Data analysis using Python with SQL

In [15]:

```python
import numpy as np
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt

database = 'database.sqlite'
```

In [16]:

```python
conn = sqlite3.connect(database)

tables = pd.read_sql("""SELECT *
                        FROM sqlite_master
                        WHERE type='table';""", conn)
tables
```

Out[16]:

	type	name	tbl_name	rootpage	sql
0	table	sqlite_sequence	sqlite_sequence	4	CREATE TABLE sqlite_sequence(name,seq)
1	table	Player_Attributes	Player_Attributes	11	CREATE TABLE "Player_Attributes" (\n\t`id`\tIN...
2	table	Player	Player	14	CREATE TABLE `Player` (\n\t`id`\tINTEGER PRIMA...
3	table	Match	Match	18	CREATE TABLE `Match` (\n\t`id`\tINTEGER PRIMAR...
4	table	League	League	24	CREATE TABLE `League` (\n\t`id`\tINTEGER PRIMA...
5	table	Country	Country	26	CREATE TABLE `Country` (\n\t`id`\tINTEGER PRIM...
6	table	Team	Team	29	CREATE TABLE "Team" (\n\t`id`\tINTEGER PRIMARY...
7	table	Team_Attributes	Team_Attributes	2	CREATE TABLE `Team_Attributes` (\n\t`id`\tINTE...

List of countries

In [17]:

```python
countries = pd.read_sql("""SELECT *
                        FROM Country;""", conn)
countries
```

Out[17]:

	id	name
0	1	Belgium
1	1729	England
2	4769	France
3	7809	Germany
4	10257	Italy
5	13274	Netherlands
6	15722	Poland
7	17642	Portugal
8	19694	Scotland
9	21518	Spain
10	24558	Switzerland

List of leagues and their country

In [18]:

```python
leagues = pd.read_sql("""SELECT *
                        FROM League
                        JOIN Country ON Country.id = League.country_id;""", conn)
leagues
```

Out[18]:

	id	country_id	name	id	name
0	1	1	Belgium Jupiler League	1	Belgium
1	1729	1729	England Premier League	1729	England
2	4769	4769	France Ligue 1	4769	France
3	7809	7809	Germany 1. Bundesliga	7809	Germany
4	10257	10257	Italy Serie A	10257	Italy
5	13274	13274	Netherlands Eredivisie	13274	Netherlands
6	15722	15722	Poland Ekstraklasa	15722	Poland
7	17642	17642	Portugal Liga ZON Sagres	17642	Portugal
8	19694	19694	Scotland Premier League	19694	Scotland
9	21518	21518	Spain LIGA BBVA	21518	Spain
10	24558	24558	Switzerland Super League	24558	Switzerland

List of teams

In [19]:

```python
teams = pd.read_sql("""SELECT *
                        FROM Team
                        ORDER BY team_long_name
                        LIMIT 10;""", conn)
teams
```

Out[19]:

	id	team_api_id	team_fifa_api_id	team_long_name	team_short_name
0	16848	8350	29	1. FC Kaiserslautern	KAI
1	15624	8722	31	1. FC Köln	FCK
2	16239	8165	171	1. FC Nürnberg	NUR
3	16243	9905	169	1. FSV Mainz 05	MAI
4	11817	8576	614	AC Ajaccio	AJA
5	11074	108893	111989	AC Arles-Avignon	ARL
6	49116	6493	1714	AC Bellinzona	BEL
7	26560	10217	650	ADO Den Haag	HAA
8	9537	8583	57	AJ Auxerre	AUX
9	9547	9829	69	AS Monaco	MON

List of matches

Note that the Team tables are joined using left join. The reason is I would prefer to keep the matches in the output

ORDER defines the order of the output, and comes before the LIMIT and after the WHERE

In [20]:

```python
detailed_matches = pd.read_sql("""SELECT Match.id,
                                        Country.name AS country_name,
                                        League.name AS league_name,
                                        season,
                                        stage,
                                        date,
                                        HT.team_long_name AS  home_team,
                                        AT.team_long_name AS away_team,
                                        home_team_goal,
                                        away_team_goal
                                FROM Match
                                JOIN Country on Country.id = Match.country_id
                                JOIN League on League.id = Match.league_id
                                LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
                                LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
                                WHERE country_name = 'Spain'
                                ORDER by date
                                LIMIT 10;""", conn)
detailed_matches
```

Out[20]:

	id	country_name	league_name	season	stage	date	home_team	away_team	home_team_goal	away_team_goal
0	21518	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-30 00:00:00	Valencia CF	RCD Mallorca	3	0
1	21525	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-30 00:00:00	RCD Espanyol	Real Valladolid	1	0
2	21519	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	CA Osasuna	Villarreal CF	1	1
3	21520	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	RC Deportivo de La Coruña	Real Madrid CF	2	1
4	21521	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	CD Numancia	FC Barcelona	1	0
5	21522	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	Racing Santander	Sevilla FC	1	1
6	21523	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	Real Sporting de Gijón	Getafe CF	1	2
7	21524	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	Real Betis Balompié	RC Recreativo	0	1
8	21526	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	Athletic Club de Bilbao	UD Almería	1	3
9	21527	Spain	Spain LIGA BBVA	2008/2009	1	2008-08-31 00:00:00	Atlético Madrid	Málaga CF	4	0

Basic analytics

In this example, we will base it on the previous query, remove the match and date information, and look at it at the country-league-season level.

- Define which tables to use, and connect them (FROM + JOIN)
- Keep only the rows that apply to the conditions (WHERE)
- Group the data by the required level (if need) (GROUP BY)
- Order the output of the new table (ORDER BY)
- Add more conditions that would filter the new created table (HAVING)
- Limit to number of rows - would cut it according the soring and the having filtering (LIMIT)

In [21]:

```python
leages_by_season = pd.read_sql("""SELECT Country.name AS country_name,
                                        League.name AS league_name,
                                        season,
                                        count(distinct stage) AS number_of_stages,
                                        count(distinct HT.team_long_name) AS number_of_teams,
                                        avg(home_team_goal) AS avg_home_team_scors,
                                        avg(away_team_goal) AS avg_away_team_goals,
                                        avg(home_team_goal-away_team_goal) AS avg_goal_dif,
                                        avg(home_team_goal+away_team_goal) AS avg_goals,
                                        sum(home_team_goal+away_team_goal) AS total_goals
                                FROM Match
                                JOIN Country on Country.id = Match.country_id
                                JOIN League on League.id = Match.league_id
                                LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
                                LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
                                WHERE country_name in ('Spain', 'Germany', 'France', 'Italy', 'England')
                                GROUP BY Country.name, League.name, season
                                HAVING count(distinct stage) > 10
                                ORDER BY Country.name, League.name, season DESC
                                ;""", conn)
leages_by_season
```

Out[21]:

	country_name	league_name	season	number_of_stages	number_of_teams	avg_home_team_scors	avg_away_team_goals	avg_goal_dif	avg_goals	total_goals
0	England	England Premier League	2015/2016	38	20	1.492105	1.207895	0.284211	2.700000	1026
1	England	England Premier League	2014/2015	38	20	1.473684	1.092105	0.381579	2.565789	975
2	England	England Premier League	2013/2014	38	20	1.573684	1.194737	0.378947	2.768421	1052
3	England	England Premier League	2012/2013	38	20	1.557895	1.239474	0.318421	2.797368	1063
4	England	England Premier League	2011/2012	38	20	1.589474	1.215789	0.373684	2.805263	1066
5	England	England Premier League	2010/2011	38	20	1.623684	1.173684	0.450000	2.797368	1063
6	England	England Premier League	2009/2010	38	20	1.697368	1.073684	0.623684	2.771053	1053
7	England	England Premier League	2008/2009	38	20	1.400000	1.078947	0.321053	2.478947	942
8	France	France Ligue 1	2015/2016	38	20	1.436842	1.089474	0.347368	2.526316	960
9	France	France Ligue 1	2014/2015	38	20	1.410526	1.081579	0.328947	2.492105	947
10	France	France Ligue 1	2013/2014	38	20	1.415789	1.039474	0.376316	2.455263	933
11	France	France Ligue 1	2012/2013	38	20	1.468421	1.076316	0.392105	2.544737	967
12	France	France Ligue 1	2011/2012	38	20	1.473684	1.042105	0.431579	2.515789	956
13	France	France Ligue 1	2010/2011	38	20	1.342105	1.000000	0.342105	2.342105	890
14	France	France Ligue 1	2009/2010	38	20	1.389474	1.021053	0.368421	2.410526	916
15	France	France Ligue 1	2008/2009	38	20	1.286842	0.971053	0.315789	2.257895	858
16	Germany	Germany 1. Bundesliga	2015/2016	34	18	1.565359	1.264706	0.300654	2.830065	866
17	Germany	Germany 1. Bundesliga	2014/2015	34	18	1.588235	1.166667	0.421569	2.754902	843
18	Germany	Germany 1. Bundesliga	2013/2014	34	18	1.748366	1.411765	0.336601	3.160131	967
19	Germany	Germany 1. Bundesliga	2012/2013	34	18	1.591503	1.343137	0.248366	2.934641	898
20	Germany	Germany 1. Bundesliga	2011/2012	34	18	1.660131	1.199346	0.460784	2.859477	875
21	Germany	Germany 1. Bundesliga	2010/2011	34	18	1.647059	1.274510	0.372549	2.921569	894
22	Germany	Germany 1. Bundesliga	2009/2010	34	18	1.513072	1.316993	0.196078	2.830065	866
23	Germany	Germany 1. Bundesliga	2008/2009	34	18	1.699346	1.222222	0.477124	2.921569	894
24	Italy	Italy Serie A	2015/2016	38	20	1.471053	1.105263	0.365789	2.576316	979
25	Italy	Italy Serie A	2014/2015	38	20	1.498681	1.187335	0.311346	2.686016	1018
26	Italy	Italy Serie A	2013/2014	38	20	1.536842	1.186842	0.350000	2.723684	1035
27	Italy	Italy Serie A	2012/2013	38	20	1.494737	1.144737	0.350000	2.639474	1003
28	Italy	Italy Serie A	2011/2012	38	20	1.511173	1.072626	0.438547	2.583799	925
29	Italy	Italy Serie A	2010/2011	38	20	1.431579	1.081579	0.350000	2.513158	955
30	Italy	Italy Serie A	2009/2010	38	20	1.542105	1.068421	0.473684	2.610526	992
31	Italy	Italy Serie A	2008/2009	38	20	1.521053	1.078947	0.442105	2.600000	988
32	Spain	Spain LIGA BBVA	2015/2016	38	20	1.618421	1.126316	0.492105	2.744737	1043
33	Spain	Spain LIGA BBVA	2014/2015	38	20	1.536842	1.118421	0.418421	2.655263	1009
34	Spain	Spain LIGA BBVA	2013/2014	38	20	1.631579	1.118421	0.513158	2.750000	1045
35	Spain	Spain LIGA BBVA	2012/2013	38	20	1.686842	1.184211	0.502632	2.871053	1091
36	Spain	Spain LIGA BBVA	2011/2012	38	20	1.678947	1.084211	0.594737	2.763158	1050
37	Spain	Spain LIGA BBVA	2010/2011	38	20	1.636842	1.105263	0.531579	2.742105	1042
38	Spain	Spain LIGA BBVA	2009/2010	38	20	1.600000	1.113158	0.486842	2.713158	1031
39	Spain	Spain LIGA BBVA	2008/2009	38	20	1.660526	1.236842	0.423684	2.897368	1101

In [22]:

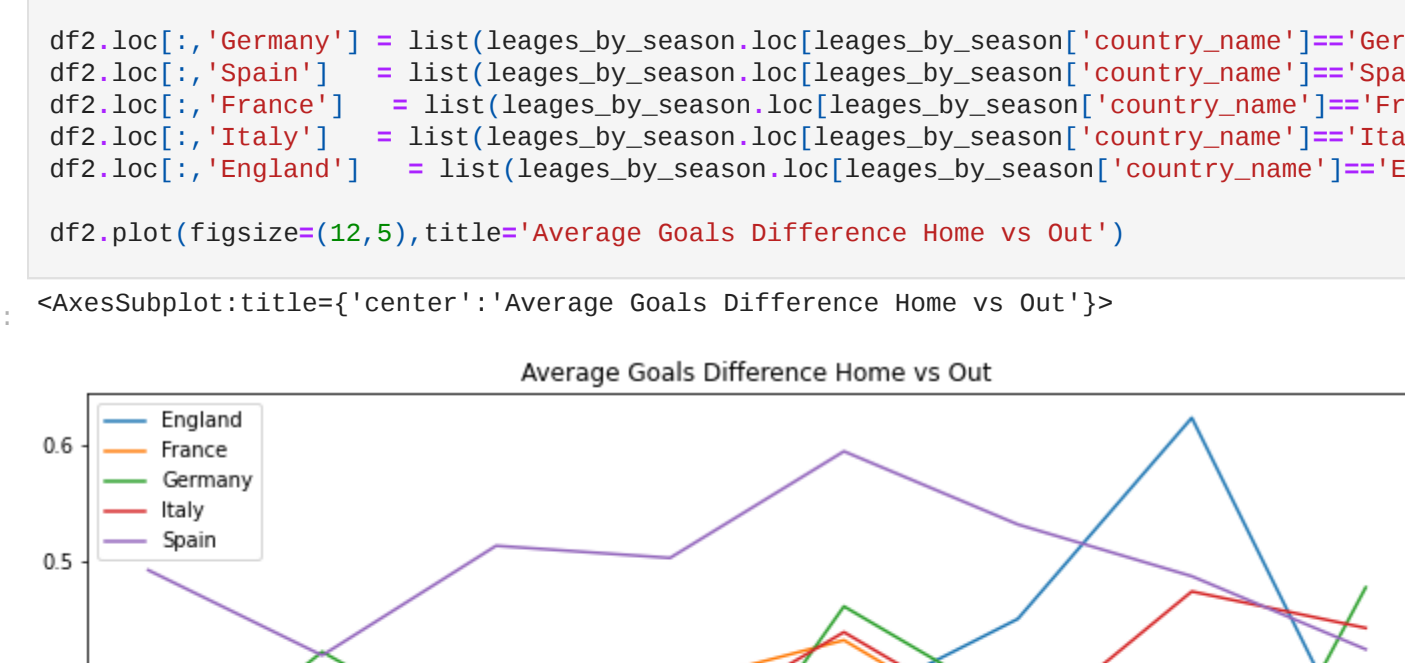
```python
df = pd.DataFrame(index=np.sort(leages_by_season['season'].unique()), columns=leages_by_season['country_name'].unique())

df.loc[:,'Germany'] = list(leages_by_season.loc[leages_by_season['country_name']=='Germany','avg_goals'])
df.loc[:,'Spain']   = list(leages_by_season.loc[leages_by_season['country_name']=='Spain','avg_goals'])
df.loc[:,'France']   = list(leages_by_season.loc[leages_by_season['country_name']=='France','avg_goals'])
df.loc[:,'Italy']   = list(leages_by_season.loc[leages_by_season['country_name']=='Italy','avg_goals'])
df.loc[:,'England']   = list(leages_by_season.loc[leages_by_season['country_name']=='England','avg_goals'])

df.plot(figsize=(12,5),title='Average Goals per Game Over Time')
```

Out[22]:

```
<AxesSubplot:title={'center':'Average Goals per Game Over Time'}>
```



In [23]:

```python
df2 = pd.DataFrame(index=np.sort(leages_by_season['season'].unique()), columns=leages_by_season['country_name'].unique())

df2.loc[:,'Germany'] = list(leages_by_season.loc[leages_by_season['country_name']=='Germany','avg_goal_dif'])
df2.loc[:,'Spain']   = list(leages_by_season.loc[leages_by_season['country_name']=='Spain','avg_goal_dif'])
df2.loc[:,'France']   = list(leages_by_season.loc[leages_by_season['country_name']=='France','avg_goal_dif'])
df2.loc[:,'Italy']   = list(leages_by_season.loc[leages_by_season['country_name']=='Italy','avg_goal_dif'])
df2.loc[:,'England']   = list(leages_by_season.loc[leages_by_season['country_name']=='England','avg_goal_dif'])

df2.plot(figsize=(12,5),title='Average Goals Difference Home vs Out')
```

Out[23]:

```
<AxesSubplot:title={'center':'Average Goals Difference Home vs Out'}>
```

Sub Queries

Group the attributes table, to a different key - player level only (without season). I used average

In [24]:

```python
players_height = pd.read_sql("""SELECT CASE
                                        WHEN ROUND(height)<165 then 165
                                        WHEN ROUND(height)>195 then 195
                                        ELSE ROUND(height)
                                        END AS calc_height,
                                        COUNT(height) AS distribution,
                                        (avg(PA_Grouped.avg_overall_rating)) AS avg_overall_rating,
                                        (avg(PA_Grouped.avg_potential)) AS avg_potential,
                                        AVG(weight) AS avg_weight
                                FROM PLAYER
                                LEFT JOIN (SELECT Player_Attributes.player_api_id,
                                        avg(Player_Attributes.overall_rating) AS avg_overall_rating,
                                        avg(Player_Attributes.potential) AS avg_potential
                                        FROM Player_Attributes
                                        GROUP BY Player_Attributes.player_api_id)
                                        AS PA_Grouped ON PLAYER.player_api_id = PA_Grouped.player_api_id
                                GROUP BY calc_height
                                ORDER BY calc_height
                                    ;""", conn)
players_height
```

Out[24]:

	calc_height	distribution	avg_overall_rating	avg_potential	avg_weight
0	165.0	74	67.365543	73.327754	139.459459
1	168.0	118	67.500518	73.124182	144.127119
2	170.0	403	67.726903	73.379056	147.799007
3	173.0	530	66.980272	72.848746	152.824528
4	175.0	1188	66.805204	72.258774	156.111953
5	178.0	1489	66.367212	71.943339	160.665547
6	180.0	1388	66.419053	71.846394	165.261527
7	183.0	1954	66.634380	71.754555	170.167861
8	185.0	1278	66.928964	71.833475	174.636933
9	188.0	1305	67.094253	72.151949	179.278161
10	191.0	652	66.997649	71.846159	184.791411
11	193.0	470	67.485141	72.459225	188.795745
12	195.0	211	67.425619	72.615373	196.464455

In [25]:

```python
players_height.dtypes
```

Out[25]:

```
calc_height           float64
distribution            int64
avg_overall_rating    float64
avg_potential         float64
avg_weight            float64
dtype: object
```

In [26]:

```python
players_height.plot('calc_height', 'avg_overall_rating' ,figsize=(12,5),title='Potential vs Height')
```

Out[26]:

```
<AxesSubplot:title={'center':'Potential vs Height'}, xlabel='calc_height'>
```