Database Normalization

◊ Abstract

Data Normalization:

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

There are two primary advantages of having a highly normalized data schema:

  Increased consistency:
  Information is stored in one place and one place only, reducing the possibility of inconsistent data.

  Easier object-to-data mapping:
  Highly-normalized data schemas in general are closer conceptually to object-oriented schemas because the object-oriented goals of promoting high cohesion and loose coupling between classes results in similar solutions (at least from a data point of view).

Three most common forms of normalization

    First normal form (1NF),
    Second normal form (2NF),
    Third normal form (3NF))

 The motive of this assignment is to understand normalization of Database.
 We will be using raw dataset for FIFA 2019 Players. We will normalize the database and prepare Entity relationship diagram and concetula model of the same.

 Below are the columns we extracted from Database:

 'ID',
 'Name',
 'Age',
 'Photo',
 'Nationality',
 'Flag',
 'Overall',
 'Potential',
 'Club',
 'Club Logo',
 'Value',
 'Wage',
 'Special',
 'Preferred Foot',
 'International Reputation',

```
'Weak Foot',
'Skill Moves',
'Work Rate',
'Body Type',
'Real Face',
'Position',
'Jersey Number',
'Joined',
'Loaned From',
'Contract Valid Until',
'Height',
'Weight',
'LS',
'ST',
'RS',
'LW',
'LF',
'CF',
'RF',
'RW',
'LAM',
'CAM',
'RAM',
'LM',
'LCM',
'CM',
'RCM',
'RM',
'LWB',
'LDM',
'CDM',
'RDM',
'RWB',
'LB',
'LCB',
'CB',
'RCB',
'RB',
'Crossing',
'Finishing',
'HeadingAccuracy',
'ShortPassing',
'Volleys',
'Dribbling',
'Curve',
'FKAccuracy',
'LongPassing',
'BallControl',
'Acceleration',
'SprintSpeed',
'Agility',
'Reactions',
'Balance',
'ShotPower',
'Jumping',
'Stamina',
'Strength',
```

```
'LongShots',
'Aggression',
'Interceptions',
'Positioning',
'Vision',
'Penalties',
'Composure',
'Marking',
'StandingTackle',
'SlidingTackle',
'GKDiving',
'GKHandling',
'GKKicking',
'GKPositioning',
'GKReflexes',
'Release Clause'
```

In [1]:
```
import requests
import pandas as pd
import json
import os
```

In [2]:
```
FIFA_Player_Data = pd.read_csv("FIFA 19 Player Database.csv")
list(FIFA_Player_Data.columns)
```

Out[2]:
```
['Unnamed: 0',
 'ID',
 'Name',
 'Age',
 'Photo',
 'Nationality',
 'Flag',
 'Overall',
 'Potential',
 'Club',
 'Club Logo',
 'Value',
 'Wage',
 'Special',
 'Preferred Foot',
 'International Reputation',
 'Weak Foot',
 'Skill Moves',
 'Work Rate',
```

◊◊ Normalization - 1NF

Form The first normalization should eliminate the repeate record and assign a unique identifier for every record. In my FIFA Player data example given below, the Table has a Unique Identifier as ID with no repeated records for that ID.

In [3]:
```
FIFA_Player_Data
```

Out[3]:

	Unnamed: 0	ID	Name	Age	Photo	Na
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	A
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	
...	...	...	...	...	...	
18202	18202	238813	J. Lundstram	19	https://cdn.sofifa.org/players/4/19/238813.png	

◊ Checking for repeate record

In [4]:
```
FIFA_Player_Data.duplicated()
```

Out[4]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

◊◊ Normalization - 2NF

The second normal form requires that it meets the first normal form and should be non-prime attribute of a relation, which means the attribute is not a part of any candidate key of the relation. In last table, FIFA Player data can be divide to Multiple tables using second normal form, As shown

below. And for each table we will validate the condition for 1NF.

In [5]:
```
Player_Personal_Info = FIFA_Player_Data[['ID','Name','Age','Nationality','Hei
                                        'Skill Moves','Preferred Foot','Posi
Player_Personal_Info.to_csv("Player_Personal_Info.csv",encoding="utf-8",index
Player_Personal_Info = pd.read_csv("Player_Personal_Info.csv")
Player_Personal_Info
```

Out[5]:

	ID	Name	Age	Nationality	Height	Weight	Body Type	Skill Moves	Preferred Foot	P
0	158023	L. Messi	31	Argentina	5'7	159lbs	Messi	4.0	Left	
1	20801	Cristiano Ronaldo	33	Portugal	6'2	183lbs	C. Ronaldo	5.0	Right	
2	190871	Neymar Jr	26	Brazil	5'9	150lbs	Neymar	5.0	Right	
3	193080	De Gea	27	Spain	6'4	168lbs	Lean	1.0	Right	
4	192985	K. De Bruyne	27	Belgium	5'11	154lbs	Normal	4.0	Right	
...	...	...	...	...	...	...	...	...	...	...
18202	238813	J. Lundstram	19	England	5'9	134lbs	Lean	2.0	Right	
18203	243165	N. Christoffersson	19	Sweden	6'3	170lbs	Normal	2.0	Right	
18204	241638	B. Worman	16	England	5'8	148lbs	Normal	2.0	Right	
18205	246268	D. Walker-Rice	17	England	5'10	154lbs	Lean	2.0	Right	
18206	246269	G. Nugent	16	England	5'10	176lbs	Lean	2.0	Right	

18207 rows × 10 columns

◊ Checking for repeate record:

In [6]:
```
Player_Personal_Info.duplicated()
```

Out[6]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

In [7]:
```
Player_Ranking = FIFA_Player_Data[['ID','Overall','Potential','International
Player_Ranking.to_csv("Player_Ranking.csv",encoding="utf-8",index=False)
Player_Ranking = pd.read_csv("Player_Ranking.csv")
Player_Ranking
```

Out[7]:

	ID	Overall	Potential	International Reputation
0	158023	94	94	5.0
1	20801	94	94	5.0
2	190871	92	93	5.0
3	193080	91	93	4.0
4	192985	91	92	4.0
...	...	...	...	...
18202	238813	47	65	1.0
18203	243165	47	63	1.0
18204	241638	47	67	1.0
18205	246268	47	66	1.0
18206	246269	46	66	1.0

18207 rows × 4 columns

◊ Checking for repeate record:

In [8]:
```
Player_Ranking.duplicated()
```

Out[8]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

In [9]:
```
Player_Club_Info = FIFA_Player_Data[['ID','Club','Value','Wage',
                                     'Jersey Number','Joined','Real Face']]
Player_Club_Info.to_csv("Player_Club_Info.csv",encoding="utf-8",index=False)
Player_Club_Info = pd.read_csv("Player_Club_Info.csv")
Player_Club_Info
```

Out[9]:

	ID	Club	Value	Wage	Jersey Number	Joined	Real Face
0	158023	FC Barcelona	€110.5M	€565K	10.0	Jul 1, 2004	Yes
1	20801	Juventus	€77M	€405K	7.0	Jul 10, 2018	Yes
2	190871	Paris Saint-Germain	€118.5M	€290K	10.0	Aug 3, 2017	Yes
3	193080	Manchester United	€72M	€260K	1.0	Jul 1, 2011	Yes
4	192985	Manchester City	€102M	€355K	7.0	Aug 30, 2015	Yes
...	...	...	...	...	...	...	...
18202	238813	Crewe Alexandra	€60K	€1K	22.0	May 3, 2017	No
18203	243165	Trelleborgs FF	€60K	€1K	21.0	Mar 19, 2018	No
18204	241638	Cambridge United	€60K	€1K	33.0	Jul 1, 2017	No
18205	246268	Tranmere Rovers	€60K	€1K	34.0	Apr 24, 2018	No
18206	246269	Tranmere Rovers	€60K	€1K	33.0	Oct 30, 2018	No

18207 rows × 7 columns

◊ Checking for repeate record:

In [10]:
```
Player_Club_Info.duplicated()
```

Out[10]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

In [11]:
```
Player_Contract = FIFA_Player_Data[['ID','Contract Valid Until','Loaned From'
Player_Contract.to_csv("Player_Contract.csv",encoding="utf-8",index=False)
Player_Contract = pd.read_csv("Player_Contract.csv")
Player_Contract
```

Out[11]:

	ID	Contract Valid Until	Loaned From	Release Clause	Work Rate
0	158023	2021	NaN	€226.5M	Medium/ Medium
1	20801	2022	NaN	€127.1M	High/ Low
2	190871	2022	NaN	€228.1M	High/ Medium
3	193080	2020	NaN	€138.6M	Medium/ Medium
4	192985	2023	NaN	€196.4M	High/ High
...	...	...	...	...	...
18202	238813	2019	NaN	€143K	Medium/ Medium
18203	243165	2020	NaN	€113K	Medium/ Medium
18204	241638	2021	NaN	€165K	Medium/ Medium
18205	246268	2019	NaN	€143K	Medium/ Medium
18206	246269	2019	NaN	€165K	Medium/ Medium

18207 rows × 5 columns

◊ Checking for repeate record:

In [12]:
```
Player_Contract.duplicated()
```

Out[12]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

In [13]:
```
Player_Position_Stats = FIFA_Player_Data[['ID','LS','ST','RS','LW','LF','CF',
                                          'RAM','LM','LCM','CM','RCM','RM','L
                                          'RWB','LB','LCB','CB','RCB','RB']]
Player_Position_Stats.to_csv("Player_Position_Stats.csv",encoding="utf-8",inc
Player_Position_Stats = pd.read_csv("Player_Position_Stats.csv")
Player_Position_Stats
```

Out[13]:

	ID	LS	ST	RS	LW	LF	CF	RF	RW	LAM	...	LWB	LDM	CDM
0	158023	88+2	88+2	88+2	92+2	93+2	93+2	93+2	92+2	93+2	...	64+2	61+2	61+
1	20801	91+3	91+3	91+3	89+3	90+3	90+3	90+3	89+3	88+3	...	65+3	61+3	61+
2	190871	84+3	84+3	84+3	89+3	89+3	89+3	89+3	89+3	89+3	...	65+3	60+3	60+
3	193080	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
4	192985	82+3	82+3	82+3	87+3	87+3	87+3	87+3	87+3	88+3	...	77+3	77+3	77+
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18202	238813	42+2	42+2	42+2	44+2	44+2	44+2	44+2	44+2	45+2	...	44+2	45+2	45+
18203	243165	45+2	45+2	45+2	39+2	42+2	42+2	42+2	39+2	40+2	...	30+2	31+2	31+
18204	241638	45+2	45+2	45+2	45+2	46+2	46+2	46+2	45+2	44+2	...	34+2	30+2	30+
18205	246268	47+2	47+2	47+2	47+2	46+2	46+2	46+2	47+2	45+2	...	36+2	32+2	32+
18206	246269	43+2	43+2	43+2	45+2	44+2	44+2	44+2	45+2	45+2	...	46+2	46+2	46+

18207 rows × 27 columns

◊ Checking for repeate record:

In [14]:
```
Player_Position_Stats.duplicated()
```

Out[14]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

In [15]:
```
Player_Skill_Stats = FIFA_Player_Data[['ID','Crossing','Finishing','HeadingAc
                                      'Dribbling','Curve','FKAccuracy','Long
                                      'Acceleration','SprintSpeed','Agility'
                                      'ShotPower','Jumping','Stamina','Stren
                                      'Aggression','Interceptions','Position
                                      'Composure','Marking','StandingTackle'
Player_Skill_Stats.to_csv("Player_Skill_Stats.csv",encoding="utf-8",index=Fal
Player_Skill_Stats = pd.read_csv("Player_Skill_Stats.csv")
Player_Skill_Stats
```

Out[15]:

	ID	Crossing	Finishing	HeadingAccuracy	ShortPassing	Volleys	Dribbling	Cu
0	158023	84.0	95.0	70.0	90.0	86.0	97.0	9
1	20801	84.0	94.0	89.0	81.0	87.0	88.0	8
2	190871	79.0	87.0	62.0	84.0	84.0	96.0	8
3	193080	17.0	13.0	21.0	50.0	13.0	18.0	2
4	192985	93.0	82.0	55.0	92.0	82.0	86.0	8
...	...	...	...	...	...	...	...	
18202	238813	34.0	38.0	40.0	49.0	25.0	42.0	3
18203	243165	23.0	52.0	52.0	43.0	36.0	39.0	3
18204	241638	25.0	40.0	46.0	38.0	38.0	45.0	3
18205	246268	44.0	50.0	39.0	42.0	40.0	51.0	3
18206	246269	41.0	34.0	46.0	48.0	30.0	43.0	4

◊ Checking for repeate record:

In [16]:
```
Player_Skill_Stats.duplicated()
```

Out[16]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

In [17]:
```
Player_GK_Stats = FIFA_Player_Data[['ID','GKDiving','GKHandling','GKKicking',
Player_GK_Stats.to_csv("Player_GK_Stats.csv",encoding="utf-8",index=False)
Player_GK_Stats = pd.read_csv("Player_GK_Stats.csv")
Player_GK_Stats
```

Out[17]:

	ID	GKDiving	GKHandling	GKKicking	GKPositioning	GKReflexes
0	158023	6.0	11.0	15.0	14.0	8.0
1	20801	7.0	11.0	15.0	14.0	11.0
2	190871	9.0	9.0	15.0	15.0	11.0
3	193080	90.0	85.0	87.0	88.0	94.0
4	192985	15.0	13.0	5.0	10.0	13.0
...	...	...	...	...	...	...
18202	238813	10.0	13.0	7.0	8.0	9.0
18203	243165	10.0	9.0	9.0	5.0	12.0
18204	241638	6.0	5.0	10.0	6.0	13.0
18205	246268	14.0	6.0	14.0	8.0	9.0
18206	246269	10.0	15.0	9.0	12.0	9.0

18207 rows × 6 columns

◊ Checking for repeate record:

In [18]:
```
Player_GK_Stats.duplicated()
```

Out[18]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202    False
18203    False
18204    False
18205    False
18206    False
Length: 18207, dtype: bool
```

◊◊ Normalization - 3NF

The third normal form requires that it meets the second normal form and should be no any prime attribute of a relation, which means all the attributes in a table are only decided by the candidate keys of that relation. As show below in the Club_info Table, attributes Club and Club Logo stands as a Candidate key for this table same for the Nationality_Info table as well.

In [19]:
```
Club_Info = FIFA_Player_Data[['Club','Club Logo']]
Club_Info.to_csv("Club_Info.csv",encoding="utf-8",index=False)
Club_Info = pd.read_csv("Club_Info.csv")
Club_Info
```

Out[19]:

	Club	Club Logo
0	FC Barcelona	https://cdn.sofifa.org/teams/2/light/241.png
1	Juventus	https://cdn.sofifa.org/teams/2/light/45.png
2	Paris Saint-Germain	https://cdn.sofifa.org/teams/2/light/73.png
3	Manchester United	https://cdn.sofifa.org/teams/2/light/11.png
4	Manchester City	https://cdn.sofifa.org/teams/2/light/10.png
...	...	...
18202	Crewe Alexandra	https://cdn.sofifa.org/teams/2/light/121.png
18203	Trelleborgs FF	https://cdn.sofifa.org/teams/2/light/703.png
18204	Cambridge United	https://cdn.sofifa.org/teams/2/light/1944.png
18205	Tranmere Rovers	https://cdn.sofifa.org/teams/2/light/15048.png
18206	Tranmere Rovers	https://cdn.sofifa.org/teams/2/light/15048.png

18207 rows × 2 columns

◊ Checking for repeate record:

In [20]:
```
Club_Info.duplicated()
```

Out[20]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202     True
18203     True
18204     True
18205     True
18206     True
Length: 18207, dtype: bool
```

As seen in the above Club_Info Table we have duplicates values wich does not satisfy the validation for the table to be in 1NF.

In Order to validate our table to be in 1NF we clean the table by removing the duplicate vaules as shown below.

In [21]:
```
# sorting by first name
Club_Info.sort_values("Club", inplace = True)
# dropping ALL duplicte values
Club_Info.drop_duplicates(subset ="Club",
                     keep = "last", inplace = True)
Club_Info.to_csv("Club_Info.csv",encoding="utf-8",index=False)
Club_Info_Normalized = pd.read_csv("Club_Info.csv")
Club_Info_Normalized
```

Out[21]:

	Club	Club Logo
0	SSV Jahn Regensburg	https://cdn.sofifa.org/teams/2/light/543.png
1	1. FC Heidenheim 1846	https://cdn.sofifa.org/teams/2/light/111235.png
2	1. FC Kaiserslautern	https://cdn.sofifa.org/teams/2/light/29.png
3	1. FC Köln	https://cdn.sofifa.org/teams/2/light/31.png
4	1. FC Magdeburg	https://cdn.sofifa.org/teams/2/light/110588.png
...	...	...
647	Çaykur Rizespor	https://cdn.sofifa.org/teams/2/light/101037.png
648	Örebro SK	https://cdn.sofifa.org/teams/2/light/705.png
649	Östersunds FK	https://cdn.sofifa.org/teams/2/light/113173.png
650	Śląsk Wrocław	https://cdn.sofifa.org/teams/2/light/111092.png
651	NaN	https://cdn.sofifa.org/flags/159.png

652 rows × 2 columns

◊ Checking for repeate record:

In [22]:
```
Club_Info_Normalized.duplicated()
```

Out[22]:
```
0      False
1      False
2      False
3      False
4      False
       ...
647    False
648    False
649    False
650    False
651    False
Length: 652, dtype: bool
```

In [23]:
```
Nationality_Info = FIFA_Player_Data[['Nationality','Flag']]
Nationality_Info.to_csv("Nationality_Info.csv",encoding="utf-8",index=False)
Nationality_Info = pd.read_csv("Nationality_Info.csv")
Nationality_Info
```

Out[23]:

	Nationality	Flag
0	Argentina	https://cdn.sofifa.org/flags/52.png
1	Portugal	https://cdn.sofifa.org/flags/38.png
2	Brazil	https://cdn.sofifa.org/flags/54.png
3	Spain	https://cdn.sofifa.org/flags/45.png
4	Belgium	https://cdn.sofifa.org/flags/7.png
...	...	...
18202	England	https://cdn.sofifa.org/flags/14.png
18203	Sweden	https://cdn.sofifa.org/flags/46.png
18204	England	https://cdn.sofifa.org/flags/14.png
18205	England	https://cdn.sofifa.org/flags/14.png
18206	England	https://cdn.sofifa.org/flags/14.png

18207 rows × 2 columns

◊ Checking for repeate record:

In [24]:
```
Nationality_Info.duplicated()
```

Out[24]:
```
0        False
1        False
2        False
3        False
4        False
         ...
18202     True
18203     True
18204     True
18205     True
18206     True
Length: 18207, dtype: bool
```

We observe the same in our Nationality_Info Table, as we have duplicates values wich does not satisfy the validation for the table to be in 1NF.

In Order to validate our table to be in 1NF we clean the table by removing the duplicate vaules as shown below.

In [25]:
```
# sorting by first name
Nationality_Info.sort_values("Nationality", inplace = True)
# dropping ALL duplicte values
Nationality_Info.drop_duplicates(subset ="Nationality",
                     keep = "last", inplace = True)
Nationality_Info.to_csv("Nationality_Info.csv",encoding="utf-8",index=False)
Nationality_Info = pd.read_csv("Nationality_Info.csv")
Nationality_Info
```

Out[25]:

	Nationality	Flag
0	Afghanistan	https://cdn.sofifa.org/flags/149.png
1	Albania	https://cdn.sofifa.org/flags/1.png
2	Algeria	https://cdn.sofifa.org/flags/97.png
3	Andorra	https://cdn.sofifa.org/flags/2.png
4	Angola	https://cdn.sofifa.org/flags/98.png
...	...	...
159	Uzbekistan	https://cdn.sofifa.org/flags/191.png
160	Venezuela	https://cdn.sofifa.org/flags/61.png
161	Wales	https://cdn.sofifa.org/flags/50.png
162	Zambia	https://cdn.sofifa.org/flags/147.png
163	Zimbabwe	https://cdn.sofifa.org/flags/148.png

164 rows × 2 columns

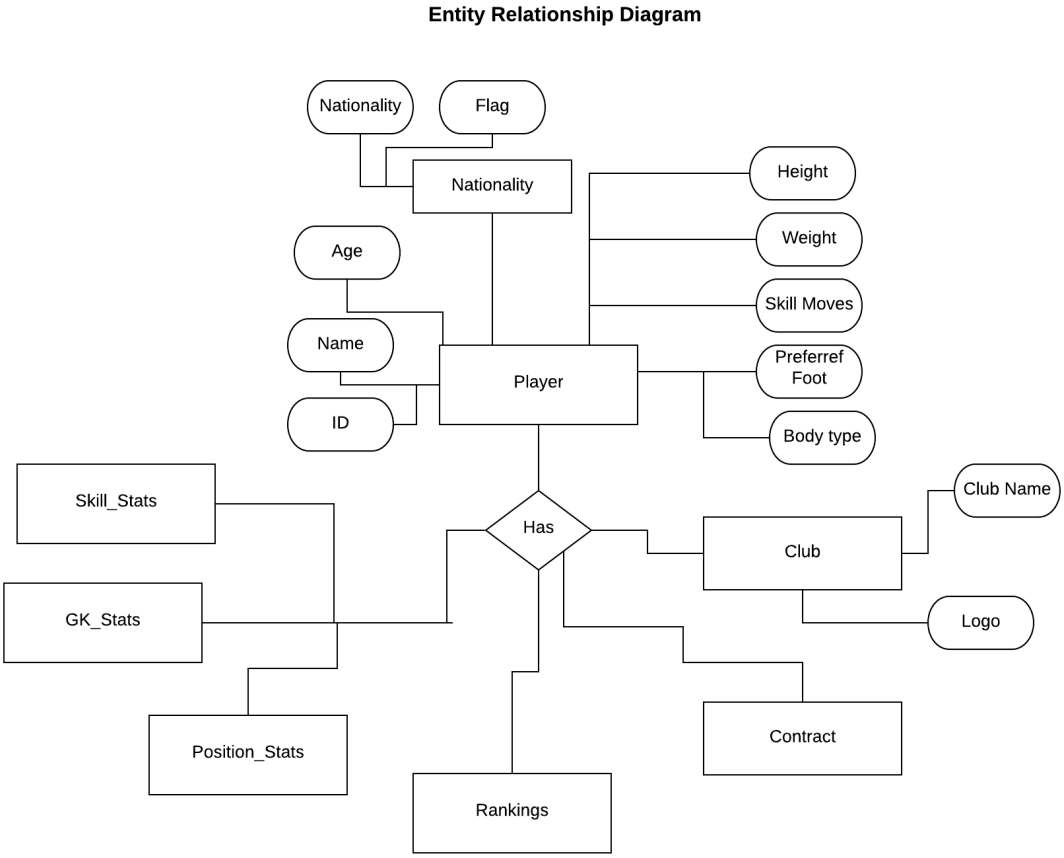
◊ Checking for repeate record:

In [26]:
```
Nationality_Info.duplicated()
```

Out[26]:
```
0      False
1      False
2      False
3      False
4      False
       ...
159    False
160    False
161    False
162    False
163    False
Length: 164, dtype: bool
```

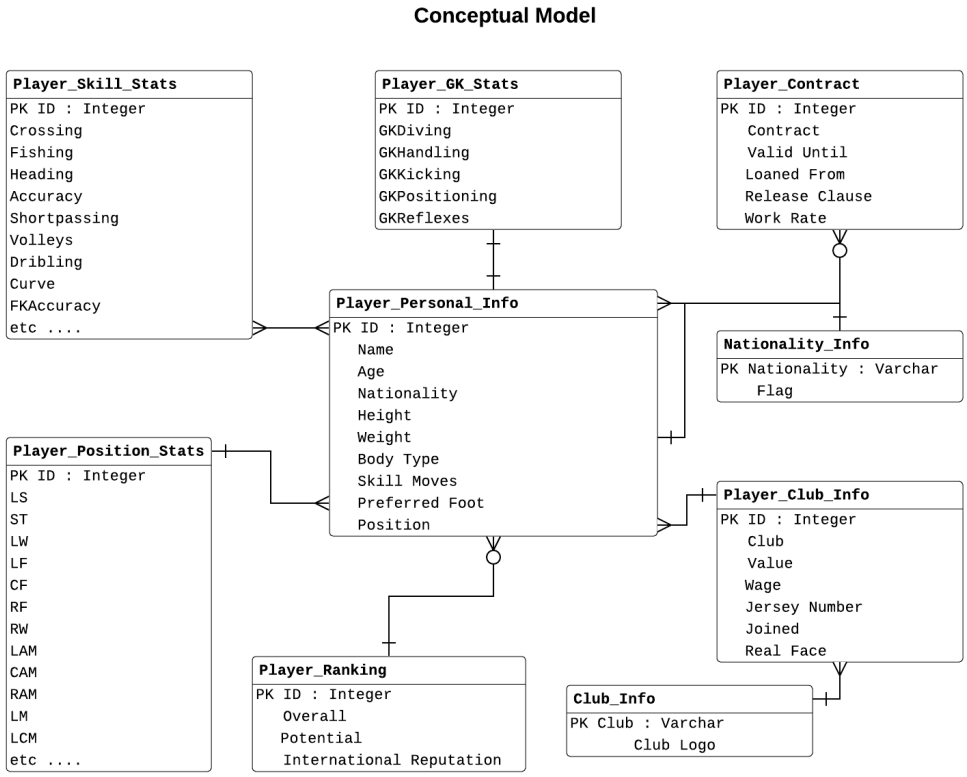
In [27]:
```
### ◊ ENTITY RELATIONSHIP DIAGRAM:
#Entity Relationship model using Image library
from IPython.display import Image
Image("DMDD-Assgnment1-ERD.png")
```

Out[27]:



In [28]:
```
### ◊ CONCEPTUAL:
from IPython.display import Image
Image("DMDD-Assgnment2-Conceptual Model.png")
```

Out[28]:



◊ AUDIT VALIDITY/ACCURACY:

With the above data we validate the accuracy by checking if any column had null value or junk value. We used few data validation functions like duplicate, del and lambda functions, with these functions or commands on the affected rows and columns, the not null values were validated from the database which gave a valid and accuarate data.

◊ AUDIT COMPLETNESS:

In real world, a list of all FIFA players will be displayed or presented, similarly when we compare it with above data too, we get proper real time data showing correct information for all FIFA players.

◊ AUDIT CONSISTENCY/UNIFORMITY:

The datasets which we used in this assignment shows a uniform relationship between each of the dataset that we extracted since they are linked to each other by a common attribute player ID.

◊ REPORT:

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

Database normalization is a process used to organize a database into tables and columns. The idea is that a table should be about a specific topic and that and only supporting topics included.

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normalization is used for mainly two purposes

1. Eliminating redundant(useless) data.
2. Ensuring data dependencies make sense i.e. data is logically stored.

Normalization Rule Normalization rules are divided into the following normal forms:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF
5. Fourth Normal Form

A Candidate Key can be any column or a combination of columns that can qualify as unique key in database. There can be multiple Candidate Keys in one table. A Primary Key is a column or a combination of columns that uniquely identify a record. So basically you can choose any Candidate Key as the Primary Key.

First normal form (1NF) • Each table has a primary key: minimal set of attributes which can uniquely identify a record • The values in each column of a table are atomic (No multi-value attributes allowed). • There are no repeating groups: two columns do not store similar information in the same table.

Second normal form (2NF) • All requirements for 1st NF must be met. • No partial dependencies. • No calculated data

Third normal form (3NF) • All requirements for 2nd NF must be met. • Eliminate fields that do not directly depend on the primary key; that is no transitive dependencies.

◊ CONCLUSION:

We understood Databse normalization which is as below: Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

Database normalization is a process used to organize a database into tables and columns.
The idea is that a table should be about a specific topic and that and only supporting topics included.

In the process, we got familiar to conceptual database model which is defined as a structured view of the data required to support a processes, record events, and track related performance measures. This model focuses on identifying the data used in the business but not its processing flow or physical characteristics.

Also we got acquianted with entity relationship diagram (ERD), also known as an entity relationship model, which is a graphical representation that depicts relationships among people, objects, places, concepts or events (data) within a system. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout--- an organization.

◊ CONTRIBUTION:

Self efforts : 30%
External source: 30%
Guidance by the Professor : 30%
Guidance by Teaching Assistant: 10%

◊ CITATIONS:

Below are the sources from where we reffered and pulled code, data.

https://www.themoviedb.org/ (https://www.themoviedb.org/)
https://www.crummy.com/software/BeautifulSoup/bs4/doc/
(https://www.crummy.com/software/BeautifulSoup/bs4/doc/)
https://www.pythonforbeginners.com/beautifulsoup/beautifulsoup-4-python
(https://www.pythonforbeginners.com/beautifulsoup/beautifulsoup-4-python)
https://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html
(https://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html)
https://www.tutorialspoint.com/python_pandas/python_pandas_function_application.htm
(https://www.tutorialspoint.com/python_pandas/python_pandas_function_application.htm)
https://pandas.pydata.org/ (https://pandas.pydata.org/)

◊ LICENSE:

Copyright 2020 Srushti Dhamangaonkar

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.