In [2]:
```
import pandas as pd
import numpy as np
import datetime as dt
from datetime import date
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
```

In [3]:
```
pd.set_option("display.max_columns",100)
```

In [4]:
```
df=pd.read_csv(r"C:\Users\berid\OneDrive\Desktop\mydata\fifa\players_22.csv")
```

```
C:\Users\berid\AppData\Local\Programs\Python\Python39\lib\site-packages\IPython\core\interactiveshell.py:3251: DtypeWarning: Columns (25,108) have mixed types.Specify dtype option
on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [5]:
```
df["dob"]=pd.to_datetime(df["dob"])
```

In [6]:
```
df.head(5)
```

Out[6]:

	sofifa_id	player_url	short_name	long_name	player_positions	overall	potential	value_eur	wage_eur	age	dob	height_cm	weight_kg	club_team_id	club_name	league_name	league_lev
0	158023	https://sofifa.com/player/158023/lionel-messi/...	L. Messi	Lionel Andrés Messi Cuccittini	RW, ST, CF	93	93	78000000.0	320000.0	34	1987-06-24	170	72	73.0	Paris Saint-Germain	French Ligue 1	1
1	188545	https://sofifa.com/player/188545/robert-lewand...	R. Lewandowski	Robert Lewandowski	ST	92	92	119500000.0	270000.0	32	1988-08-21	185	81	21.0	FC Bayern München	German 1. Bundesliga	1
2	20801	https://sofifa.com/player/20801/c-ronaldo-dos-...	Cristiano Ronaldo	Cristiano Ronaldo dos Santos Aveiro	ST, LW	91	91	45000000.0	270000.0	36	1985-02-05	187	83	11.0	Manchester United	English Premier League	1
3	190871	https://sofifa.com/player/190871/neymar-da-sil...	Neymar Jr	Neymar da Silva Santos Júnior	LW, CAM	91	91	129000000.0	270000.0	29	1992-02-05	175	68	73.0	Paris Saint-Germain	French Ligue 1	1
4	192985	https://sofifa.com/player/192985/kevin-de-bruy...	K. De Bruyne	Kevin De Bruyne	CM, CAM	91	91	125500000.0	350000.0	30	1991-06-28	181	70	10.0	Manchester City	English Premier League	1

5 rows × 110 columns

find number of leagues and teams

In [7]:
```
df.club_name.nunique()
```

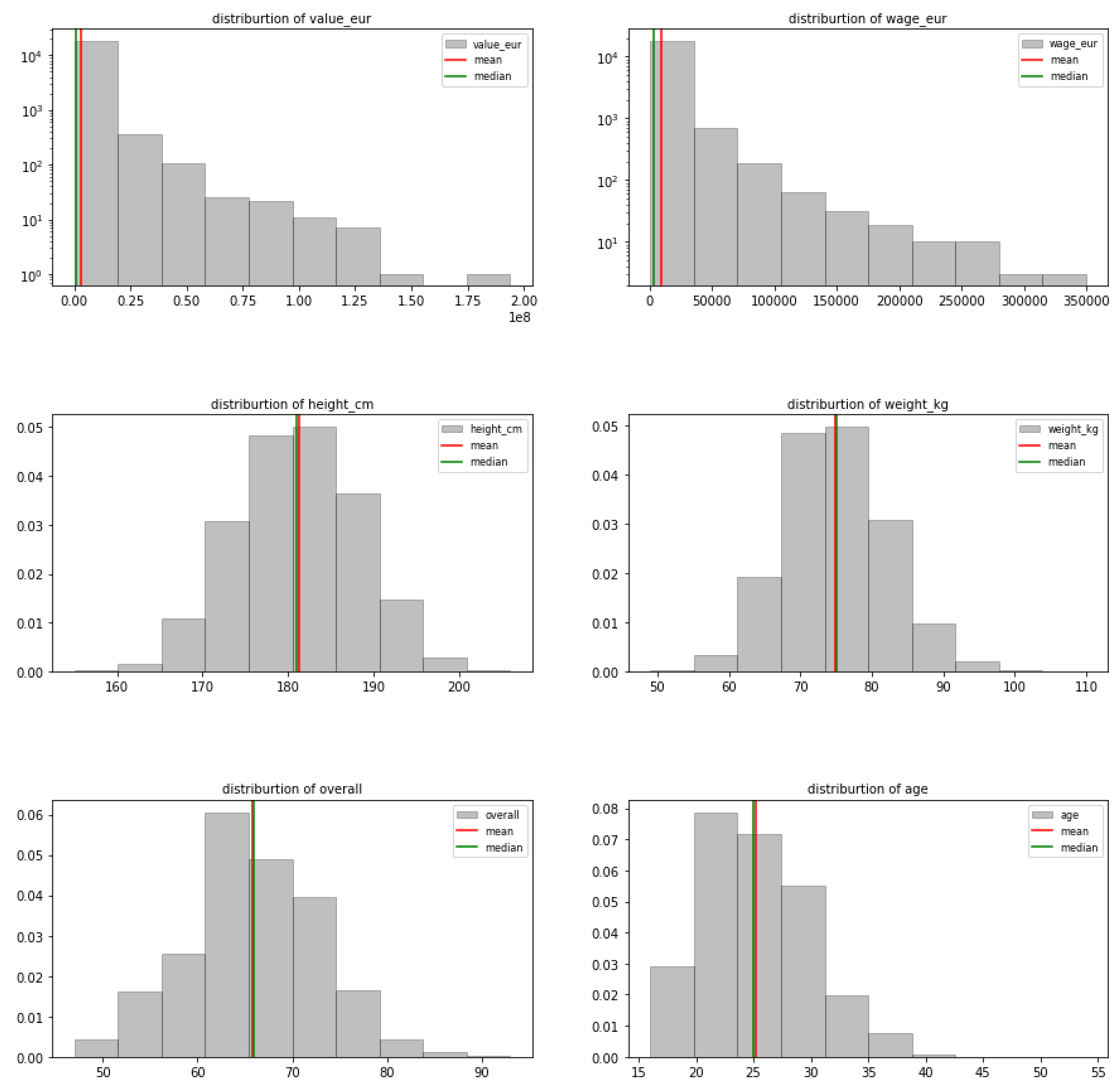
Out[7]:
```
701
```

In [8]:
```
df.league_name.nunique()
```

Out[8]:
```
55
```

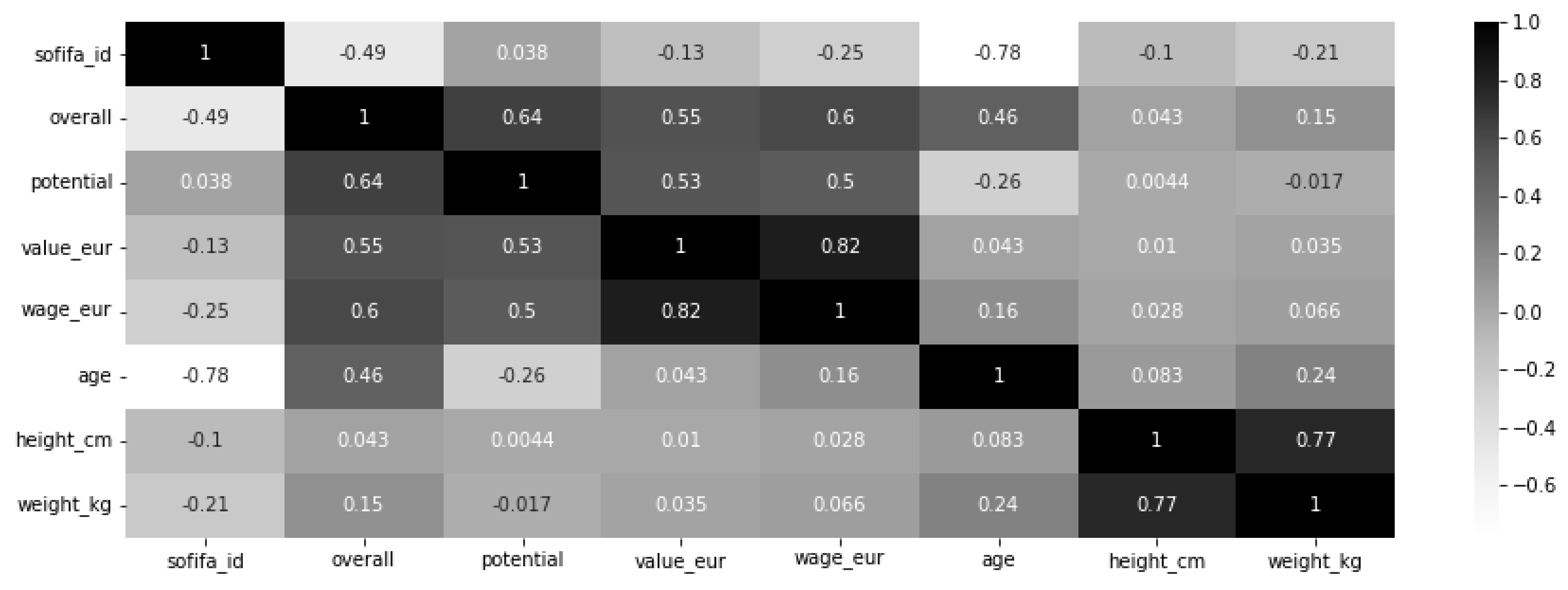
plot the distribution of player value, wage, height, weight, overall rating,age

In [9]:
```python
fig,axes=plt.subplots(3,2,figsize=(15,15))
columns=["value_eur","wage_eur","height_cm","weight_kg","overall","age"]
for ax, col in zip(axes.ravel(),columns):
    if col in ["value_eur","wage_eur"]:
        ax.hist(df[col],ec="k",log=True,color="k",alpha=0.25,label=col)
        ax.axvline(df[col].mean(),label="mean",color="r")
        ax.axvline(df[col].median(),label="median",color="g")
        ax.legend(fontsize=8)
    else:
        ax.hist(df[col],ec="k",density=True,color="k",alpha=0.25,label=col)
        ax.axvline(df[col].mean(),label="mean",color="r")
        ax.axvline(df[col].median(),label="median",color="g")
        ax.legend(fontsize=8)
    ax.set_title("distriburtion of "+col,size=10)
    plt.subplots_adjust(hspace=0.5)
plt.show()
```



find if there is any correlation between columns

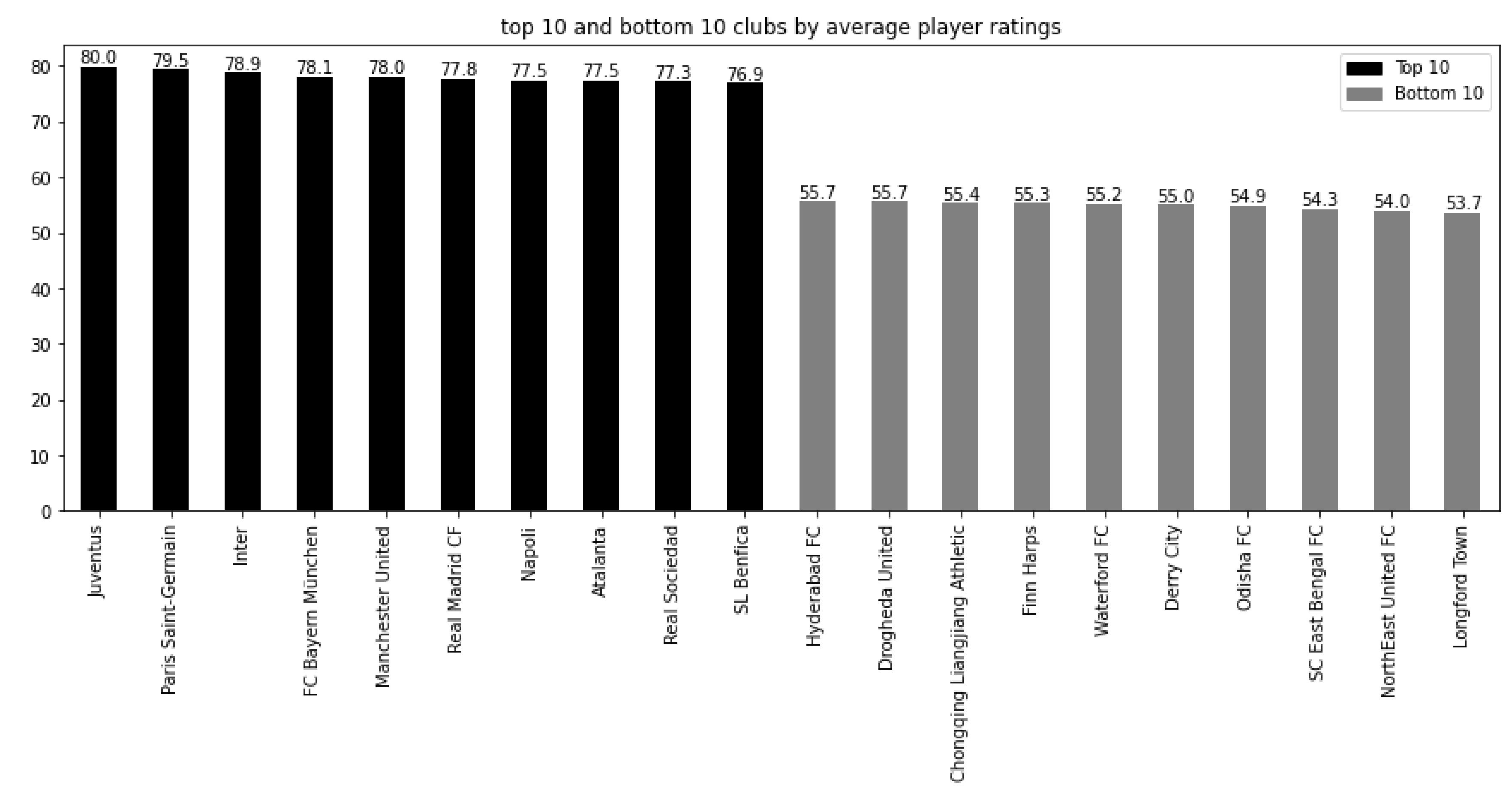
In [10]:
```python
plt.figure(figsize=(15,5))
sns.heatmap(df.iloc[:,:13].corr(),annot=True,cmap="Greys")
plt.show()
```



plot top 10 clubs with the highest overall rating

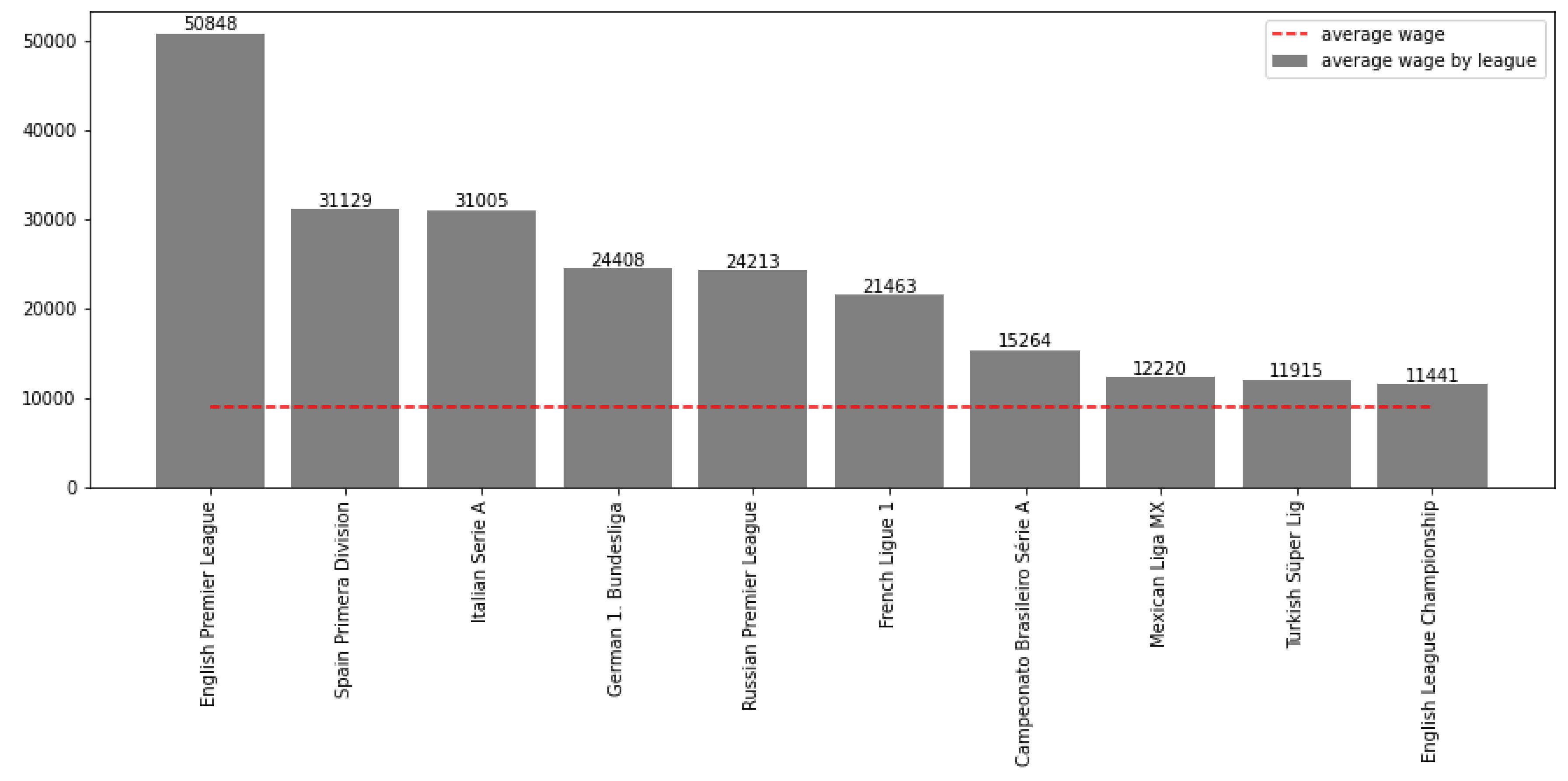
In [11]:
```
grouped_top=df.groupby("club_name")["overall"].mean().reset_index().sort_values("overall",ascending=False).head(10)
grouped_bottom=df.groupby("club_name")["overall"].mean().reset_index().sort_values("overall",ascending=False).tail(10)
unioned=pd.concat([grouped_top,grouped_bottom])
ax=unioned.plot(kind="bar",x="club_name",y="overall",xlabel="",figsize=(15,5),color=[*['k']*len(grouped_top.club_name), *['grey']*len(grouped_bottom.club_name)])
plt.title("top 10 and bottom 10 clubs by average player ratings")
legends = [
    matplotlib.patches.Patch(color="k", label="Top 10"),
    matplotlib.patches.Patch(color="grey", label="Bottom 10"),
]
ax.legend(handles=legends, prop={"size": 10})
def value_labels(x,y):
    for i in range(len(x)):
        plt.text(i, round(y.iloc[i],1),round(y.iloc[i],1), size=10, ha="center",va="bottom")
value_labels(unioned.club_name,unioned.overall)

plt.show()
```



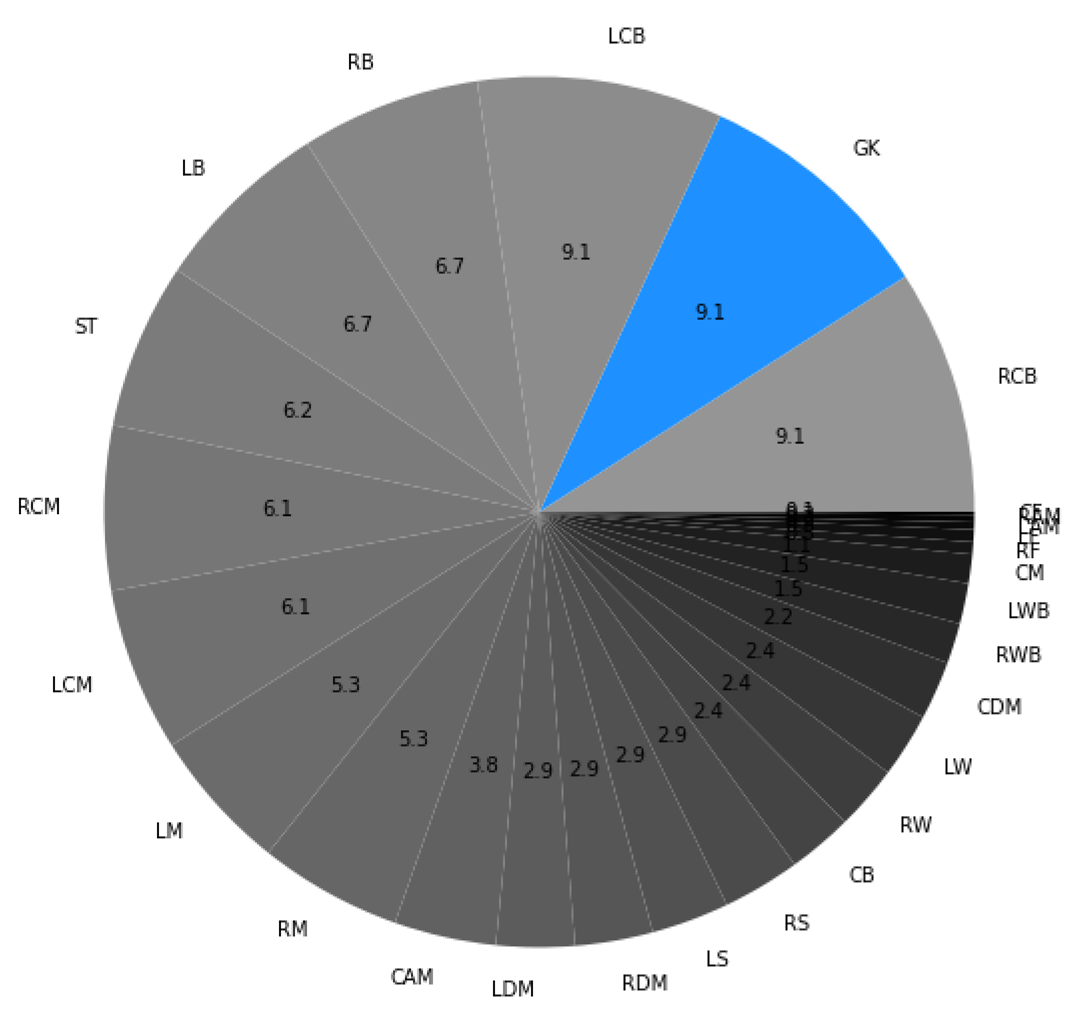
find 10 leagues with highest footballer salaries

In [12]:
```
grouped=df.groupby("league_name")["wage_eur"].mean().reset_index().sort_values("wage_eur",ascending=False).head(10)
grouped["average_salary"]=df.wage_eur.mean()
plt.figure(figsize=(15,5))
plt.bar(grouped.league_name,grouped.wage_eur,color="k",alpha=.5,label="average wage by league")
plt.plot(grouped.league_name,grouped.average_salary,color="r",ls="--",label="average wage")
plt.legend()
plt.xticks(rotation=90)
def value_counts(y):
    for i in range(len(y)):
        plt.text(i,round(y.iloc[i]),round(y.iloc[i]),size=10,ha="center",va="bottom")
value_counts(grouped.wage_eur)
plt.show()
```



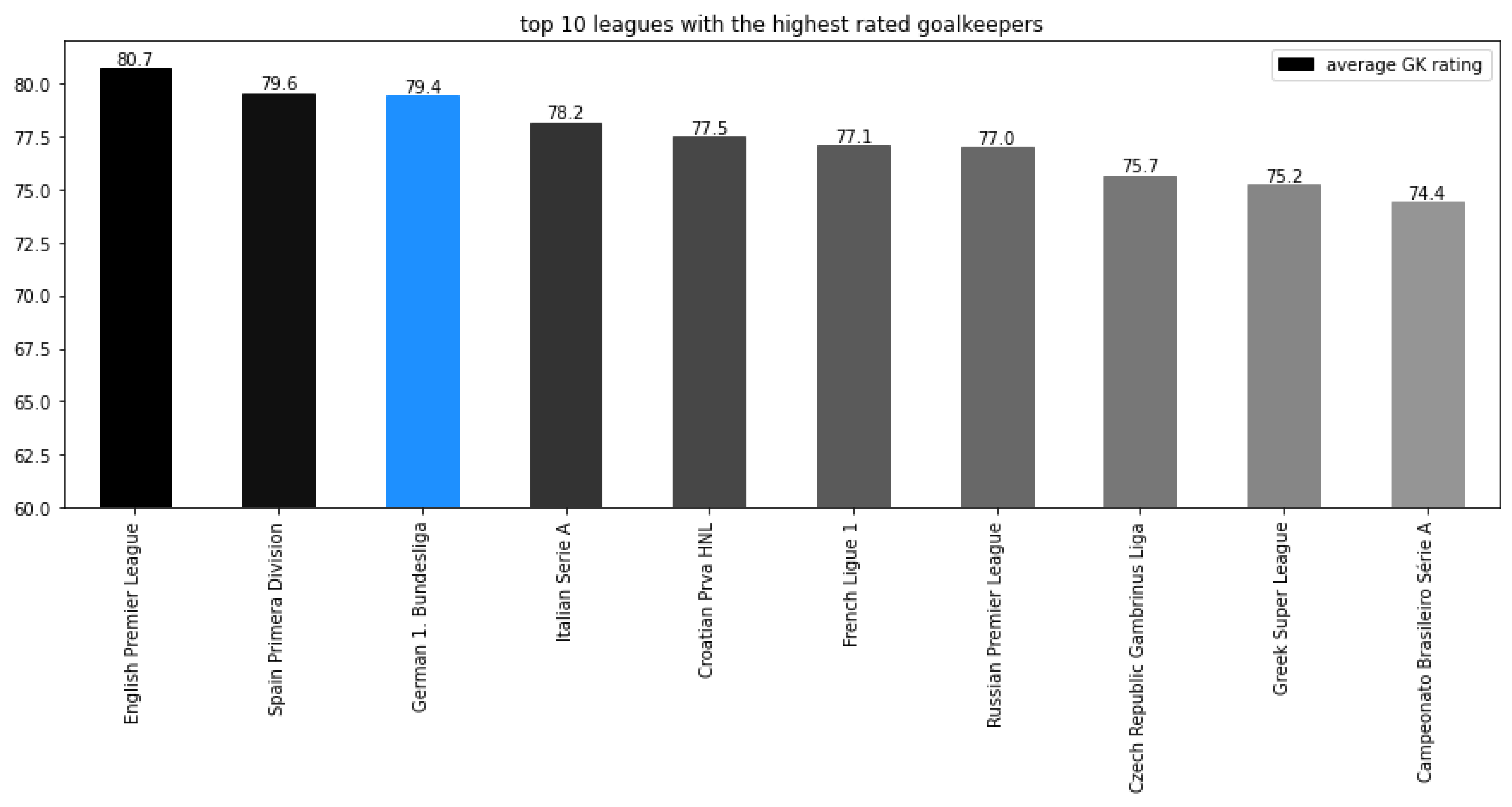
most common positions

In [13]:
```
values=df.club_position.value_counts(sort=True).reset_index()[2:]
cmap=plt.get_cmap("Greys")
colors=list(cmap(np.linspace(0.5,1,len(values["index"]))))
colors[1]="dodgerblue"
plt.figure(figsize=(10,10))
plt.pie(values.club_position,labels=values["index"],autopct="%1.1f",colors=colors)
plt.show()
```



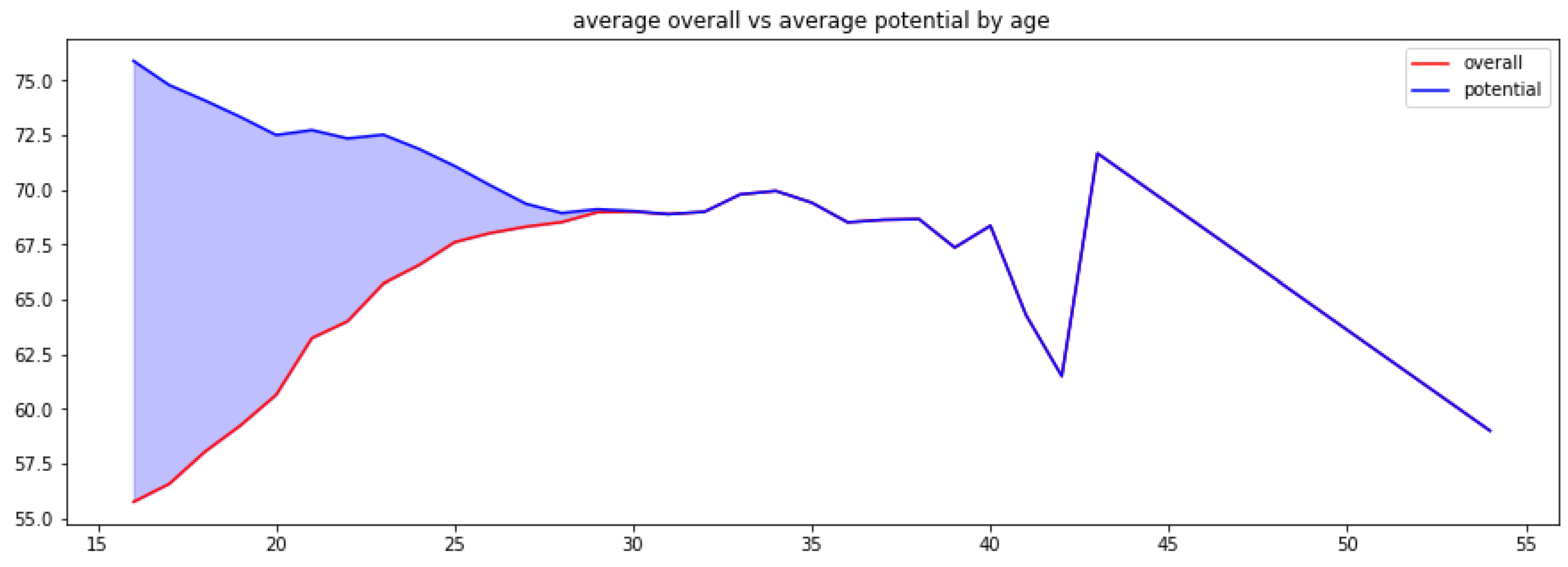
plot top 10 league with the highest rated goalkeepers

In [14]:
```
grouped=df.groupby("league_name").apply(lambda x:x[x["club_position"]=="GK"]["overall"].mean()).reset_index(name="overall").sort_values("overall",ascending=False).head(10)
cmap=plt.get_cmap("Greys")
colors=list(cmap(np.linspace(1,0.5,len(grouped["league_name"]))))
colors[2]="dodgerblue"
ax=grouped.plot(kind="bar",figsize=(15,5),title="top 10 leagues with the highest rated goalkeepers",\
             x="league_name",xlabel="",legend=True,color=colors)
def value_labels(y):
    for i in range(len(y)):
        plt.text(i,round(y.iloc[i],1),round(y.iloc[i],1),size=10,ha="center",va="bottom")
value_labels(grouped.overall)
plt.ylim(60,82)
for i,c in zip(range(0, 10),colors):
    ax.get_children()[i].set_color(c)
#or color=[color for color in colors]
plt.legend(["average GK rating"])
plt.show()
```



plot average overall and average potential ratings by age

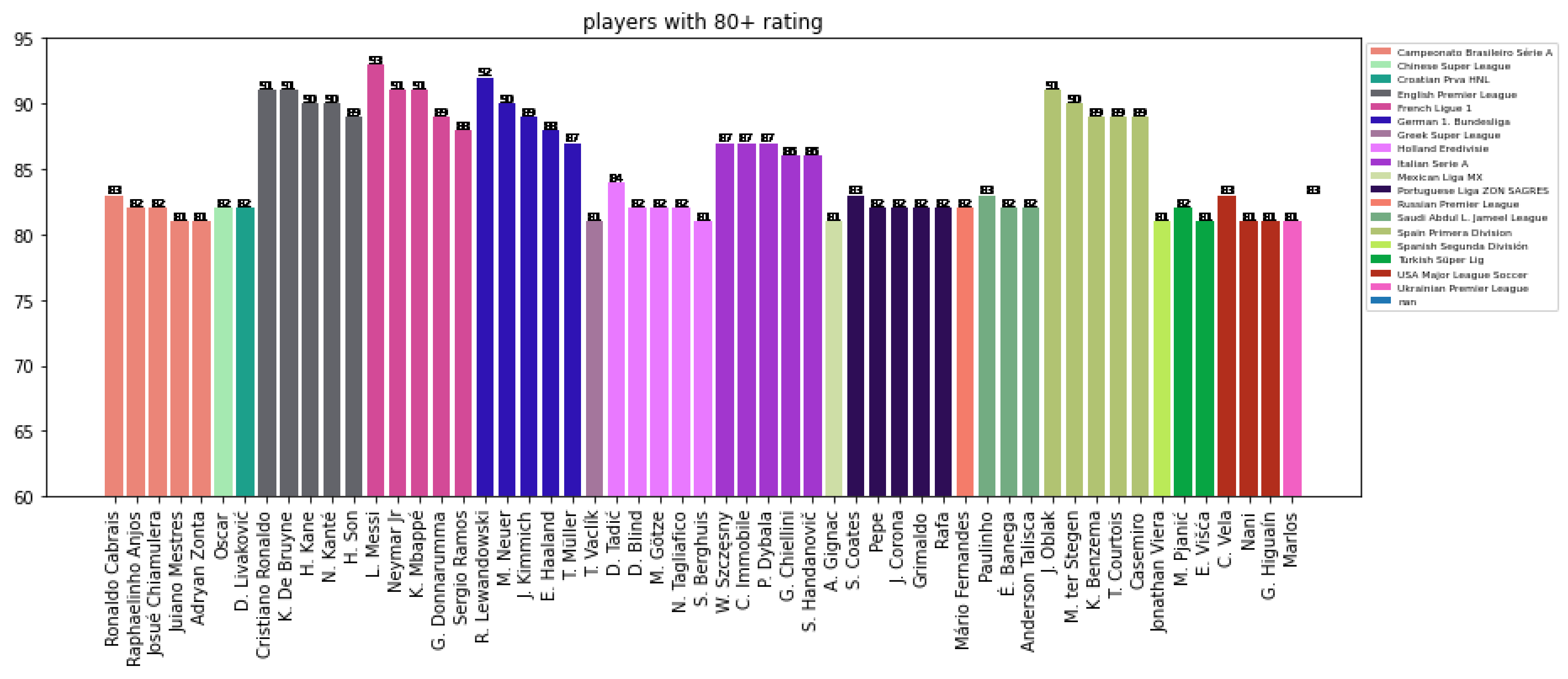
In [15]:
```
grouped=df.groupby("age").agg({"overall":"mean","potential":"mean","short_name":"count"}).reset_index().sort_values("age")
plt.figure(figsize=(15,5))
plt.plot(grouped["age"],grouped["overall"],color="r",label="overall")
plt.plot(grouped["age"],grouped["potential"],color="b",label="potential")
plt.title("average overall vs average potential by age")
plt.fill_between(grouped.age,grouped.overall,grouped.potential,
              where=(grouped.overall>grouped.potential),color="r",interpolate=True,alpha=.25)
plt.fill_between(grouped.age,grouped.overall,grouped.potential,
              where=(grouped.overall<grouped.potential),color="b",interpolate=True,alpha=.25)
plt.legend()
plt.show()
```



find top 5 footballers in each league and thier clubs, having footballer's rating is more than 80

In [17]:
```
grouped=df.sort_values(["league_name","overall"],ascending=[True,False]).groupby("league_name").head(5)
grouped=grouped[grouped["overall"]>80]
import random
leagues=grouped.league_name.unique()
plt.figure(figsize=(14,5))
for l in leagues:
    rgb = (random.random(), random.random(), random.random())
    plt.bar(grouped[grouped.league_name==l].short_name,grouped[grouped.league_name==l].overall,color=[rgb],label=l)
    plt.xticks(rotation=90)
    plt.title("players with 80+ rating")
    plt.legend(bbox_to_anchor=(1,1),fontsize=6)
    def value_labels(y):
        for i in range(len(y)):
            plt.text(i,round(y.iloc[i]),round(y.iloc[i]),size=7,ha="center",va="bottom",rotation=0)
    value_labels(grouped.overall)
    plt.ylim(60,95)
plt.show()
```

```
C:\Users\berid\AppData\Local\Temp\ipykernel_2032\318399413.py:8: MatplotlibDeprecationWarning: Support for passing numbers through unit converters is deprecated since 3.5 and suppo
rt will be removed two minor releases later; use Axis.convert_units instead.
  plt.bar(grouped[grouped.league_name==l].short_name,grouped[grouped.league_name==l].overall,color=[rgb],label=l)
```



In [ ]: