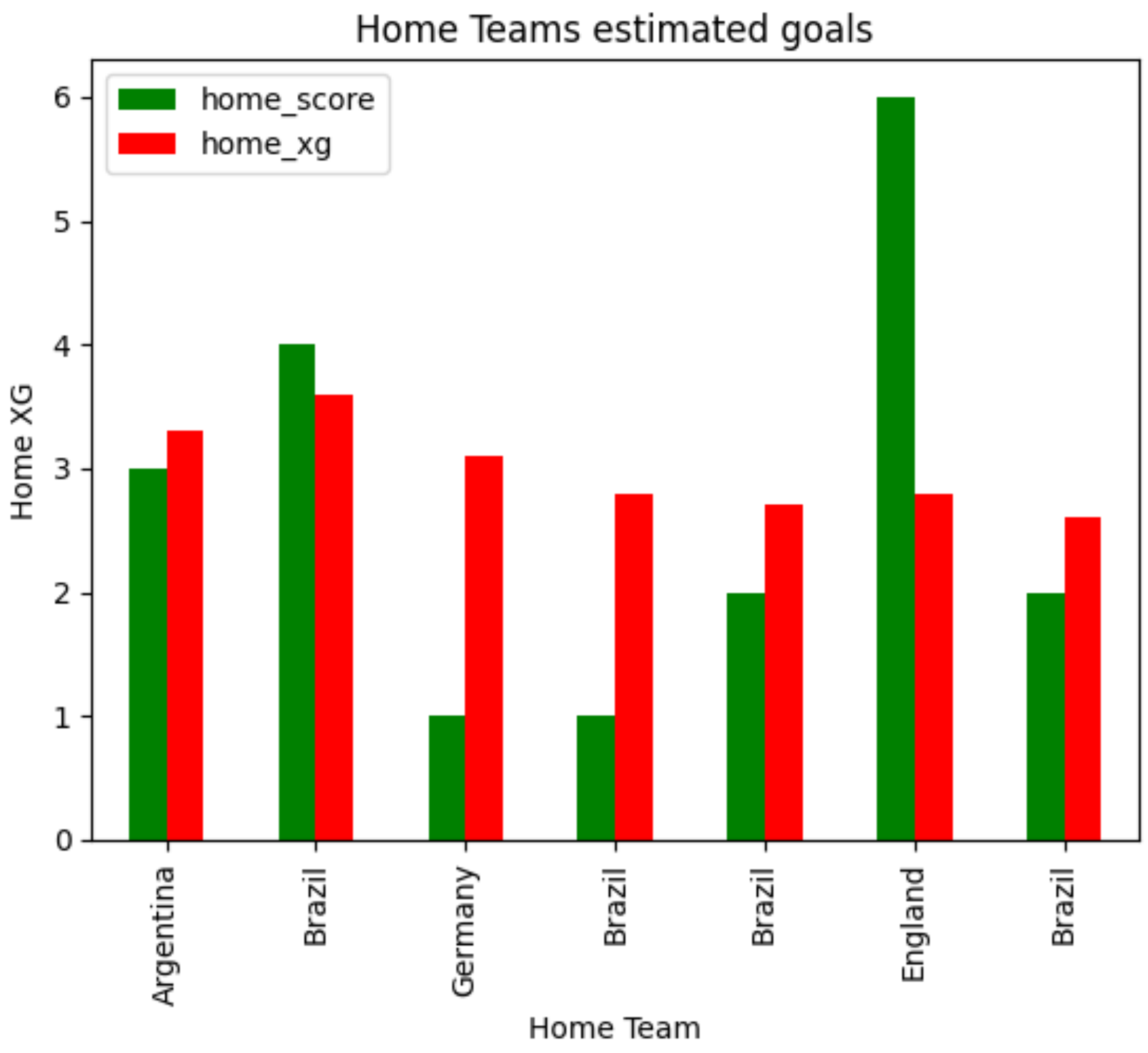
```
df3.plot(x="home_team",y="home_xg",kind="bar")
plt.xlabel("Home Team")
plt.ylabel("Home XG")
plt.title("Home Teams estimated goals")
plt.show()
```

Show hidden output

```
df3
```

Show hidden output

```
df3.plot(x="home_team",y= ['home_score','home_xg'],kind="bar",color=["green","r
plt.xlabel("Home Team")
plt.ylabel("Home XG")
plt.title("Home Teams estimated goals")
plt.show()
```



```
# groupb
df.groupby("home_team")
```

Show hidden output

```
type(df4)
```

```
pandas.core.groupby.generic.DataFrameGroupBy
def __init__(obj: NDFrameT, keys: _KeysArgType | None=None, axis:
Axis=0, level: IndexLabel | None=None, grouper: ops.BaseGrouper |
None=None, exclusions: frozenset[Hashable] | None=None, selection:
IndexLabel | None=None, as_index: bool=True, sort: bool=True,
group_keys: bool=True, observed: bool | lib.NoDefault=lib.no_default,
dropna: bool=True) -> None

Class for grouping and aggregating relational data.

See aggregate, transform, and apply functions on this object.

It's easiest to use obj.groupby(...) to use GroupBy, but you can also do:
```

```
df4 = df.groupby("home_team")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/pandas/core/groupby/groupby.py in
_agg_py_fallback(self, how, values, ndim, alt)
   1941         try:
-> 1942             res_values = self._grouper.agg_series(ser, alt,
preserve_dtype=True)
   1943         except Exception as err:

                              17 frames

TypeError: Could not convert string 'RussiaSloveniaSpainNorthern
IrelandChileAustria' to numeric

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/pandas/core/groupby/groupby.py in
_agg_py_fallback(self, how, values, ndim, alt)
   1944             msg = f"agg function failed [how->{how},dtype->
{ser.dtype}]"
   1945             # preserve the kind of exception that raised
-> 1946             raise type(err)(msg) from err
   1947
```

```
df.groupby("home_team")["home_score"].mean()
```

Show hidden output

```
df.describe()
```

Show hidden output

```
dg = df.groupby("home_team")["home_xg"]
```

```
print(dg.mean())
```

```
home_team
Algeria               NaN
Angola                NaN
Argentina        1.842857
Australia        0.750000
Austria               NaN
                   ...
Uruguay          1.240000
Wales            0.600000
West Germany          NaN
Yugoslavia            NaN
Zaire                 NaN
Name: home_xg, Length: 82, dtype: float64
```

```
das = pd.DataFrame({
    'Branch':['CSE','ECE','CSE','BBA','BBA'],
    'Name':['virat','suresh','prasad','priya','leela'],
    'Salary':[10000,20000,15000,15000,13000],
})
```

```
das
```

	Branch	Name	Salary
0	CSE	virat	10000
1	ECE	suresh	20000
2	CSE	prasad	15000
3	BBA	priya	15000
4	BBA	leela	13000

```
sv = das.groupby('Branch')['Salary']
```

```
das.describe()
```

```
            Salary
count     5.000000
mean  14600.000000
std    3646.916506
min   10000.000000
25%   13000.000000
50%   15000.000000
75%   15000.000000
max   20000.000000
```

```
sv.agg(['mean','min'])
```

```
           mean    min
Branch
BBA     14000.0  13000
CSE     12500.0  10000
ECE     20000.0  20000
```

```
sv2= das.groupby(['Branch','Name'])['Salary']
```

```
sv2.max()
```

```
                 Salary
Branch  Name
BBA     leela     13000
        priya     15000
CSE     prasad    15000
        virat     10000
ECE     suresh    20000

dtype: int64
```