In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import datetime as dt
import mysql.connector
```

In [2]:
```
mydb=mysql.connector.connect(host="localhost",user="root",password="gigaberosql",database="soccer")
```

In [3]:
```
country=pd.read_sql("select * from country",mydb)
league=pd.read_sql("select * from league",mydb)
matches=pd.read_sql("select * from matches",mydb)
player=pd.read_sql("select * from player",mydb)
playerattributes=pd.read_sql("select * from player_attributes",mydb)
team=pd.read_sql("select * from team",mydb)
teamattributes=pd.read_sql("select * from team_attributes",mydb)

matches_view=pd.read_sql("select * from matches_view",mydb)
```

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

Out[4]:

	id	country	league	season	date	home_team	home_team_goal	away_team_goal	away_team
0	1477	Belgium	Belgium Jupiler League	2014/2015	2014-09-20	Royal Excel Mouscron	1	2	KRC Genk
1	1629	Belgium	Belgium Jupiler League	2015/2016	2016-02-06	Royal Excel Mouscron	0	1	KRC Genk
2	1459	Belgium	Belgium Jupiler League	2014/2015	2014-08-30	KV Oostende	1	1	KRC Genk
3	1523	Belgium	Belgium Jupiler League	2015/2016	2015-10-27	KV Oostende	3	2	KRC Genk
4	1116	Belgium	Belgium Jupiler League	2012/2013	2012-12-26	Waasland-Beveren	1	1	KRC Genk

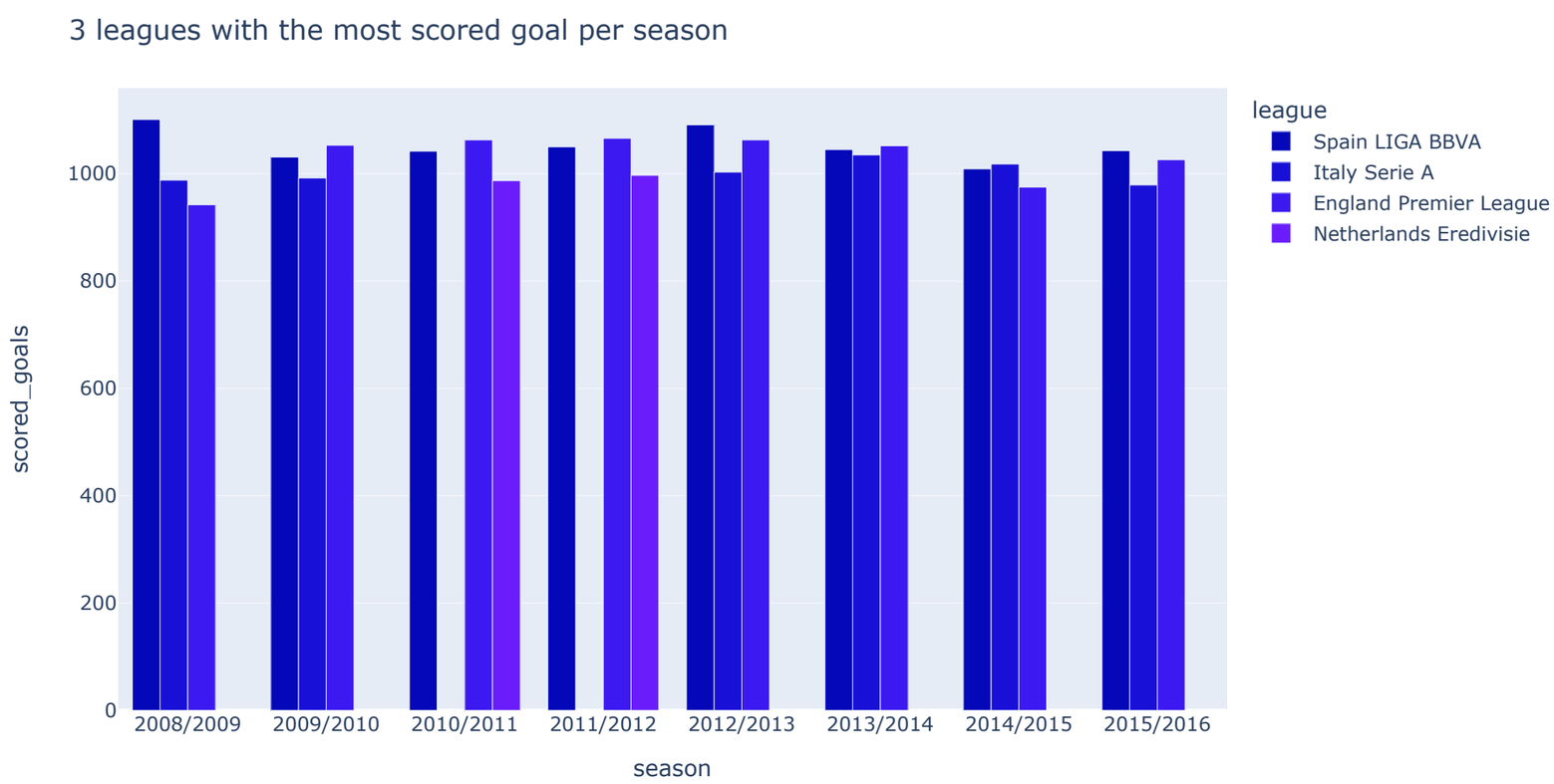
find the 3 leagues with the most scored goals in each season

In [5]:
```
code="""
with a as
(select season,league, sum(home_team_goal+away_team_goal) as scored_goals,
        row_number() over(partition by season order by sum(home_team_goal+away_team_goal) desc) as rn
from matches_view mv
group by season, league
order by season, scored_goals desc)
select *
from a
where rn in (1,2,3)
"""
df=pd.read_sql(code,mydb)
df
```

Out[5]:

	season	league	scored_goals	rn
0	2008/2009	Spain LIGA BBVA	1101.0	1
1	2008/2009	Italy Serie A	988.0	2
2	2008/2009	England Premier League	942.0	3
3	2009/2010	England Premier League	1053.0	1
4	2009/2010	Spain LIGA BBVA	1031.0	2
5	2009/2010	Italy Serie A	992.0	3
6	2010/2011	England Premier League	1063.0	1
7	2010/2011	Spain LIGA BBVA	1042.0	2
8	2010/2011	Netherlands Eredivisie	987.0	3
9	2011/2012	England Premier League	1066.0	1
10	2011/2012	Spain LIGA BBVA	1050.0	2

In [6]:
```
fig=px.bar(df,x="season",y="scored_goals",
           color="league",
           barmode="group",
           color_discrete_sequence=px.colors.sequential.Plotly3)
fig.update_layout(title="3 leagues with the most scored goal per season ")
fig.show()
```

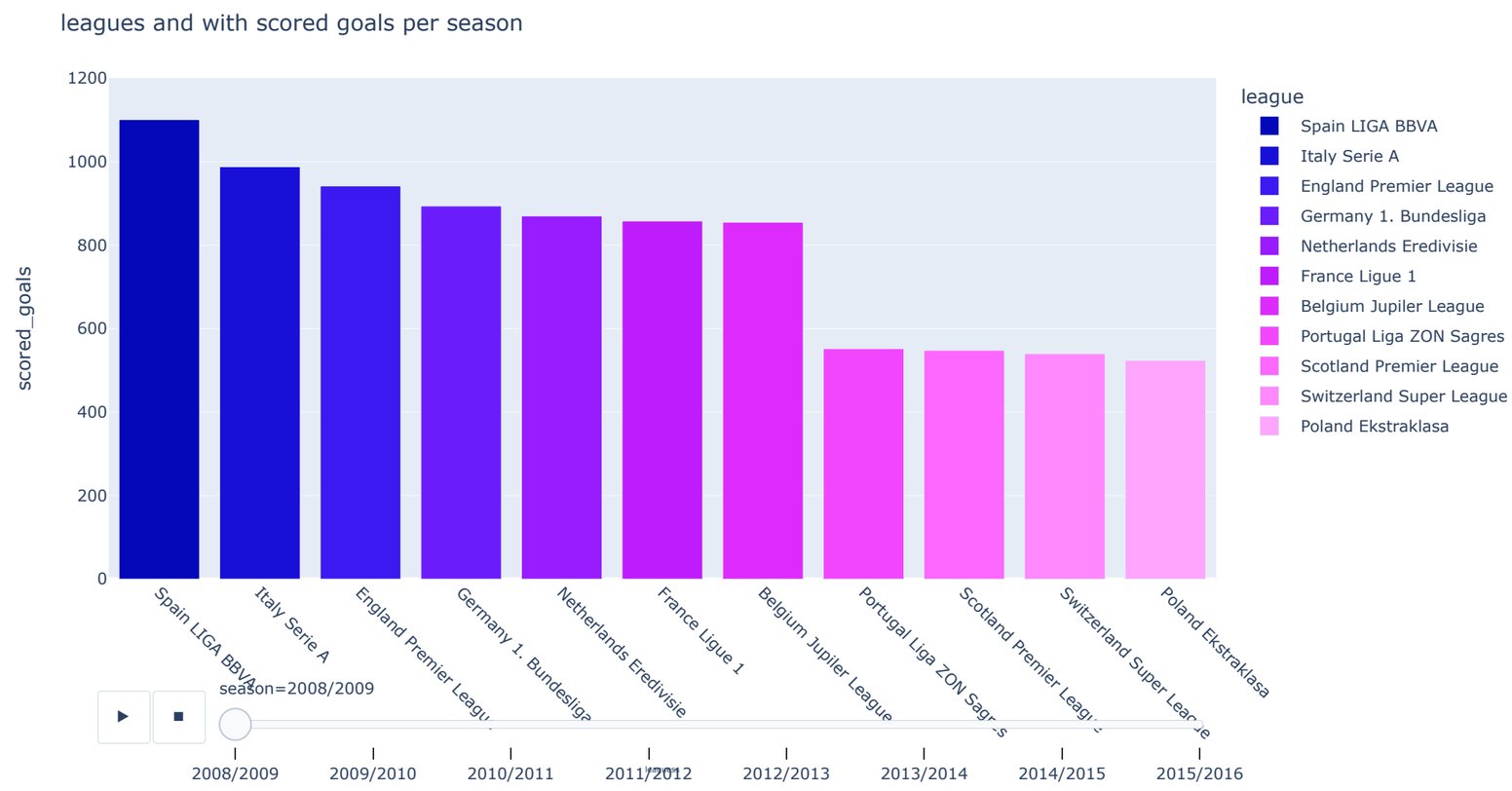


In [7]:
```
code="""
select season,league, sum(home_team_goal+away_team_goal) as scored_goals
from matches_view mv
group by season, league
order by season, scored_goals desc
"""
df=pd.read_sql(code,mydb)
df
```

Out[7]:

	season	league	scored_goals
0	2008/2009	Spain LIGA BBVA	1101.0
1	2008/2009	Italy Serie A	988.0
2	2008/2009	England Premier League	942.0
3	2008/2009	Germany 1. Bundesliga	894.0
4	2008/2009	Netherlands Eredivisie	870.0
...	...	...	...
83	2015/2016	Portugal Liga ZON Sagres	831.0
84	2015/2016	Belgium Jupiler League	694.0
85	2015/2016	Scotland Premier League	650.0
86	2015/2016	Poland Ekstraklasa	635.0
87	2015/2016	Switzerland Super League	566.0

In [8]:
```
fig=px.bar(df,x="league",y="scored_goals",animation_frame="season",
           color="league",
          color_discrete_sequence=px.colors.sequential.Plotly3,
          range_y=(0,1200))
fig.update_layout(title="leagues and with scored goals per season",font=dict(size=10))
fig.update_xaxes(title_text="leaguess",tickangle=45,title_font=dict(size=5))
fig.show()
```



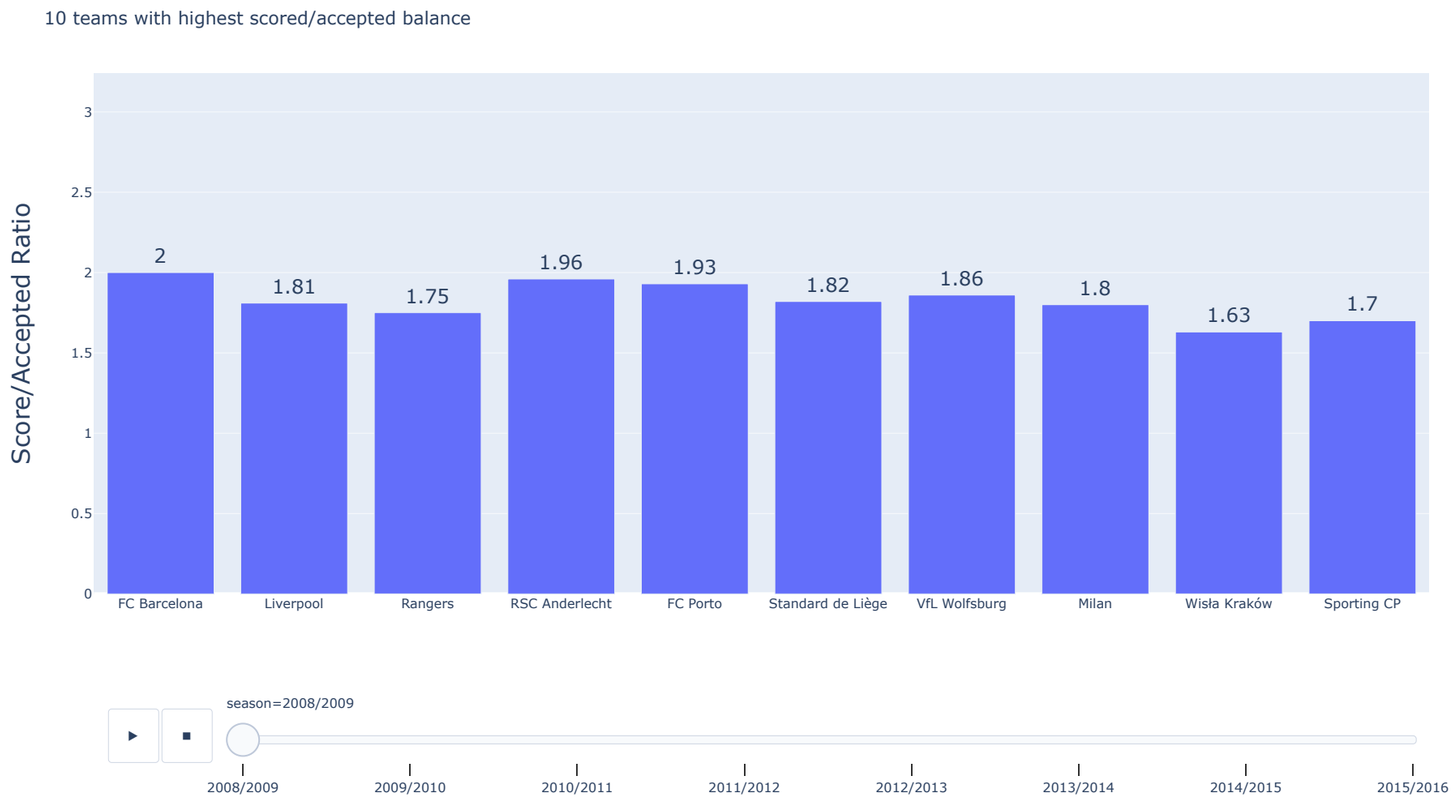
find 10 teams with the best scored/accepted ratio in each season. (Also calculate goal balance)

In [9]:
```
code="""
with a as

(select home.season as season, home.home_team as team,
goal_balance1+goal_balance2 as goal_balance,
round((home_team_ratio+away_team_ratio)/2,2) as `scored/accepted`,
(avg_home_scored+avg_away_scored)/2 as avg_scored_per_match,
(avg_home_accepted+avg_away_accepted)/2 as avg_accepted_per_match,
row_number() over(partition by home.season order by (home_team_ratio+away_team_ratio)/2 desc) as rn

from
(select season, home_team,sum(home_team_goal-away_team_goal) as goal_balance1,avg(home_team_goal/away_team_goal) as home_team_ratio,
avg(home_team_goal) as avg_home_scored,avg(away_team_goal) as avg_home_accepted
from matches_view mv
group by season, home_team) as home
join
(select season, away_team,sum(away_team_goal-home_team_goal) as goal_balance2,avg(away_team_goal/home_team_goal) as away_team_ratio,
avg(away_team_goal) as avg_away_scored,avg(home_team_goal) as avg_away_accepted
from matches_view mv
group by season, away_team) as away
on home.season=away.season and home.home_team=away.away_team
order by season,goal_balance desc)

select *
from a
#where rn between 1 and 10
"""
df=pd.read_sql(code,mydb)
df
```

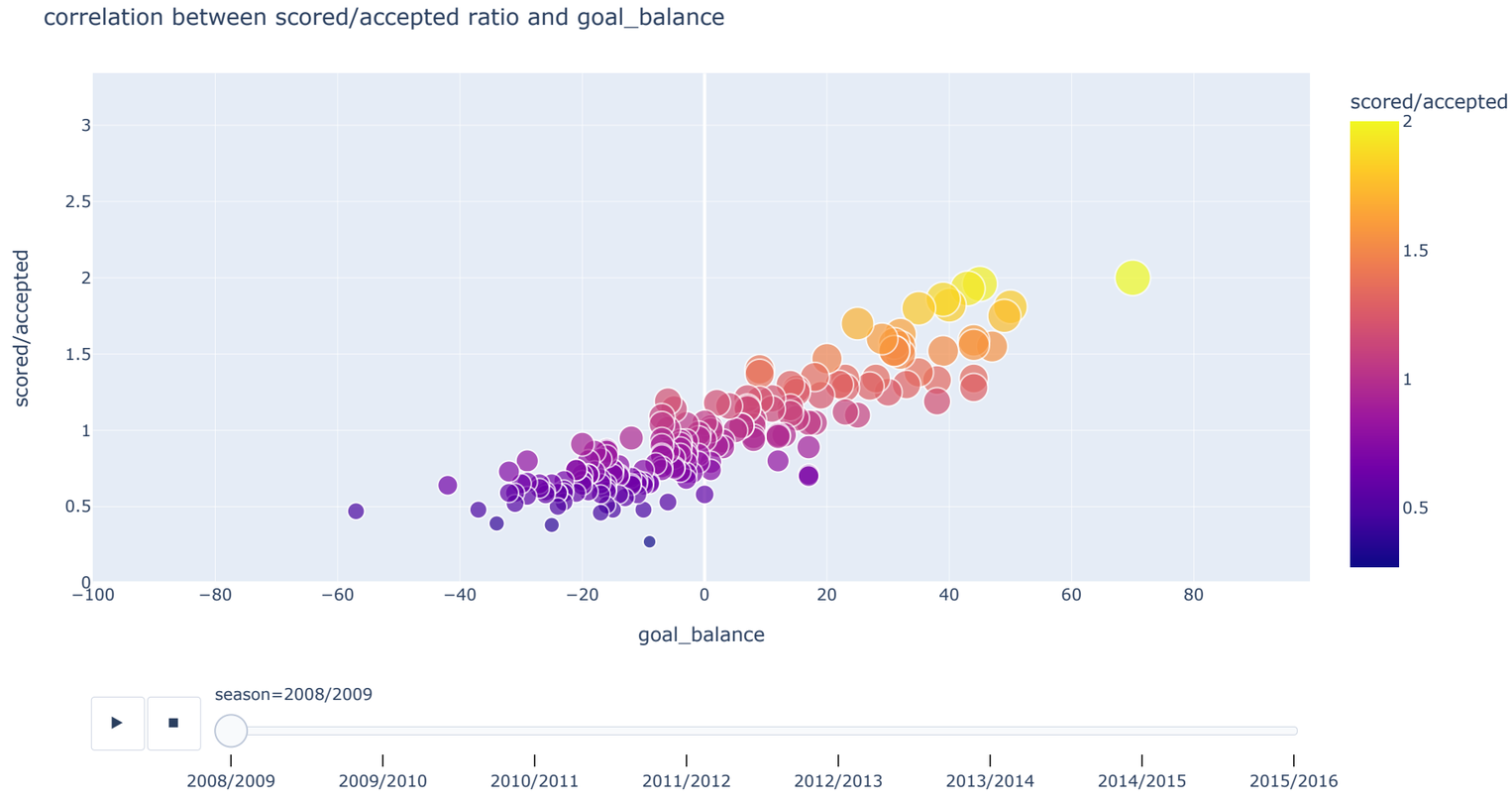
0	2008/2009	FC Barcelona	70.0	2.00	2.76315	0.92105	1
1	2008/2009	Liverpool	50.0	1.81	2.02630	0.71050	6
2	2008/2009	Rangers	49.0	1.75	2.02630	0.73685	8
3	2008/2009	Celtic	47.0	1.55	2.10525	0.86840	15
4	2008/2009	RSC Anderlecht	45.0	1.96	2.20585	0.88240	2
...	...	...	...	...	...	...	...
1473	2015/2016	RCD Espanyol	-34.0	0.75	1.05265	1.94740	135
1474	2015/2016	Frosinone	-41.0	0.48	0.92105	2.00000	183
1475	2015/2016	SC Cambuur	-46.0	0.58	0.97060	2.32355	178
1476	2015/2016	Aston Villa	-49.0	0.35	0.71050	2.00000	187
1477	2015/2016	ES Troyes AC	-55.0	0.46	0.73685	2.18420	184

1478 rows × 7 columns

In [10]:
```
ndf=df[df.rn.isin([i for i in range(1,11)])]

fig=px.bar(ndf,
           x="team",y="scored/accepted",
           animation_frame="season",
           text_auto=True,
           range_y=(0,df["scored/accepted"].max()))
fig.update_layout(title_text="10 teams with highest scored/accepted balance",font=dict(size=8))
fig.update_xaxes(title_text=None)
fig.update_yaxes(title_text="Score/Accepted Ratio",title_font=dict(size=15))
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)

fig.show()
```

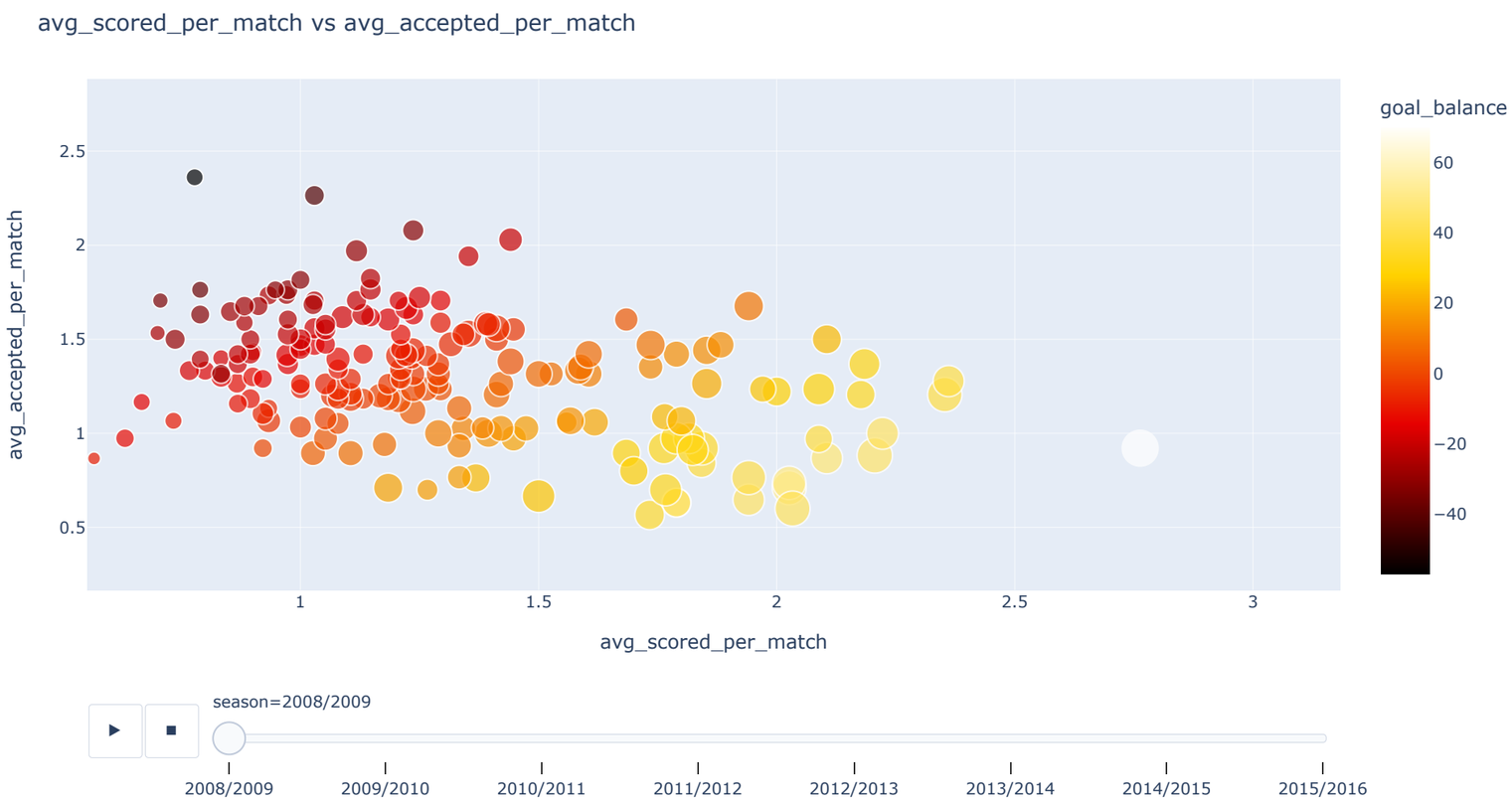


In [11]:
```
fig=px.scatter(df[df["scored/accepted"].isna()==False],"goal_balance",y="scored/accepted",
           animation_frame="season",
           color="scored/accepted",
           size="scored/accepted",
           hover_name="team",
           range_y=(0,df["scored/accepted"].max()+.1),
          range_x=(-100,df["goal_balance"].max()+10))

fig.update_layout(title_text="correlation between scored/accepted ratio and goal_balance",font=dict(size=10))
fig.show()

print("correlation between scored/accepted ratio and goal_balance is "+str(round(df["scored/accepted"].corr(df["goal_balance"]),2)))
```



correlation between scored/accepted ratio and goal_balance is 0.89

In [12]:
```
df=df.fillna(0)

fig=px.scatter(df,x="avg_scored_per_match",y="avg_accepted_per_match",
               size=df["scored/accepted"], color="goal_balance",
               animation_frame="season",
               range_x=(df["avg_scored_per_match"].min(),df["avg_scored_per_match"].max()),
               range_y=(df["avg_accepted_per_match"].min(),df["avg_accepted_per_match"].max()),
                color_continuous_scale="hot",
               hover_name="team")
fig.update_layout(title_text="avg_scored_per_match vs avg_accepted_per_match", font=dict(size=10))
fig.show()
```



In [13]:
```
# teams that have a good offence, also have a good defence or if a team has a bad defence it is also bad in attack
```

how many matches has bayern won against dortmund from 2008 to 2016?

In [14]:
```
code="""
select home_wins+away_wins as wins
from
(select home_team, away_team, count(if(home_team_goal>away_team_goal,home_team_goal,Null)) as home_wins
from matches_view mv
where home_team="FC Bayern Munich" and away_team="Borussia Dortmund") as home
join
(select away_team, home_team, count(if(away_team_goal>home_team_goal,away_team_goal,Null)) as away_wins
from matches_view mv
where away_team="FC Bayern Munich" and home_team="Borussia Dortmund") as away
on home.home_team=away.away_team and home.away_team=away.home_team
"""
df=pd.read_sql(code,mydb)
df
```

Out[14]:

	wins
0	7

find teams that has won more than 75% of matches in a season

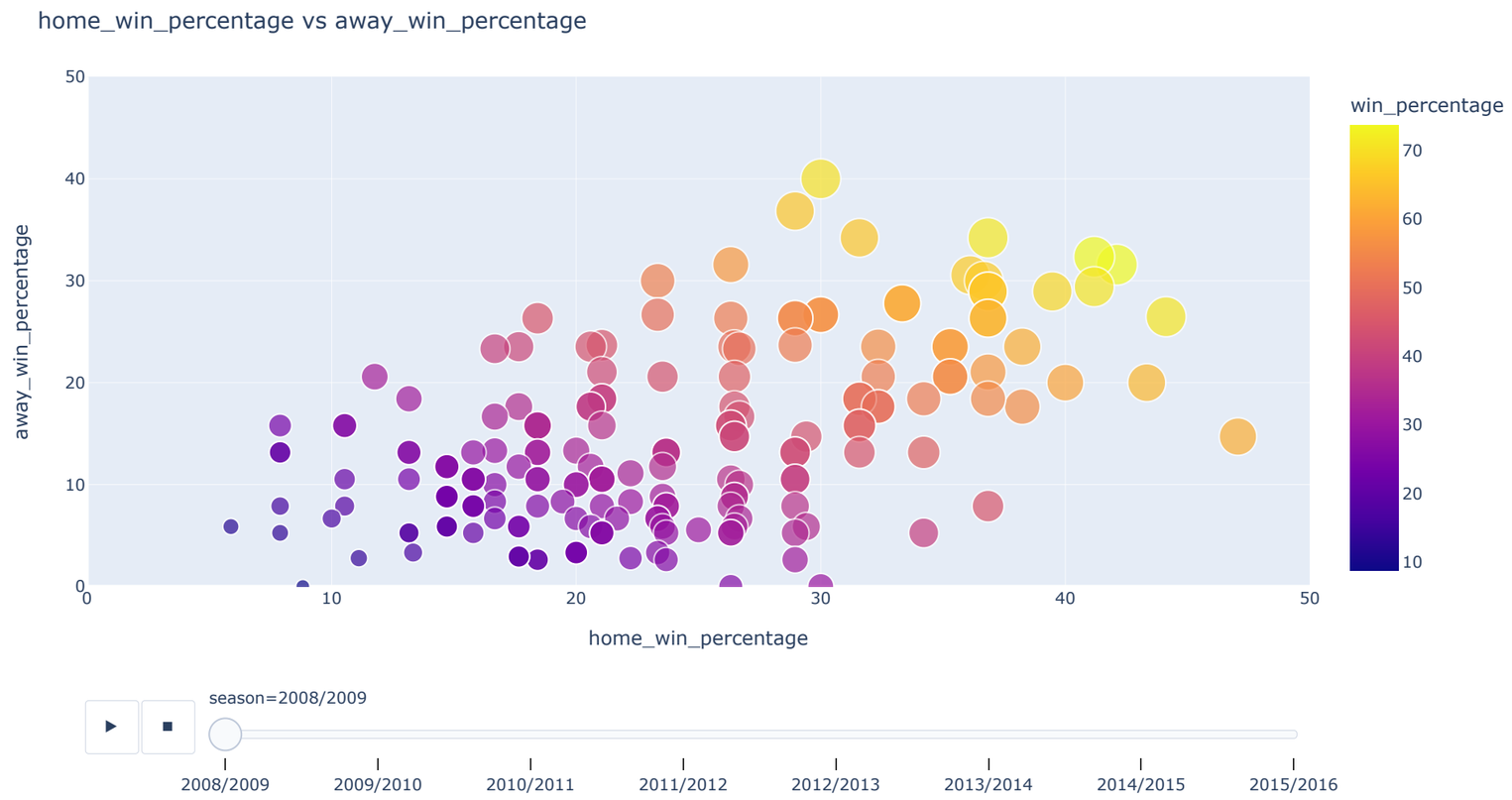
In [15]:
```
code="""
select home.season as season,home.home_team as team, home_wins, away_wins, home_wins+away_wins as total_wins,
home_matches+away_matches as total_matches, round(100*(home_wins+away_wins)/(home_matches+away_matches),2) as win_percentage,
round(100*(home_wins)/(home_matches+away_matches),2) as home_win_percentage,
round(100*(away_wins)/(home_matches+away_matches),2) as away_win_percentage

from
(select season, home_team, count(if(home_team_goal>away_team_goal,home_team,Null)) as home_wins, count(home_team) as home_matches
from matches_view mv
group by season, home_team) as home
join
(select season, away_team, count(if(away_team_goal>home_team_goal,home_team,Null)) as away_wins, count(away_team) as away_matches
from matches_view mv
group by season, away_team) as away
on home.home_team=away.away_team and home.season=away.season
#having win_percentage>=75
order by season, win_percentage desc
"""
df=pd.read_sql(code,mydb)
df
```

Out[15]:

	season	team	home_wins	away_wins	total_wins	total_matches	win_percentage	home_win_percentage	away_win_percentage
0	2008/2009	Manchester United	16	12	28	38	73.68	42.11	31.58
1	2008/2009	AZ	14	11	25	34	73.53	41.18	32.35
2	2008/2009	FC Barcelona	14	13	27	38	71.05	36.84	34.21
3	2008/2009	RSC Anderlecht	14	10	24	34	70.59	41.18	29.41
4	2008/2009	Standard de Liège	15	9	24	34	70.59	44.12	26.47
...	...	...	...	...	...	...	...	...	...
1473	2015/2016	Polonia Bytom	2	2	4	30	13.33	6.67	6.67
1474	2015/2016	Hellas Verona	4	1	5	38	13.16	10.53	2.63
1475	2015/2016	SC Cambuur	2	1	3	34	8.82	5.88	2.94
1476	2015/2016	ES Troyes AC	1	2	3	38	7.89	2.63	5.26
1477	2015/2016	Aston Villa	2	1	3	38	7.89	5.26	2.63

1478 rows × 9 columns

In [16]:
```
fig=px.scatter(df,x="home_win_percentage",y="away_win_percentage",
               color="win_percentage",size="win_percentage",
               hover_name="team",
               animation_frame="season",
               range_x=(0,df.home_win_percentage.max()),range_y=(0,df.away_win_percentage.max()))

fig.update_layout(title_text="home_win_percentage vs away_win_percentage", font=dict(size=10))

fig.show()
```



find 3 teams for each season and league with the longest consecutive home win chain

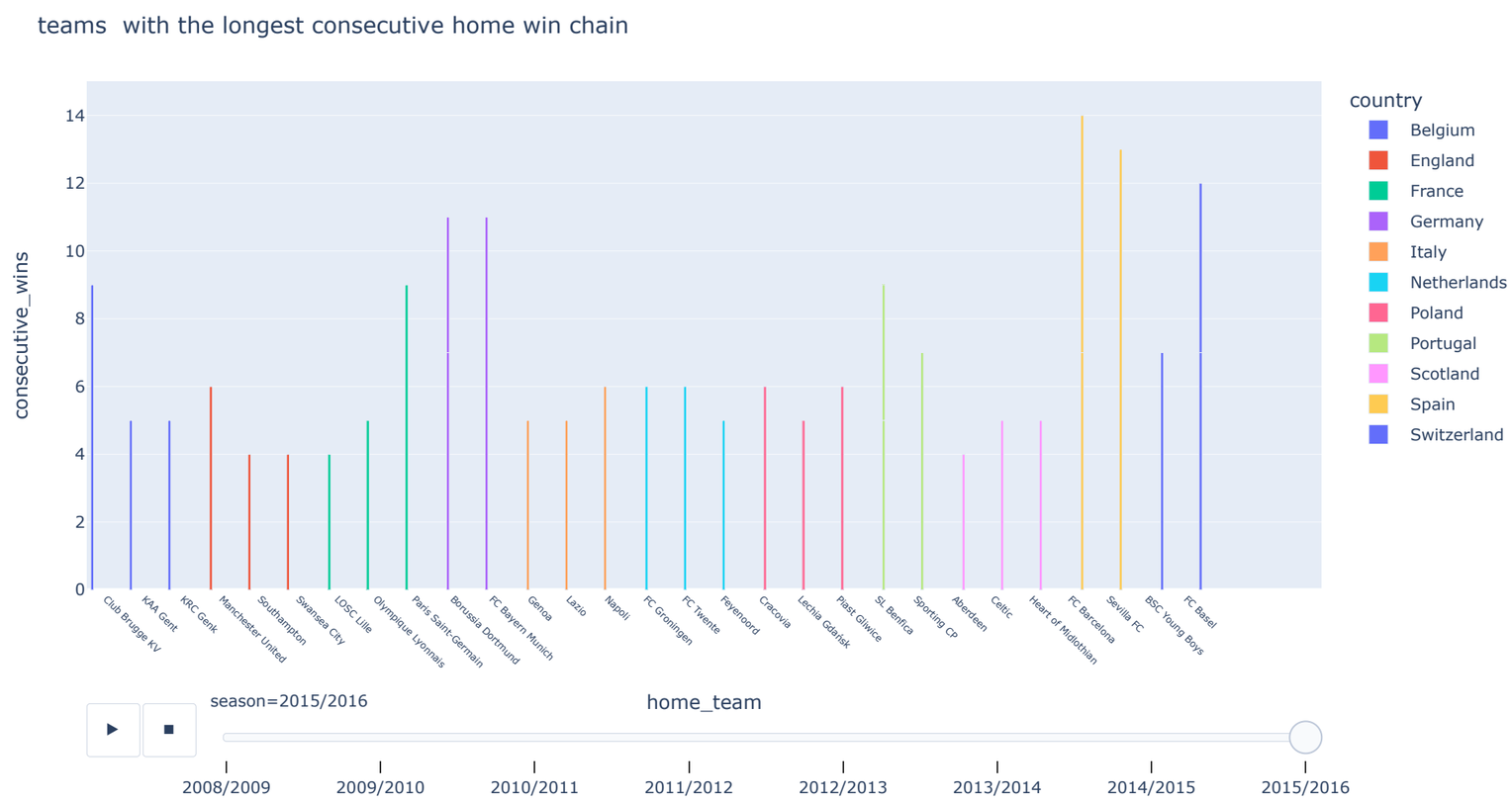
In [43]:
```
code="""
with c as
(with b as
(with a as
(select mv.*,row_number() over(partition by season,country,home_team order by date) as rn
from matches_view mv)
select a.*, ifnull(lag(rn) over(partition by season,country,home_team order by date),0) as previous_rn, rn-ifnull(lag(rn) over(partition by season,country,home_team order by date),0
from a
where home_team_goal <= away_team_goal)
select season,country,home_team,consecutive_wins,
row_number() over(partition by  season,country order by consecutive_wins desc) as rn2
from b
order by season,country,home_team, consecutive_wins desc)
select season,country,home_team,consecutive_wins
from c
where rn2 in (1,2,3)
"""
df=pd.read_sql(code,mydb)
df
```

Out[43]:

	season	country	home_team	consecutive_wins
0	2008/2009	Belgium	Beerschot AC	4.0
1	2008/2009	Belgium	Club Brugge KV	4.0
2	2008/2009	Belgium	KVC Westerlo	5.0
3	2008/2009	England	Manchester City	6.0
4	2008/2009	England	Manchester United	12.0
...	...	...	...	...
259	2015/2016	Spain	FC Barcelona	7.0
260	2015/2016	Spain	Sevilla FC	13.0
261	2015/2016	Switzerland	BSC Young Boys	7.0
262	2015/2016	Switzerland	FC Basel	7.0
263	2015/2016	Switzerland	FC Basel	5.0

264 rows × 4 columns

In [44]:
```
fig=px.bar(df,x="home_team",y="consecutive_wins",
       color="country", barmode="group",
       animation_frame="season",
      hover_name="home_team",
          range_y=(0,15))
fig.update_layout(title_text="teams  with the longest consecutive home win chain",font=dict(size=10))
fig.update_xaxes(tickangle=45,tickfont=dict(size=6))

fig.update_traces(marker_line_width=0.1)

fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 1500


fig.show()
```



which team has the highest win percentage against which team. Create procedure to find win percentage between two teams

In [45]:
```
code="""
select home.country,home.league,home.home_team, home.away_team, round((home_wins+away_wins)/(home_matches+away_matches)*100,2) as win_percentage,
home_wins+away_wins as wins, home_matches+away_matches as matches
from
(select country,league, home_team, away_team, count(if(home_team_goal>away_team_goal,home_team,Null)) as home_wins, count(home_team) as home_matches
from matches_view
group by home_team, away_team) as home
join
(select country,league, home_team, away_team, count(if(home_team_goal<away_team_goal,home_team,Null)) as away_wins, count(away_team) as away_matches
from matches_view
group by home_team, away_team) as away
on home.home_team=away.away_team and home.away_team=away.home_team
having matches>10 #and country="germany"
order by league, win_percentage desc
"""
df=pd.read_sql(code,mydb)
df
```

Out[45]:

	country	league	home_team	away_team	win_percentage	wins	matches
0	Belgium	Belgium Jupiler League	Club Brugge KV	KVC Westerlo	91.67	11	12
1	Belgium	Belgium Jupiler League	RSC Anderlecht	KV Kortrijk	85.71	12	14
2	Belgium	Belgium Jupiler League	RSC Anderlecht	KVC Westerlo	83.33	10	12
3	Belgium	Belgium Jupiler League	KV Kortrijk	KSV Cercle Brugge	83.33	10	12
4	Belgium	Belgium Jupiler League	RSC Anderlecht	KSV Cercle Brugge	75.00	9	12
...	...	...	...	...	...	...	...
1861	Switzerland	Switzerland Super League	Neuchâtel Xamax	FC Basel	7.14	1	14
1862	Switzerland	Switzerland Super League	FC Sion	FC Basel	6.25	2	32
1863	Switzerland	Switzerland Super League	FC Vaduz	FC Basel	0.00	0	12
1864	Switzerland	Switzerland Super League	FC Lausanne-Sports	FC Basel	0.00	0	12
1865	Switzerland	Switzerland Super League	FC Vaduz	FC Zürich	0.00	0	12

1866 rows × 7 columns