MongoDB

Javier Ramírez Alarcón

Máster Big Data, Data Science & Inteligencia Artificial 2023-2024

Contenido

Introducción

El objetivo del presente trabajo buscar aplicar todos los conocimientos adquiridos en el máster y bajo nuestra propia curiosidad para garantizar el dominio de la herramienta.

Para ello se ha seleccionado las bases de datos relacionados al ámbito del futbol, dentro de las cuales destacan las fuentes: players, players_valuations, appearences, etc. A las cuales se le aplican diversas transformaciones para poder generar mayor valor a dichas fuentes y obtener análisis que tengan mayor impacto. Asimismo, el presente trabajo ha incorporado el uso de las bases de datos MongoDB con el lenguaje de programación Python mediante la librería pymongo para aprovechar los beneficios tanto de Mongo como los de Python.

Dentro de los campos almacenados en cada colección se detallarán a continuación algunos que hemos usado primordialmente:

» player_id: Identificador del jugador

» First_name: Primer nombre del jugador

» last_name: Apellido del jugador

» name: Nombre complete del jugador

» last_season: Ultima temporada que jugó en Europa

» current_club_id: Identificador del club actual en el cual milita el jugador

» player_code: Código de jugador

» country_of_birth: País de nacimiento del jugador

» city_of_birth: Ciudad de nacimiento del jugador

» country_of_citizenship: País donde el jugador posee ciudadanáa

» date_of_birth: Fecha de nacimiento:

» sub_position: Subposición de jugador

» position: Posición del jugador

» foot: Pierna principal del jugador

» height_in_cm: Altura del jugador en cm

» market_value_in_eur: Valor de mercado del jugador en un periodo determinado

» highest_market_value_in_eur: Máximo valor de mercado que alcanzó el jugador

» contract_expiration_date: Fecha de expiración de contrato del jugador

» image_url: Imagen del jugador

» current_club_domestic_competition_id: Identificador de la competición del actual club del jugador

» current_club_name: Nombre del actual club del jugador

» current_club_id: Identificador del actual club del jugador

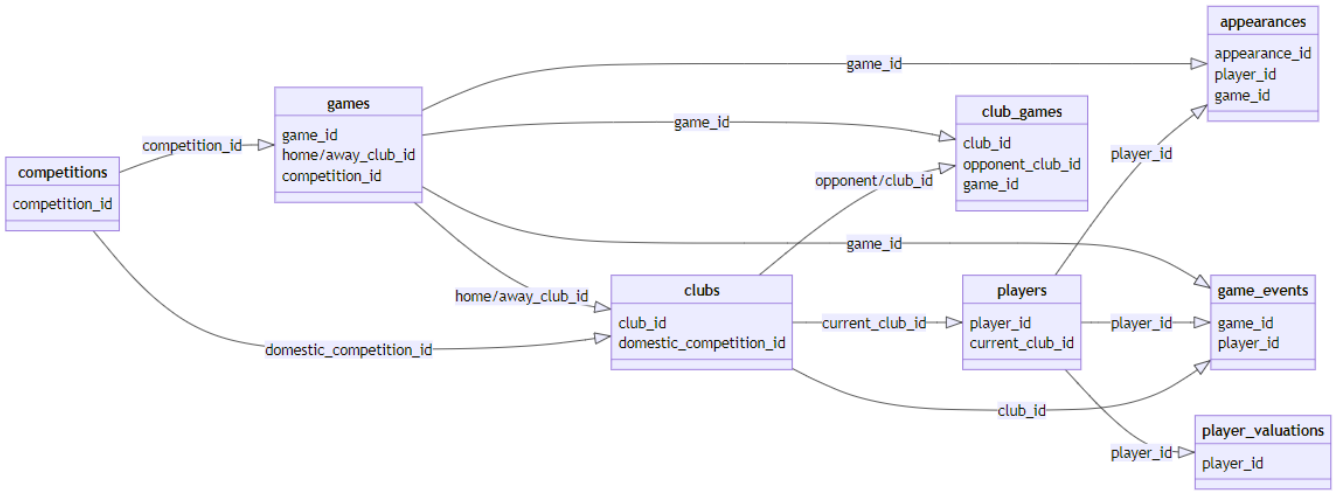
» yellow_cards: Cantidad de tarjetas amarillas

» red_cards: Cantidad de tarjetas rojas

» goals: Cantidad de goles anotados

» assists: Cantidad de asistencias brindadas

» minutes_played: Cantidad de minutos jugados



1. Cargar / Importar dataset

a. Importación de Librerías

Como se mencionó inicialmente en el presente proyecto se hace uso de la librería pymongo para tener una mayor eficiencia en el manejo de datos de la base de datos de Mongo. Por lo cual iniciaremos importando las librerías que serán utilizadas tanto para usar pymongo, manejos de datos, lectura de imágenes, generación de estilo de lectura, etc.

```
[1]: import csv
     import pandas as pd
     import json
     from pymongo import MongoClient
     from PIL import Image
     import requests
     import re
     from datetime import datetime
     from prettytable import PrettyTable
```

b. Crear conexión con mongo

Procederemos a generar la conexión entre Python y Mongo.

```
[3]: MONGO_URI = 'mongodb://localhost'

[4]: client = MongoClient(MONGO_URI)
```

Una vez generada la conexión buscaremos listar las colecciones existentes en la base de datos de Mongo.

```
[5]: print(client.list_database_names())
     ['admin', 'clases', 'config', 'local', 'nuevo']
```

c. Definir base de datos y colecciones

Continuamos con la generación de la base de datos y las colecciones que serán de nuestra utilidad para nuestros futuros análisis.

```
[5]: db = client['TareaNoSQL']

[6]: cplayers = db['Players']
     cplayervaluation = db['PlayerValuations']
     cclubs = db['Clubs']
     cappearances = db['Appearances']
```

d. Importación de datos

A continuación, procederemos a realizar la importación de datos desde los ficheros planos hacia la base de datos y las colecciones correspondientes.

```
[12]: players = pd.read_csv("C:/Users/jr119/Downloads/transfermarkt/players.csv")
      player_valuations = pd.read_csv("C:/Users/jr119/Downloads/transfermarkt/player_valuations.csv")
      clubs = pd.read_csv("C:/Users/jr119/Downloads/transfermarkt/clubs.csv")
      appearances = pd.read_csv("C:/Users/jr119/Downloads/transfermarkt/appearances.csv")

[13]: df_players = players.to_dict(orient = "records")
      df_player_valuations = player_valuations.to_dict(orient = "records")
      df_clubs = clubs.to_dict(orient = "records")
      df_appearances = appearances.to_dict(orient = "records")
```

```
[14]: db.cplayers.insert_many(df_players)
      db.cplayervaluation.insert_many(df_player_valuations)
      db.cclubs.insert_many(df_clubs)
      db.cappearances.insert_many(df_appearances)
```

2. Ejercicios sobre inserción, actualización, proyección y filtrado

a. Ejercicios sobre inserción

En este ejercicio se busca agregar un documento como tal en una colección y agregar un campo a toda una colección para evidenciar el manejo de los comandos de inserción en la base de datos de Mongo.

Por lo cual definiremos los datos que poseerá el documento que deseamos insertar en la colección, agregaremos dicho documento a la colección y luego buscaremos dicho documento en toda la colección para evidenciar su correcta inserción.

```
[17]: document_to_insert = [
          {
              "player_id": '587698',
              "first_name": 'Paolo',
              "last_name": 'Guerrero',
              "name": 'Paolo Guerrero',
              "last_season": 2023,
              "current_club_id": '14172',
              "player_code": 'paolo-guerrero',
              "country_of_birth": 'Peru',
              "city_of_birth": 'Lima',
              "country_of_citizenship": 'Peru',
              "date_of_birth": '1984-01-01',
              "sub_position": 'Centre-Forward',
              "position": 'Attack',
              "foot": 'right',
              "height_in_cm": 185,
              "market_value_in_eur": 35000,
              "highest_market_value_in_eur": 8000000,
              "contract_expiration_date": '2023-12-31 00:00:00',
              "agent_name": '',
              "image_url": 'https://img.a.transfermarkt.technology/portrait/header/2989-1687728679.jpg?lm=1',
              "url": 'https://www.transfermarkt.es/paolo-guerrero/profil/spieler/2989',
              "current_club_domestic_competition_id": 'ECUA1',
              "current_club_name": 'LDU Quito'
          }
      ]
```

```
[18]: result = db.cplayers.insert_many(document_to_insert)
```

```
[19]: for inserted_id in result.inserted_ids:
          print(f"Documento insertado con ID: {inserted_id}")
```

```
Documento insertado con ID: 65419d295af52a865c5f0217
```

```
[20]: for i in db.cplayers.find({"name":"Paolo Guerrero"},{}):
          print(i)
```

```
{'_id': ObjectId('65419d295af52a865c5f0217'), 'player_id': '587698', 'first_name': 'Paolo', 'last_name': 'Guerrero', 'name': 'Paolo Guerrero', 'last_season': 2023, 'current_club_id': '14172', 'player_code': 'paolo-guerrero', 'country_of_birth': 'Peru', 'city_of_birth': 'Lima', 'country_of_citizenship': 'Peru', 'date_of_birth': '1984-01-01', 'sub_position': 'Centre-Forward', 'position': 'Attack', 'foot': 'right', 'height_in_cm': 185, 'market_value_in_eur': 35000, 'highest_market_value_in_eur': 8000000, 'contract_expiration_date': '2023-12-31 00:00:00', 'agent_name': '', 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/2989-1687728679.jpg?lm=1', 'url': 'https://www.transfermarkt.es/paolo-guerrero/profil/spieler/2989', 'current_club_domestic_competition_id': 'ECUA1', 'current_club_name': 'LDU Quito'}
```

Por otro lado, se busca realizar el agregado de un nuevo campo a toda la colección “Players” por lo cual generaremos el campo que corresponde a la edad de los jugadores “age” calculada en base al campo “date_of_birth”, tomando en cuenta que usamos la librería datetime y un manejo de errores ya que se tienen jugadores de los cuales se desconocen el valor del campo “date_of_birth”.

```
[21]: for document in db.cplayers.find({}):
          date_of_birth = document.get('date_of_birth')

          if date_of_birth:
              try:
                  birth_date = datetime.strptime(str(date_of_birth), '%Y-%m-%d')
                  today = datetime.now()
                  age = today.year - birth_date.year - ((today.month, today.day) < (birth_date.month, birth_date.day))
              except ValueError:
                  age = None
          else:
              age = None
          db.cplayers.update_one({'_id': document['_id']}, {'$set': {'age': age}})
```

```
[22]: for i in db.cplayers.find({}).limit(1):
          print(i)
```

{'_id': ObjectId('6542a6a0aa0ab5da15484310'), 'player_id': 598, 'first_name': 'Timo', 'last_name': 'Hildebrand', 'name': 'Timo Hildebrand', 'last_season': 2014, 'current_club_id': 24, 'player_code': 'timo-hildebrand', 'country_of_birth': 'Germany', 'city_of_birth': 'Worms', 'country_of_citizenship': 'Germany', 'date_of_birth': '1979-04-05', 'sub_position': 'Goalkeeper', 'position': 'Goalkeeper', 'foot': nan, 'height_in_cm': nan, 'market_value_in_eur': nan, 'highest_market_value_in_eur': 10000000.0, 'contract_expiration_date': nan, 'agent_name': nan, 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/598-1540373444.jpg?lm=1', 'url': 'https://www.transfermarkt.co.uk/timo-hildebrand/profil/spieler/598', 'current_club_domestic_competition_id': 'L1', 'current_club_name': 'Eintracht Frankfurt', 'age': 44}

De igual manera generaremos otro campo que nos servirá en un futuro análisis el cual es calculado en base al campo “age” creado previamente, dicho campo es “ageRange” que nos indica a que rango de edad pertenece cada jugador, debemos tomar en cuenta que estamos filtrando a los jugadores que tienen edad igual a “null” o “None” para evitar algún tipo de warning o error.

```
[43]: for document in db.cplayers.find({"age":{'$ne': None}},{}):
          age = document['age']
          if age <= 18:
              document['ageRange'] = '1. Menor a 18'
          elif 18 < age <= 23:
              document['ageRange'] = '2. De 18 a 23'
          elif 23 < age <= 27:
              document['ageRange'] = '3. De 23 a 27'
          elif 27 < age <= 30:
              document['ageRange'] = '4. De 27 a 30'
          elif 30 < age <= 35:
              document['ageRange'] = '5. De 30 a 35'
          else:
              document['ageRange'] = '6. Mayor a 35'
          db.cplayers.update_one({'_id': document['_id']}, {'$set': {'ageRange': document['ageRange']}})
```

```
[24]: for i in db.cplayers.find({}).limit(1):
          print(i)
```

{'_id': ObjectId('6542a6a0aa0ab5da15484310'), 'player_id': 598, 'first_name': 'Timo', 'last_name': 'Hildebrand', 'name': 'Timo Hildebrand', 'last_season': 2014, 'current_club_id': 24, 'player_code': 'timo-hildebrand', 'country_of_birth': 'Germany', 'city_of_birth': 'Worms', 'country_of_citizenship': 'Germany', 'date_of_birth': '1979-04-05', 'sub_position': 'Goalkeeper', 'position': 'Goalkeeper', 'foot': nan, 'height_in_cm': nan, 'market_value_in_eur': nan, 'highest_market_value_in_eur': 10000000.0, 'contract_expiration_date': nan, 'agent_name': nan, 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/598-1540373444.jpg?lm=1', 'url': 'https://www.transfermarkt.co.uk/timo-hildebrand/profil/spieler/598', 'current_club_domestic_competition_id': 'L1', 'current_club_name': 'Eintracht Frankfurt', 'age': 44, 'ageRange': '6. Mayor a 35'}

b. Ejercicios sobre actualización

En el siguiente ejercicio buscamos poder actualizar documentos de manera en que podamos evidenciar que en caso que sea necesario realizar alguna actualización de algún campo este proceso es viable.

Para el presente ejercicio hemos detectado que para el jugador Cristiano Ronaldo el campo “first_name” tiene el valor “nan” el cual es incorrecto, por el lado del campo “last_name” posee el valor del nombre del jugador y el campo “name” posee el mismo valor que el campo anterior mencionado para lo cual para estandarizar los datos procederemos a actualizar dichos campos con los valores que deberían ser los correctos. Para lo cual evidenciaremos los valores iniciales en contraste con los valores modificados

```
[23]: for i in db.cplayers.find({"name":"Cristiano Ronaldo"},{}):
          print(i)
```

{'_id': ObjectId('65419ce55af52a865c32fa97'), 'player_id': 8198, 'first_name': nan, 'last_name': 'Cristiano Ronaldo', 'name': 'Cristiano Ronaldo', 'last_season': 2022, 'current_club_id': 985, 'player_code': 'cristiano-ronaldo', 'country_of_birth': 'Portugal', 'city_of_birth': 'Funchal', 'country_of_citizenship': 'Portugal', 'date_of_birth': '1985-02-05', 'sub_position': 'Centre-Forward', 'position': 'Attack', 'foot': 'right', 'height_in_cm': 187.0, 'market_value_in_eur': 15000000.0, 'highest_market_value_in_eur': 120000000.0, 'contract_expiration_date': '2025-06-30 00:00:00', 'agent_name': 'Gestifute', 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/8198-1685035469.png?lm=1', 'url': 'https://www.transfermarkt.co.uk/cristiano-ronaldo/profil/spieler/8198', 'current_club_domestic_competition_id': 'GB1', 'current_club_name': 'Manchester United', 'age': 38}

Definimos cuáles serán los nuevos valores a insertar para realizar la actualización adecuada.

```
[24]: new_values = {
          "first_name": "Cristiano",
          "last_name": "dos Santos Aveiro",
          "name": "Cristiano Ronaldo dos Santos Aveiro"
      }
```

Aplicamos el proceso de actualización de campos y obtendremos la cantidad de cambios que se han realizado para cada campo

```
[25]: for field, new_value in new_values.items():
          filter_query = {"name":"Cristiano Ronaldo"}
          update_query = {'$set': {field: new_value}}
          result = db.cplayers.update_many(filter_query, update_query)
          print(f" {result.modified_count} documento(s) actualizado(s) para {field}.")
```

```
1 documento(s) actualizado(s) para first_name.
1 documento(s) actualizado(s) para last_name.
1 documento(s) actualizado(s) para name.
```

Visualizamos el documento modificado de la colección y evidenciaremos la actualización de los valores.

```
[29]: for i in db.cplayers.find({"name":"Cristiano Ronaldo dos Santos Aveiro"},{}):
          print(i)
```

{'_id': ObjectId('6542a6a0aa0ab5da1548501a'), 'player_id': 8198, 'first_name': 'Cristiano', 'last_name': 'dos Santos Aveiro', 'name': 'Cristiano Ronaldo dos Santos Aveiro', 'last_season': 2022, 'current_club_id': 985, 'player_code': 'cristiano-ronaldo', 'country_of_birth': 'Portugal', 'city_of_birth': 'Funchal', 'country_of_citizenship': 'Portugal', 'date_of_birth': '1985-02-05', 'sub_position': 'Centre-Forward', 'position': 'Attack', 'foot': 'right', 'height_in_cm': 187.0, 'market_value_in_eur': 15000000.0, 'highest_market_value_in_eur': 120000000.0, 'contract_expiration_date': '2025-06-30 00:00:00', 'agent_name': 'Gestifute', 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/8198-1685035469.png?lm=1', 'url': 'https://www.transfermarkt.co.uk/cristiano-ronaldo/profil/spieler/8198', 'current_club_domestic_competition_id': 'GB1', 'current_club_name': 'Manchester United', 'age': 38, 'ageRange': '6. Mayor a 35'}

Aplicaremos los mismos cambios para otro jugador reconocido llamado Neymar para poder estandarizar la estructura de los datos.

```
[30]: for i in db.cplayers.find({"name":"Neymar"},{}):
          print(i)
```

{'_id': ObjectId('6542a6a1aa0ab5da15488b9f'), 'player_id': 68290, 'first_name': nan, 'last_name': 'Neymar', 'name': 'Neymar', 'last_season': 2022, 'current_club_id': 583, 'player_code': 'neymar', 'country_of_birth': 'Brazil', 'city_of_birth': 'Mogi das Cruzes', 'country_of_citizenship': 'Brazil', 'date_of_birth': '1992-02-05', 'sub_position': 'Left Winger', 'position': 'Attack', 'foot': 'right', 'height_in_cm': 175.0, 'market_value_in_eur': 60000000.0, 'highest_market_value_in_eur': 180000000.0, 'contract_expiration_date': '2025-06-30 00:00:00', 'agent_name': nan, 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/68290-1669394812.jpg?lm=1', 'url': 'https://www.transfermarkt.co.uk/neymar/profil/spieler/68290', 'current_club_domestic_competition_id': 'FR1', 'current_club_name': 'Paris Saint-Germain', 'age': 31, 'ageRange': '5. De 30 a 35'}

```
[28]: new_values = {
          "first_name": "Neymar",
          "last_name": "da Silva Santos",
          "name": "Neymar da Silva Santos"
      }
```

```
[32]: for field, new_value in new_values.items():
          filter_query = {"name":"Neymar"}
          update_query = {'$set': {field: new_value}}
          result = db.cplayers.update_many(filter_query, update_query)
          print(f" {result.modified_count} documento(s) actualizado(s) para {field}.")
```

```
1 documento(s) actualizado(s) para first_name.
1 documento(s) actualizado(s) para last_name.
1 documento(s) actualizado(s) para name.
```

```
[33]: for i in db.cplayers.find({"name":"Neymar da Silva Santos"},{}):
          print(i)
```

{'_id': ObjectId('6542a6a1aa0ab5da15488b9f'), 'player_id': 68290, 'first_name': 'Neymar', 'last_name': 'da Silva Santos', 'name': 'Neymar da Silva Santos', 'last_season': 2022, 'current_club_id': 583, 'player_code': 'neymar', 'country_of_birth': 'Brazil', 'city_of_birth': 'Mogi das Cruzes', 'country_of_citizenship': 'Brazil', 'date_of_birth': '1992-02-05', 'sub_position': 'Left Winger', 'position': 'Attack', 'foot': 'right', 'height_in_cm': 175.0, 'market_value_in_eur': 60000000.0, 'highest_market_value_in_eur': 180000000.0, 'contract_expiration_date': '2025-06-30 00:00:00', 'agent_name': nan, 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/68290-1669394812.jpg?lm=1', 'url': 'https://www.transfermarkt.co.uk/neymar/profil/spieler/68290', 'current_club_domestic_competition_id': 'FR1', 'current_club_name': 'Paris Saint-Germain', 'age': 31, 'ageRange': '5. De 30 a 35'}

c. Ejercicios sobre proyección

Con el siguiente ejercicio buscamos hacer uso de las proyecciones para poder mostrar o ocultar los campos que sean de nuestra conveniencia o que nos permitan visualizar mejor los datos de los análisis generados.

Por ejemplo, a continuación, se presentan los 15 jugadores más altos que se encuentran en la colección de “players” proyectando los campos “name”, “position”, “height_in_cm” y “current_club_name”, se evidencia que se oculta el valor del campo “_id”.

Como se evidencia obtenemos que la gran mayoría de jugadores con mayor talla se encuentran en la posición de arqueros o guardametas.

```
[36]: for i in db.cplayers.find({ "height_in_cm": { "$gt": 175} },{"name":1,"height_in_cm":1,"current_club_name":1,"position":1,"_id":0}).sort("height_in_cm",-
          print(i)
```

```
{'name': 'Kristof Van Hout', 'position': 'Goalkeeper', 'height_in_cm': 207.0, 'current_club_name': 'KVC Westerlo'}
{'name': 'Kjell Scherpen', 'position': 'Goalkeeper', 'height_in_cm': 206.0, 'current_club_name': 'Vitesse Arnhem'}
{'name': 'Denys Tvardovskyi', 'position': 'Goalkeeper', 'height_in_cm': 206.0, 'current_club_name': 'Shakhtar Donetsk'}
{'name': 'Kevin Gadellaa', 'position': 'Goalkeeper', 'height_in_cm': 206.0, 'current_club_name': 'FC Utrecht'}
{'name': 'Isaak Touré', 'position': 'Defender', 'height_in_cm': 206.0, 'current_club_name': 'FC Lorient'}
{'name': 'Norbert Haymamba', 'position': 'Goalkeeper', 'height_in_cm': 205.0, 'current_club_name': 'FC Arouca'}
{'name': 'Vanja Ivesa', 'position': 'Goalkeeper', 'height_in_cm': 205.0, 'current_club_name': 'Elazigspor'}
{'name': 'Tom Hülsmann', 'position': 'Goalkeeper', 'height_in_cm': 205.0, 'current_club_name': 'Bayern Munich'}
{'name': 'Lucas Bergström', 'position': 'Goalkeeper', 'height_in_cm': 205.0, 'current_club_name': 'Chelsea FC'}
{'name': 'Melih Akyüz', 'position': 'Goalkeeper', 'height_in_cm': 204.0, 'current_club_name': 'Kasimpasa'}
{'name': 'Idé Colubali', 'position': 'Attack', 'height_in_cm': 204.0, 'current_club_name': 'Boavista FC'}
{'name': 'Betim Halimi', 'position': 'Goalkeeper', 'height_in_cm': 204.0, 'current_club_name': 'Olimpik Donetsk'}
{'name': 'Dmitri Stajila', 'position': 'Goalkeeper', 'height_in_cm': 204.0, 'current_club_name': 'Kuban Krasnodar (-2018)'}
{'name': 'Jacob Samnik', 'position': 'Goalkeeper', 'height_in_cm': 204.0, 'current_club_name': 'Hobro IK'}
{'name': 'Marijan Coric', 'position': 'Goalkeeper', 'height_in_cm': 204.0, 'current_club_name': 'Parma Calcio 1913'}
```

Por otro lado, buscamos proyectar a los 15 jugadores más jóvenes pertenecientes a las 5 grandes ligas del futbol europeo para analizar su crecimiento futbolistico y para presentar mejor los datos lo mostraremos en formato de tabla. Tomar en cuenta que se proyectarán los campos “name”, “height_in_cm”, “current_club_name”, “position” y “edad”, asimismo se continúa ocultando el campo “_id”

```
•[33]: tabla1 = PrettyTable()
tabla1.field_names = ["name", "height_in_cm","current_club_name","position","age"]
for documento in db.cplayers.find({"current_club_domestic_competition_id":
                                   {"$in":["L1","ES1","GB1","IT1","FR1"]},
                                   "$and":[{"age":{'$ne': None}}]},
                                  {"name":1,"height_in_cm":1,"current_club_name":1,"position":1,"age":1,"_id":0}).sort("age",1).limit(15):
    campo1 = documento["name"]
    campo2 = documento["height_in_cm"]
    campo3 = documento["current_club_name"]
    campo4 = documento["position"]
    campo5 = documento["age"]
    tabla1.add_row([campo1, campo2, campo3,campo4, campo5])
print(tabla1)
```

name	height_in_cm	current_club_name	position	age
Ethan Mbappé	176.0	Paris Saint-Germain	Midfield	16
Ethan Nwaneri	165.0	Arsenal FC	Midfield	16
David Pejičić	nan	Udinese Calcio	Midfield	16
Amidou Doumbouya	182.0	OGC Nice	Defender	16
Tommaso Vannucchi	197.0	ACF Fiorentina	Goalkeeper	16
Lamine Yamal	180.0	FC Barcelona	Attack	16
Mamadu Silla	nan	AC Ajaccio	Missing	16
David Odogu	190.0	VfL Wolfsburg	Defender	17
Bastien Meupiyou	191.0	FC Nantes	Defender	17
Archie Gray	nan	Leeds United	Midfield	17
Alphadjo Cissè	nan	Hellas Verona	Attack	17
Oliver Scarles	nan	West Ham United	Midfield	17
Marc Guiu	187.0	FC Barcelona	Attack	17
Tommaso Martinelli	194.0	ACF Fiorentina	Goalkeeper	17
Timo Schlieck	195.0	RB Leipzig	Goalkeeper	17

Continuamos mostrando ejercicios de proyección, en el siguiente ejemplo se busca ver cuales son los 5 mas grandes estadios de futbol de las 5 grandes ligas que pueden albergar la mayor cantidad de asistentes para un partido de futbol, los cuales también serán mostrado en formato de tabla. Tomar en [illegible]e" y

[37]:
```
tabla2 = PrettyTable()
tabla2.field_names = ["name", "stadium_name","stadium_seats"]
for documento in db.cclubs.find({"domestic_competition_id":{"$in":["L1","ES1","GB1","IT1","FR1"]},
                                 "$and":[{"last_season": 2023}]},
                                {"name":1,"stadium_name":1,"stadium_seats":1,"_id":0}).sort("stadium_seats",-1).limit(5):
    campo1 = documento["name"]
    campo2 = documento["stadium_name"]
    campo3 = documento["stadium_seats"]
    tabla2.add_row([campo1, campo2, campo3])
print(tabla2)
```

```
+--------------------+--------------------+----------------+
|        name        |    stadium_name    | stadium_seats  |
+--------------------+--------------------+----------------+
| Borussia Dortmund | SIGNAL IDUNA PARK |     81365      |
|    Real Madrid     | Santiago Bernabéu |     81044      |
|      AC Milan      |  Giuseppe Meazza  |     75923      |
|    Inter Milan     |  Giuseppe Meazza  |     75923      |
|   Bayern Munich    |   Allianz Arena   |     75024      |
+--------------------+--------------------+----------------+
```

d. Ejercicios sobre filtrado

Finalmente, para los ejercicios de filtrado evidenciaremos que se puede filtrar toda una colección por valores específicos, para nuestro ejemplo mostraremos los valores de los 2 más grandes jugadores activos que han marcado una época en el futbol "Cristiano Ronaldo" y "Lionel Messi", como detalle extra capturaremos la URL perteneciente a la imagen de cada uno para poder mostrar su imagen y sus datos.

[38]:
```
for i in db.cplayers.find({"name":"Cristiano Ronaldo dos Santos Aveiro"},{}):
    print(i)
    cr7_url=i["image_url"]
```

{'_id': ObjectId('6542a6a0aa0ab5da1548501a'), 'player_id': 8198, 'first_name': 'Cristiano', 'last_name': 'dos Santos Aveiro', 'name': 'Cristiano Ronaldo dos Santos Aveiro', 'last_season': 2022, 'current_club_id': 985, 'player_code': 'cristiano-ronaldo', 'country_of_birth': 'Portugal', 'city_of_birth': 'Funchal', 'country_of_citizenship': 'Portugal', 'date_of_birth': '1985-02-05', 'sub_position': 'Centre-Forward', 'position': 'Attack', 'foot': 'right', 'height_in_cm': 187.0, 'market_value_in_eur': 15000000.0, 'highest_market_value_in_eur': 120000000.0, 'contract_expiration_date': '2025-06-30 00:00:00', 'agent_name': 'Gestifute', 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/8198-1685035469.png?lm=1', 'url': 'https://www.transfermarkt.co.uk/cristiano-ronaldo/profil/spieler/8198', 'current_club_domestic_competition_id': 'GB1', 'current_club_name': 'Manchester United', 'age': 38, 'ageRange': '6. Mayor a 35'}

[41]:
```
for i in db.cplayers.find({"last_name":"Messi"},{}):
    print(i)
    lm10_url=i["image_url"]
```

{'_id': ObjectId('6542a6a0aa0ab5da1548506f'), 'player_id': 28003, 'first_name': 'Lionel', 'last_name': 'Messi', 'name': 'Lionel Messi', 'last_season': 2022, 'current_club_id': 583, 'player_code': 'lionel-messi', 'country_of_birth': 'Argentina', 'city_of_birth': 'Rosario', 'country_of_citizenship': 'Argentina', 'date_of_birth': '1987-06-24', 'sub_position': 'Right Winger', 'position': 'Attack', 'foot': 'left', 'height_in_cm': 170.0, 'market_value_in_eur': 35000000.0, 'highest_market_value_in_eur': 180000000.0, 'contract_expiration_date': nan, 'agent_name': nan, 'image_url': 'https://img.a.transfermarkt.technology/portrait/header/28003-1671435885.jpg?lm=1', 'url': 'https://www.transfermarkt.co.uk/lionel-messi/profil/spieler/28003', 'current_club_domestic_competition_id': 'FR1', 'current_club_name': 'Paris Saint-Germain', 'age': 36, 'ageRange': '6. Mayor a 35'}

El campo "image_url" almacenado previamente nos será de utilidad para mostrar la imagen de los jugadores usando la librería PIL y el método Image cargados al inicio.

[39]:
```
cr7_image = Image.open(requests.get(cr7_url, stream=True).raw)
```

[40]:
```
display(cr7_image)
#cr7_image.show() #sirve para desplegar la imagen en un visor de imagenes de la PC
```

```
[40]: lm10_image = Image.open(requests.get(lm10_url, stream=True).raw)

[41]: display(lm10_image)
      #lm10_image.show() #sirve para desplegar la imagen en un visor de imagenes de la PC
```

3. Ejercicios sobre pipeline de agregación

a. Máximo valor de mercado del último año que fueron valorados los jugadores que tuvieron actividad en Europa en las temporadas 2022 y 2023 en las 5 grandes ligas

Para el siguiente ejercicio sobre pipeline de agregación será necesario generar primero un nuevo campo para la colección “playersvaluation” el cual corresponde a la obtención del año en la cual un jugador tuvo una valoración tomando en cuenta que un jugador puede tener múltiples valoraciones dependiente de su rendimiento en cada periodo de tiempo.

```
[42]: for document in db.cplayervaluation.find({}):
          timestamp = document.get("datetime")
          if timestamp:
              year = timestamp[:4]
              db.cplayervaluation.update_many({'_id': document['_id']}, {'$set': {'year_valuation': int(year)}})
```

Visualizamos que se generó el campo “year_valuation” que corresponde al año en el cual un jugador tuvo una valoración.

```
[45]: for x in db.cplayervaluation.find({}).limit(1):
          print(x)

{'_id': ObjectId('6542a6a4aa0ab5da1549b493'), 'player_id': 3132, 'last_season': 2013, 'datetime': '2003-12-09 00:00:00', 'date': '2003-12-09', 'datewee
k': '2003-12-08', 'market_value_in_eur': 400000, 'n': 1, 'current_club_id': 126, 'player_club_domestic_competition_id': 'TR1', 'year_valuation': 2003}
```

Debido a que se quiere tener datos que nos permitan tener mayor detalle descriptivo de los datos, guardaremos los resultados de algunas proyecciones o agregaciones en una nueva colección.

```
[45]: cplayers_filtered = db['PlayersFiltered']
```

Como se muestra a continuación se insertan los datos filtrados y proyectados de la colección “players” a una nueva colección llamada “players_filtered” la cual contiene valores de los players pertenecientes a las 5 grandes ligas europeas que jugaron la temporada 2022 y 2023.

```
[46]: inserted_ids = db.cplayers_filtered.insert_many(db.cplayers.find({"current_club_domestic_competition_id":{"$in":["L1","ES1","GB1","IT1","FR1"]}
                                                                        ,"$and":[{"last_season": {"$in":[2023,2022]}}]},
                                                                       {"_id":0,"player_id":1,"name":1,"position":1,"current_club_domestic_competition_id":1,
                                                                        "current_club_name":1,"age":1,"ageRange":1}))
      print("Inserted document IDs:", inserted_ids.inserted_ids)

      Inserted document IDs: [ObjectId('6542b801aa0ab5da1574579b'), ObjectId('6542b801aa0ab5da1574579c'), ObjectId('6542b801aa0ab5da1574579d'), ObjectId('654
      2b801aa0ab5da1574579e'), ObjectId('6542b801aa0ab5da1574579f'), ObjectId('6542b801aa0ab5da157457a0'), ObjectId('6542b801aa0ab5da157457a1'), ObjectId('65
      42b801aa0ab5da157457a2'), ObjectId('6542b801aa0ab5da157457a3'), ObjectId('6542b801aa0ab5da157457a4'), ObjectId('6542b801aa0ab5da157457a5'), ObjectId('6
      542b801aa0ab5da157457a6'), ObjectId('6542b801aa0ab5da157457a7'), ObjectId('6542b801aa0ab5da157457a8'), ObjectId('6542b801aa0ab5da157457a9'), ObjectId
      ('6542b801aa0ab5da157457aa'), ObjectId('6542b801aa0ab5da157457ab'), ObjectId('6542b801aa0ab5da157457ac'), ObjectId('6542b801aa0ab5da157457ad'), ObjectI
      d('6542b801aa0ab5da157457ae'), ObjectId('6542b801aa0ab5da157457af'), ObjectId('6542b801aa0ab5da157457b0'), ObjectId('6542b801aa0ab5da157457b1'), Object
      Id('6542b801aa0ab5da157457b2'), ObjectId('6542b801aa0ab5da157457b3'), ObjectId('6542b801aa0ab5da157457b4'), ObjectId('6542b801aa0ab5da157457b5'), Objec
      tId('6542b801aa0ab5da157457b6'), ObjectId('6542b801aa0ab5da157457b7'), ObjectId('6542b801aa0ab5da157457b8'), ObjectId('6542b801aa0ab5da157457b9'), Obje
      ctId('6542b801aa0ab5da157457ba'), ObjectId('6542b801aa0ab5da157457bb'), ObjectId('6542b801aa0ab5da157457bc'), ObjectId('6542b801aa0ab5da157457bd'), Obj
      ectId('6542b801aa0ab5da157457be'), ObjectId('6542b801aa0ab5da157457bf'), ObjectId('6542b801aa0ab5da157457c0'), ObjectId('6542b801aa0ab5da157457c1'), Ob
      jectId('6542b801aa0ab5da157457c2'), ObjectId('6542b801aa0ab5da157457c3'), ObjectId('6542b801aa0ab5da157457c4'), ObjectId('6542b801aa0ab5da157457c5'), O
      bjectId('6542b801aa0ab5da157457c6'), ObjectId('6542b801aa0ab5da157457c7'), ObjectId('6542b801aa0ab5da157457c8'), ObjectId('6542b801aa0ab5da157457c9'),
```

Mostramos los datos que posee la nueva colección “players_filtered” recorriendo el cursor y limitamos el resultado a 2 documentos.

```
[48]: for x in db.cplayers_filtered.find({}).limit(2):
          print(x)
```

```
{'_id': ObjectId('6542b801aa0ab5da1574579b'), 'player_id': 18922, 'name': 'Karim Benzema', 'position': 'Attack', 'current_club_domestic_competition_id':
'ES1', 'current_club_name': 'Real Madrid', 'age': 35, 'ageRange': '5. De 30 a 35'}
{'_id': ObjectId('6542b801aa0ab5da1574579c'), 'player_id': 30321, 'name': 'Óscar Trejo', 'position': 'Midfield', 'current_club_domestic_competition_id':
'ES1', 'current_club_name': 'Rayo Vallecano', 'age': 35, 'ageRange': '5. De 30 a 35'}
```

Una vez se posee la colección “players_filtered” procederemos a generar la relación con la valoración que poseen dichos jugadores para ello aplicaremos una agregación que corresponde al siguiente calculo:

- Se filtrarán los jugadores que pertenecen a las 5 grandes ligas europeas que jugaron la temporada 2022 y 2023.
- Se agruparán mediante los campos “player_id”, “current_club_id”, “player_club_domestic_competition_id” y el campo calculado “year_valuation” generando la agregación del máximo valor de del campo “market_value_in_eur”, con esto obtenemos los valores de mercado máximos que tiene cada jugador cada año que fue valorado.
- Haremos uso del método mergeObjects para dividir los campos que posee el “_id” de la agrupación y acceder a los campos de la agrupación directamente, así como a la agregación correspondiente.
- Luego haremos un ranking de las valoraciones de mercado que han tenido los jugadores, particionados por el “player_id” y ordenadas en forma descendente por “year” el cual es el año de valoración del jugador, de esta manera obtendremos un ranking.
- Luego nos quedaremos con el valor 1 del ranking para cada jugador para garantizar que obtengamos el máximo valor de mercado de un jugador en el ultimo año que tuvo una valoración.
- Finalmente eliminaremos el campo Ranking para que no interfiera con los valores que deseamos presentar.

```
[60]: pipeline = [
    {
        '$match': {
            "player_club_domestic_competition_id":{"$in":["L1","ES1","GB1","IT1","FR1"]},
            "$and":[{"last_season": {"$in":[2023,2022]}}]
        }
    },
    {
        '$group': {
            '_id': {
                "player_id" : "$player_id", "current_club_id" : "$current_club_id",
                "player_club_domestic_competition_id" : "$player_club_domestic_competition_id","year" : "$year_valuation"},
            'max_market_value_per_year': {'$max': f'${"market_value_in_eur"}'}
        }
    },
    { "$replaceWith": { "$mergeObjects": [ "$_id",{ "max_market_value_per_year": "$max_market_value_per_year"} ] } },
    {
      "$setWindowFields": {
        "partitionBy": "$player_id",
        "sortBy": { "year": -1 },
        "output": {
          "Ranking": {
            "$denseRank": {}
          }
        }
      }
    },
    {
        "$match":{
            "Ranking":1
        }
    },
    {
        "$project": {
            "Ranking":0
        }
    }
]
```

Nota: Para evidenciar este análisis colocamos el siguiente ejemplo, en el cual podemos ver que para “Cristiano Ronaldo” tenemos múltiples valoraciones en los distintos años por lo cual queremos quedarnos con el último máximo valor de mercado que haya tenido en el ultimo año que fue valorado.

```
[59]: for i in db.cplayervaluation.aggregate(pipeline):
          if i.get("player_id")==8198:
              print(i)
```

```
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2023, 'max_market_value_per_year': 15000000, 'Ranking': 1}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2022, 'max_market_value_per_year': 30000000, 'Ranking': 2}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2021, 'max_market_value_per_year': 50000000, 'Ranking': 3}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2020, 'max_market_value_per_year': 60000000, 'Ranking': 4}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2019, 'max_market_value_per_year': 90000000, 'Ranking': 5}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2018, 'max_market_value_per_year': 120000000, 'Ranking': 6}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2017, 'max_market_value_per_year': 100000000, 'Ranking': 7}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2016, 'max_market_value_per_year': 110000000, 'Ranking': 8}
{'player_id': 8198, 'current_club_id': 985, 'player_club_domestic_competition_id': 'GB1', 'year': 2015, 'max_market_value_per_year': 120000000, 'Ranking': 9}
```

Mostramos el resultado obtenido de esta agregación.

```
[61]: for i in db.cplayervaluation.aggregate(pipeline):
          print(i)
```

```
{'player_id': 2857, 'current_club_id': 29, 'player_club_domestic_competition_id': 'GB1', 'year': 2022, 'max_market_value_per_year': 250000}
{'player_id': 3333, 'current_club_id': 1237, 'player_club_domestic_competition_id': 'GB1', 'year': 2023, 'max_market_value_per_year': 1500000}
{'player_id': 3455, 'current_club_id': 5, 'player_club_domestic_competition_id': 'IT1', 'year': 2023, 'max_market_value_per_year': 2000000}
{'player_id': 5578, 'current_club_id': 1421, 'player_club_domestic_competition_id': 'FR1', 'year': 2023, 'max_market_value_per_year': 100000}
{'player_id': 7161, 'current_club_id': 15, 'player_club_domestic_competition_id': 'L1', 'year': 2023, 'max_market_value_per_year': 13000000}
{'player_id': 7663, 'current_club_id': 150, 'player_club_domestic_competition_id': 'ES1', 'year': 2023, 'max_market_value_per_year': 1500000}
```

Esta información será almacenada en una nueva colección llamada “playervaluation_filtered” para facilitar el próximo paso del análisis.

```
[48]: cplayervaluation_filtered = db['PlayersValuation_Filtered']
```

```
[51]: inserted_ids = db.cplayervaluation_filtered.insert_many(db.cplayervaluation.aggregate(pipeline))
      print("Inserted document IDs:", inserted_ids.inserted_ids)
```

```
Inserted document IDs: [ObjectId('65419e725af52a865c5f119e'), ObjectId('65419e725af52a865c5f119f'), ObjectId('65419e725af52a865c5f11a0'), ObjectId('65419e725af52a865c5f11a1'), ObjectId('65419e725af52a865c5f11a2'), ObjectId('65419e725af52a865c5f11a3'), ObjectId('65419e725af52a865c5f11a4'), ObjectId('65419e725af52a865c5f11a5'), ObjectId('65419e725af52a865c5f11a6'), ObjectId('65419e725af52a865c5f11a7'), ObjectId('65419e725af52a865c5f11a8'), ObjectId('65419e725af52a865c5f11a9'), ObjectId('65419e725af52a865c5f11aa'), ObjectId('65419e725af52a865c5f11ab'), ObjectId('65419e725af52a865c5f11ac'), ObjectId('65419e725af52a865c5f11ad'), ObjectId('65419e725af52a865c5f11ae'), ObjectId('65419e725af52a865c5f11af'), ObjectId('65419e725af52a865c5f11b0'), ObjectId('65419e725af52a865c5f11b1'), ObjectId('65419e725af52a865c5f11b2'), ObjectId('65419e725af52a865c5f11b3'), ObjectId('65419e725af52a865c5f11b4'), ObjectId('65419e725af52a865c5f11b5'), ObjectId('65419e725af52a865c5f11b6'), ObjectId('65419e725af52a865c5f11b7'), ObjectId('65419e725af52a865c5f11b8'), ObjectId('65419e725af52a865c5f11b9'), ObjectId('65419e725af52a865c5f11ba'), ObjectId('65419e725af52a865c5f11bb'), ObjectId('65419e725af52a865c5f11bc'), ObjectId('65419e725af52a865c5f11bd'), ObjectId('65419e725af52a865c5f11be'), ObjectId('65419e725af52a865c5f11bf'), ObjectId('65419e725af52a865c5
```

Evidenciamos los valores insertados en esta nueva colección

```
[65]: for i in db.cplayervaluation_filtered.find({}):
          print(i)
```

```
{'_id': ObjectId('6542bf00aa0ab5da15746721'), 'player_id': 2857, 'current_club_id': 29, 'player_club_domestic_competition_id': 'GB1', 'year': 2022, 'max_market_value_per_year': 250000}
{'_id': ObjectId('6542bf00aa0ab5da15746722'), 'player_id': 3333, 'current_club_id': 1237, 'player_club_domestic_competition_id': 'GB1', 'year': 2023, 'max_market_value_per_year': 1500000}
{'_id': ObjectId('6542bf00aa0ab5da15746723'), 'player_id': 3455, 'current_club_id': 5, 'player_club_domestic_competition_id': 'IT1', 'year': 2023, 'max_market_value_per_year': 2000000}
{'_id': ObjectId('6542bf00aa0ab5da15746724'), 'player_id': 5578, 'current_club_id': 1421, 'player_club_domestic_competition_id': 'FR1', 'year': 2023, 'max_market_value_per_year': 100000}
```

Finalmente, en generaremos una nueva agregación en la cual podamos hacer un join (lookup) de las 2 colecciones generadas anteriormente la cual contendrá las siguientes etapas:

- Lookup de las dos colecciones realizadas por el campo “player_id” y denominado como “valuations”
- Se realizará una ordenación por el campo “max_market_value_per_year” de la colección “playervaluation_filtered”
- Se limitará el resultado para obtener solo 10 registros
- Se hará una proyección de los campos “name”,”current_club_name”, “position”, “age”, y se obtendrá el

primer y único valor de los campos “max_market_valur_per_year” y “year”

[66]:
```
pipeline = [
    {
        '$lookup': {
            'from': 'cplayervaluation_filtered',
            'localField': 'player_id',
            'foreignField': 'player_id',
            'as': 'valuations'
        }
    },
    {
        "$sort": {
            "valuations.max_market_value_per_year":-1
        }
    },
    {
        "$limit": 10
    },
    {
        '$project': {
            '_id': 0,
            'name': 1,
            'market_value_value_last_year': {'$arrayElemAt': ['$valuations.max_market_value_per_year', 0]},
            'last_year_valuation': {'$arrayElemAt': ['$valuations.year', 0]},
            'current_club_name': 1,
            'position': 1,
            'age': 1
        }
    }
]
```

De esta manera obtenemos los 10 jugadores con máximo valor de mercado que compitieron entre los años 2022 y 2023 en las 5 grandes ligas de Europa.

[67]:
```
result = db.cplayers_filtered.aggregate(pipeline)
for i in result:
    print(i)
```

```
{'name': 'Erling Haaland', 'position': 'Attack', 'current_club_name': 'Manchester City', 'age': 23, 'market_value_value_last_year': 180000000, 'last_year
_valuation': 2023}
{'name': 'Kylian Mbappé', 'position': 'Attack', 'current_club_name': 'Paris Saint-Germain', 'age': 24, 'market_value_value_last_year': 180000000, 'last_y
ear_valuation': 2023}
{'name': 'Vinicius Junior', 'position': 'Attack', 'current_club_name': 'Real Madrid', 'age': 23, 'market_value_value_last_year': 150000000, 'last_year_va
luation': 2023}
{'name': 'Victor Osimhen', 'position': 'Attack', 'current_club_name': 'SSC Napoli', 'age': 24, 'market_value_value_last_year': 120000000, 'last_year_valu
ation': 2023}
{'name': 'Jude Bellingham', 'position': 'Midfield', 'current_club_name': 'Real Madrid', 'age': 20, 'market_value_value_last_year': 120000000, 'last_year_
valuation': 2023}
{'name': 'Bukayo Saka', 'position': 'Attack', 'current_club_name': 'Arsenal FC', 'age': 22, 'market_value_value_last_year': 120000000, 'last_year_valuati
on': 2023}
{'name': 'Phil Foden', 'position': 'Attack', 'current_club_name': 'Manchester City', 'age': 23, 'market_value_value_last_year': 110000000, 'last_year_val
uation': 2023}
{'name': 'Jamal Musiala', 'position': 'Midfield', 'current_club_name': 'Bayern Munich', 'age': 20, 'market_value_value_last_year': 110000000, 'last_year_
valuation': 2023}
{'name': 'Pedri', 'position': 'Midfield', 'current_club_name': 'FC Barcelona', 'age': 20, 'market_value_value_last_year': 100000000, 'last_year_valuatio
n': 2023}
{'name': 'Federico Valverde', 'position': 'Midfield', 'current_club_name': 'Real Madrid', 'age': 25, 'market_value_value_last_year': 100000000, 'last_yea
r_valuation': 2023}
```

Tomando en cuenta el resultado obtenido podemos evidenciar que en la vida real muchos de los jugadores mostrados aun poseen altos valores de mercado por su consolidación en el futbol europeo.

b. Impacto que tienen los jugadores, competiciones, rangos de edad con la cantidad de apariciones, tomando en cuenta las 5 grandes ligas de Europa.

Para este ejercicio sobre pipeline de agregación generaremos una nueva colección que posea el agrupado de las apariciones que han tenido los jugadores para ello detallaremos los pasos realizados en esta agregación:

- Se filtrarán los jugadores que pertenecen a las 5 grandes ligas europeas que hayan tenido apariciones desde el inicio del año 2023.
- Se realizará la agrupación por los campos “player_name”, “player_id”, “player_current_club_id”, y el agrupado sumarizado de

los campos “yellow_cards”, “red_cards”, “goals”, “assists” y “minutes_played”

- Haremos uso del método mergeObjects para dividir los campos que posee el “_id” de la agrupación y acceder a los campos de la agrupación directamente, así como a la agregación correspondiente.

```
[76]: pipeline = [
    {
        '$match': {
            "competition_id":{"$in":["L1","ES1","GB1","IT1","FR1"]},
            "date":{"$gte":"2023-01-01"}
        }
    },
    {
        '$group': {
            '_id': {
                "player_name" : "$player_name",
                "player_id" : "$player_id",
                "player_current_club_id" : "$player_current_club_id"
            },
            'yellow_cards': {'$sum': f'${"yellow_cards"}'},
            'red_cards': {'$sum': f'${"red_cards"}'},
            'goals': {'$sum': f'${"goals"}'},
            'assists': {'$sum': f'${"assists"}'},
            'minutes_played': {'$sum': f'${"minutes_played"}'}
        }
    },
    { "$replaceWith":
     { "$mergeObjects":
      [ "$_id",
       { "yellow_cards": "$yellow_cards","red_cards": "$red_cards","goals": "$goals","assists": "$assists","minutes_played": "$minutes_played"}
      ]
     }
    }
]
```

```
[77]: cappearances_filtered = db['AppearancesFiltered']
```

Evidenciamos el resultado obtenido de la agregación mostrada previamente en la cual obtenemos el sumarizado de las apariciones que han tenido los jugadores.

```
[78]: for document in db.cappearances.aggregate(pipeline):
          print(document)
```

```
{'player_name': 'Rayan Aït-Nouri', 'player_id': 578391, 'player_current_club_id': 543, 'yellow_cards': 4, 'red_cards': 0, 'goals': 0, 'assists': 0, 'minutes_played': 874}
{'player_name': 'Charles De Ketelaere', 'player_id': 435772, 'player_current_club_id': 800, 'yellow_cards': 0, 'red_cards': 0, 'goals': 1, 'assists': 2, 'minutes_played': 1019}
{'player_name': 'Alessandro Zanoli', 'player_id': 397225, 'player_current_club_id': 6195, 'yellow_cards': 3, 'red_cards': 0, 'goals': 2, 'assists': 2, 'minutes_played': 1539}
{'player_name': 'Djed Spence', 'player_id': 483348, 'player_current_club_id': 148, 'yellow_cards': 0, 'red_cards': 0, 'goals': 0, 'assists': 0, 'minutes_played': 576}
{'player_name': 'Pedro Bigas', 'player_id': 203043, 'player_current_club_id': 1531, 'yellow_cards': 1, 'red_cards': 2, 'goals': 0, 'assists': 0, 'minutes_played': 965}
{'player_name': 'Juan Foyth', 'player_id': 480763, 'player_current_club_id': 1050, 'yellow_cards': 6, 'red_cards': 0, 'goals': 1, 'assists': 0, 'minutes_played': 2447}
{'player_name': 'Pau Torres', 'player_id': 399776, 'player_current_club_id': 405, 'yellow_cards': 5, 'red_cards': 0, 'goals': 2, 'assists': 0, 'minutes_played': 2484}
{'player_name': 'Dwight McNeil', 'player_id': 584769, 'player_current_club_id': 29, 'yellow_cards': 3, 'red_cards': 0, 'goals': 5, 'assists': 3, 'minutes_played': 2078}
```

Todos los documentos que se han obtenido se almacenan en la nueva colección generada que corresponde a las apariciones filtradas y agrupadas.

```
[74]: inserted_ids = db.cappearances_filtered.insert_many(db.cappearances.aggregate(pipeline))
      print("Inserted document IDs:", inserted_ids.inserted_ids)
```

```
Inserted document IDs: [ObjectId('6542c3fdaa0ab5da1574760d'), ObjectId('6542c3fdaa0ab5da1574760e'), ObjectId('6542c3fdaa0ab5da1574760f'), ObjectId('6542c3fdaa0ab5da15747610'), ObjectId('6542c3fdaa0ab5da15747611'), ObjectId('6542c3fdaa0ab5da15747612'), ObjectId('6542c3fdaa0ab5da15747613'), ObjectId('6542c3fdaa0ab5da15747614'), ObjectId('6542c3fdaa0ab5da15747615'), ObjectId('6542c3fdaa0ab5da15747616'), ObjectId('6542c3fdaa0ab5da15747617'), ObjectId('6542c3fdaa0ab5da15747618'), ObjectId('6542c3fdaa0ab5da15747619'), ObjectId('6542c3fdaa0ab5da1574761a'), ObjectId('6542c3fdaa0ab5da1574761b'), ObjectId('6542c3fdaa0ab5da1574761c'), ObjectId('6542c3fdaa0ab5da1574761d'), ObjectId('6542c3fdaa0ab5da1574761e'), ObjectId('6542c3fdaa0ab5da1574761f'), ObjectId('6542c3fdaa0ab5da15747620'), ObjectId('6542c3fdaa0ab5da15747621'), ObjectId('6542c3fdaa0ab5da15747622'), ObjectId('6542c3fdaa0ab5da15747623'), ObjectId('6542c3fdaa0ab5da15747624'), ObjectId('6542c3fdaa0ab5da15747625'), ObjectId('6542c3fdaa0ab5da15747626'), ObjectId('6542c3fdaa0ab5da15747627'), ObjectId('6542c3fdaa0ab5da15747628'), ObjectId('6542c3fdaa0ab5da15747629'), ObjectId('6542c3fdaa0ab5da1574762a'), ObjectId('6542c3fdaa0ab5da15747
```

Evidenciamos los datos que posee cada documento de la nueva colección una vez insertados los datos.

```
[79]: for x in db.cappearances_filtered.find({}).limit(2):
          print(x)
```

```
{'_id': ObjectId('6542c3fdaa0ab5da1574760d'), 'player_name': 'Aaron Cresswell', 'player_id': 92571, 'player_current_club_id': 379, 'yellow_cards': 0, 'red_cards': 0, 'goals': 0, 'assists': 1, 'minutes_played': 815}
{'_id': ObjectId('6542c3fdaa0ab5da1574760e'), 'player_name': 'Emile Smith Rowe', 'player_id': 392765, 'player_current_club_id': 11, 'yellow_cards': 0, 'red_cards': 0, 'goals': 0, 'assists': 2, 'minutes_played': 125}
```

Crearemos una nueva colección la cual contendrá el join de la tabla previamente creada y la colección que se generó para el primer análisis “player_filtered”, la cual tendrá los campos “name”, “position”, “ageRange” y “current_club_name” de la colección “player_filtered” y los campos “total_minutes_played”, “total_goals”, “total_assists”, “total_yellow_cards” y “total_red_cards” de la segunda colección “AppearancesFiltered”

```
[65]: cappearancesplayer = db['AppearancesPlayers']
```

```
[66]: pipeline = [
    {
        '$lookup': {
            'from': 'cappearances_filtered',
            'localField': 'player_id',
            'foreignField': 'player_id',
            'as': 'appearances'
        }
    },
    {
        '$project': {
            '_id': 0,
            'name': 1,
            'position': 1,
            'current_club_domestic_competition_id': 1,
            'year_appearences': {'$arrayElemAt': ['$appearances.year_appearences', 0]},
            'total_minutes_played': {'$arrayElemAt': ['$appearances.minutes_played', 0]},
            'total_goals': {'$arrayElemAt': ['$appearances.goals', 0]},
            'total_assists': {'$arrayElemAt': ['$appearances.assists', 0]},
            'total_yellow_cards': {'$arrayElemAt': ['$appearances.yellow_cards', 0]},
            'total_red_cards': {'$arrayElemAt': ['$appearances.red_cards', 0]},
            'ageRange': 1,
            'current_club_name': 1
        }
    }
]
```

Insertaremos el cruce de ambas tablas en la colección previamente creada y evidenciaremos como queda nuestra nueva colección.

```
[85]: for document in db.cplayers_filtered.aggregate(pipeline):
          db.cappearancesplayer.insert_one(document)
```

```
[88]: for i in db.cappearancesplayer.find({}).limit(2):
          print(i)

{'_id': ObjectId('6542c741aa0ab5da157481dd'), 'name': 'Karim Benzema', 'position': 'Attack', 'current_club_domestic_competition_id': 'ES1', 'current_club_name': 'Real Madrid', 'ageRange': '5. De 30 a 35', 'total_minutes_played': 1326, 'total_goals': 12, 'total_assists': 2, 'total_yellow_cards': 1, 'total_red_cards': 0}
{'_id': ObjectId('6542c741aa0ab5da157481de'), 'name': 'Óscar Trejo', 'position': 'Midfield', 'current_club_domestic_competition_id': 'ES1', 'current_club_name': 'Rayo Vallecano', 'ageRange': '5. De 30 a 35', 'total_minutes_played': 1646, 'total_goals': 1, 'total_assists': 2, 'total_yellow_cards': 4, 'total_red_cards': 0}
```

Sobre la nueva colección analizaremos la cantidad de minutos jugados por rango de edad y competición, para lo cual generaremos el pipeline correspondiente.

```
[89]: pipeline = [
    {
        '$group': {
            '_id': {
                "competition_id":"$current_club_domestic_competition_id",
                "ageRange" : "$ageRange"
            },
            'total_minutes_played': {'$sum': f'${"total_minutes_played"}'},
        }
    },
    {
        "$sort": {
            "total_minutes_played":-1,
        }
    },
    { "$replaceWith":
     { "$mergeObjects":
      [ "$_id",
       { "total_minutes_played": "$total_minutes_played"
       }
      ]
     }
    }
]
```

Y obtenemos el siguiente resultado al recorrer el cursor generado del pipeline de agregación de la última colección creada.

```
[90]: for i in db.cappearancesplayer.aggregate(pipeline):
          print(i)
```

```
{'competition_id': 'IT1', 'ageRange': '3. De 23 a 27', 'total_minutes_played': 479996}
{'competition_id': 'GB1', 'ageRange': '3. De 23 a 27', 'total_minutes_played': 467614}
{'competition_id': 'ES1', 'ageRange': '3. De 23 a 27', 'total_minutes_played': 356556}
{'competition_id': 'ES1', 'ageRange': '5. De 30 a 35', 'total_minutes_played': 327726}
{'competition_id': 'L1', 'ageRange': '3. De 23 a 27', 'total_minutes_played': 323452}
{'competition_id': 'FR1', 'ageRange': '3. De 23 a 27', 'total_minutes_played': 315980}
{'competition_id': 'ES1', 'ageRange': '4. De 27 a 30', 'total_minutes_played': 291212}
{'competition_id': 'GB1', 'ageRange': '2. De 18 a 23', 'total_minutes_played': 264920}
{'competition_id': 'FR1', 'ageRange': '2. De 18 a 23', 'total_minutes_played': 263468}
{'competition_id': 'IT1', 'ageRange': '4. De 27 a 30', 'total_minutes_played': 253168}
{'competition_id': 'GB1', 'ageRange': '4. De 27 a 30', 'total_minutes_played': 240168}
{'competition_id': 'FR1', 'ageRange': '4. De 27 a 30', 'total_minutes_played': 225698}
{'competition_id': 'IT1', 'ageRange': '2. De 18 a 23', 'total_minutes_played': 222822}
{'competition_id': 'ES1', 'ageRange': '2. De 18 a 23', 'total_minutes_played': 215994}
{'competition_id': 'L1', 'ageRange': '4. De 27 a 30', 'total_minutes_played': 213594}
{'competition_id': 'IT1', 'ageRange': '5. De 30 a 35', 'total_minutes_played': 205936}
{'competition_id': 'GB1', 'ageRange': '5. De 30 a 35', 'total_minutes_played': 197770}
{'competition_id': 'L1', 'ageRange': '5. De 30 a 35', 'total_minutes_played': 170400}
{'competition_id': 'L1', 'ageRange': '2. De 18 a 23', 'total_minutes_played': 161822}
{'competition_id': 'FR1', 'ageRange': '5. De 30 a 35', 'total_minutes_played': 157948}
{'competition_id': 'FR1', 'ageRange': '6. Mayor a 35', 'total_minutes_played': 42416}
{'competition_id': 'ES1', 'ageRange': '6. Mayor a 35', 'total_minutes_played': 42182}
{'competition_id': 'IT1', 'ageRange': '6. Mayor a 35', 'total_minutes_played': 32546}
{'competition_id': 'FR1', 'ageRange': '1. Menor a 18', 'total_minutes_played': 23450}
{'competition_id': 'GB1', 'ageRange': '6. Mayor a 35', 'total_minutes_played': 20456}
{'competition_id': 'L1', 'ageRange': '6. Mayor a 35', 'total_minutes_played': 11520}
{'competition_id': 'GB1', 'ageRange': '1. Menor a 18', 'total_minutes_played': 5480}
{'competition_id': 'L1', 'ageRange': '1. Menor a 18', 'total_minutes_played': 4258}
{'competition_id': 'ES1', 'ageRange': '1. Menor a 18', 'total_minutes_played': 1734}
{'competition_id': 'IT1', 'ageRange': '1. Menor a 18', 'total_minutes_played': 820}
```

Por otro lado, generamos un nuevo pipeline de agregación para evaluar la cantidad de penalizaciones que se tienen por cada competición.

```
[104]: pipeline = [
           {
               '$group': {
                   '_id': {
                       "competition_id":"$current_club_domestic_competition_id"
                   },
                   'total_yellow_cards': {'$sum': f'${"total_yellow_cards"}'},
                   'total_red_cards': {'$sum': f'${"total_red_cards"}'},
               }
           },
           {
               "$sort": {
                   "total_yellow_cards":-1,
                   "total_red_cards":-1,
               }
           },
           { "$replaceWith":
            { "$mergeObjects":
             [ "$_id",
              {
               "total_yellow_cards": "$total_yellow_cards",
               "total_red_cards": "$total_red_cards"
              }
             ]
            }
           }
       ]
```

```
[95]: for i in db.cappearancesplayer.aggregate(pipeline):
          print(i)
```

```
{'competition_id': 'GB1', 'total_yellow_cards': 2474, 'total_red_cards': 34}
{'competition_id': 'L1', 'total_yellow_cards': 1832, 'total_red_cards': 38}
{'competition_id': 'ES1', 'total_yellow_cards': 2976, 'total_red_cards': 112}
{'competition_id': 'FR1', 'total_yellow_cards': 1730, 'total_red_cards': 64}
{'competition_id': 'IT1', 'total_yellow_cards': 2590, 'total_red_cards': 48}
```

Finalmente, con el ultimo pipeline generado evidenciaremos a los 10 jugadores con mayor cantidad de goles y asistencias relacionados a su rango de edad correspondiente.

[102]:
```
pipeline = [
    {
        '$group': {
            '_id': {
                "name":"$name",
                "ageRange": "$ageRange"
            },
            'total_goals': {'$sum': f'${"total_goals"}'},
            'total_assists': {'$sum': f'${"total_assists"}'},
        }
    },
    {
        "$sort": {
            "total_goals":-1,
            "total_assists":-1,
        }
    },
    {
        "$limit":10
    },
    { "$replaceWith":
     { "$mergeObjects":
      [ "$_id",
       {
        "total_goals": "$total_goals",
        "total_assists": "$total_assists"
       }
      ]
     }
    }
]
```

[103]:
```
for i in db.cappearancesplayer.aggregate(pipeline):
    print(i)
```

```
{'name': 'Harry Kane', 'ageRange': '4. De 27 a 30', 'total_goals': 50, 'total_assists': 12}
{'name': 'Lautaro Martínez', 'ageRange': '3. De 23 a 27', 'total_goals': 48, 'total_assists': 10}
{'name': 'Erling Haaland', 'ageRange': '2. De 18 a 23', 'total_goals': 46, 'total_assists': 12}
{'name': 'Kylian Mbappé', 'ageRange': '3. De 23 a 27', 'total_goals': 46, 'total_assists': 8}
{'name': 'Victor Osimhen', 'ageRange': '3. De 23 a 27', 'total_goals': 46, 'total_assists': 4}
{'name': 'Alexandre Lacazette', 'ageRange': '5. De 30 a 35', 'total_goals': 40, 'total_assists': 2}
{'name': 'Serhou Guirassy', 'ageRange': '3. De 23 a 27', 'total_goals': 40, 'total_assists': 2}
{'name': 'Loïs Openda', 'ageRange': '2. De 18 a 23', 'total_goals': 36, 'total_assists': 12}
{'name': 'Mohamed Salah', 'ageRange': '5. De 30 a 35', 'total_goals': 34, 'total_assists': 24}
{'name': 'Wissam Ben Yedder', 'ageRange': '5. De 30 a 35', 'total_goals': 34, 'total_assists': 8}
```

Conclusiones

Respecto a los análisis realizados se concluye lo siguiente:

- Los jugadores que poseen grandes alturas usualmente están relacionados a posiciones defensivas dentro del futbol, si bien esto es lo que reflejan los datos posee cierta relación con la realidad ya que los existen una gran cantidad de jugadores porteros que tienen tallas altas sin embargo esto no garantiza que sean mejores o peores que los que tengan una talla más baja que el promedio
- Por otro lado, al analizar a los jugadores más jóvenes podemos iniciar un proceso de evolución de los mismos potenciando así sus cualidades para que puedan tener un mejor desempeño dentro de los primeros equipos de su club de futbol. Casos objetivos que se reflejaron en los resultados de jugadores jóvenes son los de Ethan Mbappé, Lamine Yamal y Marc Guiu; los cuales ya han sido promovidos a la primera división de sus clubes y han tenido desempeños destacados en algunos partidos oficiales.
- Por otro lado, al analizar los estadios con mayor capacidad nos brindan opciones de donde poder celebrar partidos de futbol que tengan una gran acogida de asistentes, como partidos del mundial que próximamente se celebraran en países europeos, dentro de España destaca el estadio Santiago Bernabéu que si bien la data no cuenta con los cambios que se han realizado a los estadios nos da un perfil de cuales podrían ser las posibles opciones para eventos de tal magnitud.
- En el análisis de jugadores con mayor valor de mercado de las 5 grandes ligas verdaderamente evidencias un gran % de asertividad ya que dichos jugadores que se obtuvieron han mantenido un nivel alto dentro de sus ligas y otras competiciones por lo cual se mantienen con un valor de mercado superior a otros jugadores.
- Tomando en cuenta el resultado obtenido de la relación entre la competición, el rango de edad de jugadores y el total de minutos jugados podemos evidenciar que verdaderamente los jugadores con un rango de edad entre 23 a 27 son los que disputan una mayor cantidad de minutos debido a que a esa edad los jugadores ya consolidan su formación y sus posiciones siendo de mayor utilidad para los entrenadores. A diferencia de los jugadores que recién debutan y tienen menor de 18 años que existen casos en los cuales se consolidan rápido debido a sus cualidades, pero en caso no posean las habilidades para destacar juegan pocos minutos mientras consiguen afianzarse en los equipos a los cuales pertenecen.
- Respecto al análisis de tarjetas amarillas o rojas obtenido podemos establecer una relación de concordancia debido a que por ejemplo la liga Española (ES1) tiene una mayor cantidad de tarjetas tanto amarillas como rojas y esto se evidencia en como los partidos a veces suelen ser mas cortados por las constantes faltas cometidas, por otro lado vemos que la liga Inglesa (GB1) se encuentra en una posición intermedia esto debido a

que los partidos son más fluidos y las decisiones arbitrales son menos polémicas.

- Finalmente, la relación de goles y asistencias que poseen los 10 jugadores con mayores apariciones es la correcta ya que por ejemplo obtuvimos que Harry Kane tuvo la mayor cantidad de goles y asistencias durante el 2023 pero también tenemos goleadores que son mundialmente conocidos que marcan objetivamente unas grandes cantidades de goles en cada temporada.
- Podemos concluir que los resultados obtenidos de los análisis tienen gran nivel de precisión ya que obtenemos resultados que se asemejan a la realidad en gran proporción.