Repositorio: https://github.com/Iker-Marcelo/-Iker-_actividadBDNOSQL_ATBD

1. **Seleccionar BBDD**

Dentro de las opciones disponibles, decidí utilizar MongoDB. El motivo principal, fue que es la BBDD Nosql con la que más familiarizado estoy. Sin embargo, después de estudiar el resto de opciones disponibles, sigo creyendo que era la mejor opción.

Las jugadoras tienen datos anidados, como estadísticas avanzadas y defensivas, que encajan perfectamente en el modelo de documentos de MongoDB. Mongo permite almacenar toda la información relevante en un solo documento, evitando la complejidad de múltiples tablas o nodos.

Además, MongoDB dispone de un lenguaje de consultas potente y fácil de usar.

En cuanto a las otras opciones, creo que las consultas en Cassandra son menos intuitivas para este caso de uso, InfluxDB no tiene la flexibilidad de mongo para el tipo de relaciones jerárquicas utilizadas en el proyecto, si no que está más bien orientado a manejar series temporales y métricas. Por último, Neo4j, creo que hubiese sido una buena opción, pero preferí usar mongo por las ventajas explicadas anteriormente.

2. **Análisis del dataset**

El dataset utilizado para esta práctica ha sido extraído de Kaggle. Proporciona un conjunto extenso de datos relacionados con el rendimiento en el fútbol femenino en las cinco principales ligas de Europa. El dataset contiene varios csv, pero solo he utilizado “all_players.csv”.

Este csv incluye las siguientes columnas:

Este dataset contiene estadísticas detalladas sobre jugadoras de fútbol femenino y cubre una variedad de métricas de rendimiento, desde aspectos ofensivos y defensivos hasta contribuciones generales en el campo. Aquí tienes una explicación de cada columna, categorizadas por su propósito:

1. Información básica del jugador y su equipo:

- Player: Nombre de la jugadora.
- Nation: País o nacionalidad de la jugadora.
- Pos: Posición de la jugadora en el campo (por ejemplo, portera, defensa, mediocampo, delantera).
- Squad: Nombre del equipo al que pertenece la jugadora.
- Age: Edad de la jugadora.
- Born: Año de nacimiento de la jugadora.
- Player_id: Identificador único de la jugadora.
- Squad_id: Identificador único del equipo.

2. Participación en partidos:

- MP: Total de partidos jugados.
- Starts: Total de partidos en los que la jugadora fue titular.
- Min: Total de minutos jugados.
- Mn_by_MP: Minutos promedio por partido.
- Mn_by_Start: Minutos promedio como titular.
- Compl: Número de partidos completados sin ser sustituida.
- Subs: Número de veces que fue sustituida.
- Mn_by_Sub: Minutos promedio jugados antes de ser sustituida.
- unSub: Número de partidos donde no fue sustituida.

3. Estadísticas de goles y asistencias:

- Gls: Goles anotados.
- Ast: Asistencias realizadas.
- G-PK: Goles excluyendo penales.
- PK: Penales convertidos.
- PKatt: Penales intentados.
- xG: Goles esperados según las oportunidades creadas.
- npxG: Goles esperados excluyendo penales.
- xAG: Asistencias esperadas.
- G-xG: Diferencia entre goles reales y goles esperados.
- np:G-xG: Diferencia entre goles no penales y goles esperados no penales.

4. Rendimiento de disparos:

- Sh: Total de disparos realizados.
- SoT: Disparos a puerta (on target).
- G_by_Sh: Porcentaje de disparos que terminan en gol.
- G_by_SoT: Porcentaje de disparos a puerta que terminan en gol.
- Dist: Distancia promedio de los disparos (en metros).
- FK: Disparos provenientes de tiros libres.
- npxG_by_Sh: Goles esperados por disparo excluyendo penales.

5. Contribuciones ofensivas:

- KP: Pases clave (asistencias potenciales).
- LastThird: Pases realizados en el último tercio del campo.
- PPA: Pases al área de penal.
- CrsPA: Centros al área de penal.
- Prog: Pases progresivos hacia la portería rival.
- SCA: Acciones que llevan a un disparo (Shot Creating Actions).
- SCA_PassLive: Pases en juego abierto que resultan en un disparo.
- SCA_PassDead: Pases en balones detenidos que resultan en un disparo.
- SCA_Drib: Dribles que llevan a un disparo.
- SCA_Sh: Disparos que generan rebotes aprovechados por un compañero.
- SCA_Fld: Faltas recibidas que llevan a un disparo.
- SCA_Def: Recuperaciones defensivas que llevan a un disparo.

- SCA: Acciones que llevan a un gol (Goal Creating Actions).
- GCA_PassLive: Pases en juego abierto que resultan en gol.
- GCA_PassDead: Pases en balones detenidos que resultan en gol.
- GCA_Drib: Dribles que llevan a un gol.
- GCA_Sh: Disparos que generan rebotes aprovechados por un compañero para gol.
- GCA_Fld: Faltas recibidas que llevan a un gol.
- GCA_Def: Recuperaciones defensivas que llevan a un gol.

6. Estadísticas defensivas:

- Tackles_Tkl: Total de entradas realizadas.
- Tackles_TklW: Entradas exitosas.
- Tackles_Def3rd: Entradas realizadas en el tercio defensivo.
- Tackles_Mid3rd: Entradas realizadas en el tercio medio.
- Tackles_Att3rd: Entradas realizadas en el tercio ofensivo.
- VsDribbles_Tkl: Entradas realizadas contra jugadores en drible.
- VsDribbles_Att: Intentos totales de entrada contra dribles.
- VsDribbles_Past: Dribles exitosos en contra de la jugadora.
- Blocks_Blocks: Bloqueos totales (disparos, pases, etc.).
- Blocks_Sh: Bloqueos de disparos.
- Blocks_Pass: Bloqueos de pases.
- Int: Intercepciones.
- Tkl_plus_Int: Suma de entradas e intercepciones.
- Clr: Despejes.
- Err: Errores que conducen a disparos o goles.

7. Posesión y control del balón:

- Touches_Touches: Total de toques del balón.
- Touches_DefPen: Toques en el área de penal propia.
- Touches_Def3rd: Toques en el tercio defensivo.
- Touches_Mid3rd: Toques en el tercio medio.
- Touches_Att3rd: Toques en el tercio ofensivo.
- Touches_AttPen: Toques en el área de penal rival.
- Touches_Live: Toques en jugadas abiertas.
- Dribbles_Succ: Dribles exitosos.
- Dribbles_Att: Intentos de drible.
- Dribbles_Mis: Dribles fallidos.
- Dribbles_Dis: Veces que pierde el balón durante un drible.
- Receiving_Rec: Veces que recibe un pase.
- Receiving_Prog: Veces que recibe un pase progresivo.

8. Estadísticas de faltas y penales:

- Fls: Faltas cometidas.
- Fld: Faltas recibidas.
- Off: Veces fuera de juego.
- PKwon: Penales ganados.

- PKcon: Penales cometidos.
- OG: Goles en propia puerta.
- Recov: Recuperaciones de balón.

9. Duelos aéreos:

- AerialDuels_Won: Duelos aéreos ganados.
- AerialDuels_Lost: Duelos aéreos perdidos.

10. Estadísticas avanzadas del equipo (Team Stats):

- TS_PPM: Puntos promedio por partido cuando la jugadora está en el campo.
- TS_onG: Goles anotados por el equipo cuando la jugadora está en el campo.
- TS_onGA: Goles recibidos por el equipo cuando la jugadora está en el campo.
- TS_Diff: Diferencia de goles con la jugadora en el campo.
- TS_On-Off: Impacto del rendimiento del equipo con o sin la jugadora.
- TS(xG)_onxG: Goles esperados cuando la jugadora está en el campo.
- TS(xG)_onxGA: Goles esperados en contra cuando la jugadora está en el campo.
- TS(xG)_xGDiff: Diferencia de goles esperados con la jugadora en el campo.
- TS(xG)_On-Off: Impacto de los goles esperados con o sin la jugadora.

Sin embargo, para simplificar, solo he utilizado los 6 primeros apartados.

3. Definir esquema y sentencias de creación

La base de datos women-football contiene una única colección llamada players, la cual contiene todos los datos sobre las jugadoras explicados en el apartado anterior.

La creación de la colección se realiza mediante la función create_players_collection, que utiliza el método create_collection de MongoDB con un validador basado en JSON Schema. Gracias a esto me aseguro de que los documentos insertados cumplan con la estructura definida.

Además, para poder cumplir con la práctica he creado la columna start_year, que refleja el año de inicio en el fútbol profesional de la jugadora.

4. Inserción de registros

Para leer el csv e insertar los registros en la bbdd, primero utilicé pandas para conertirlo a un dataframe.

Después, para poder hacer la consulta de filtrar por equipo, me aseguré de que se metiesen 5 jugadoras que jugasen algún equipo que empiece por Manchester, ya que a pesar de que existían en el csv, no había ninguna entre las 100 primeras filas.

Muchos campos en el dataset original contienen valores nulos o faltantes. Por lo que se establecieron reglas para manejar estos casos asignando valores por defecto. Además, para la columna start_year, se genera un número aleatorio entre 2012 y 2024.

Me aseguré de que todos los campos cumplieran con los tipos de datos especificados en el esquema. Por ejemplo, convirtiendo las edades y años de nacimiento a enteros y las estadísticas avanzadas a float.

Se utilizó la biblioteca pymongo para establecer una conexión con la base de datos MongoDB, y se insertaron los 100 documentos en la colección players utilizando una operación de inserción múltiple.

5. **Sentencias de Modificación**

Para cumplir con el requisito de modificar dos registros cambiando el nombre de las jugadoras a mayúsculas, se implementaron sentencias de modificación en la base de datos MongoDB.

```
262
263  def update_player_names_to_uppercase(db):
264      players_collection = db['players']
265      players = list(players_collection.find().limit(2))
266      for player in players:
267          updated_name = player['name'].upper()
268          players_collection.update_one(
269              {'_id': player['_id']},
270              {'$set': {'name': updated_name}}
271          )
272          print(f"Updated {player['name']} to {updated_name}")
273
```

Para realizar la modificación, seleccioné los dos primeros registros, y modifique el campo name usando la función upper() para poner el nombre en mayúsculas. Después, utilicé la operación update_one, que permite actualizar un único documento que cumpla con un criterio específico. Este criterio fue el id de la jugadora. Para asignar el nuevo valor al campo utilicé el operador $set.

6. **Consultas**

A continuación muestro una imágen con las funciones de las consultas solicitadas en el enunciado.

```python
def find_players_started_after_2020(db):
    players_collection = db['players']
    result = players_collection.find({'start_year': {'$gt': 2020}})
    for player in result:
        print(player)

def find_players_by_team_name(db):
    players_collection = db['players']
    result = players_collection.find({'team.name': {'$regex': '^Manchester'}})
    for player in result:
        print(player)

def find_players_by_nationality(db, country):
    players_collection = db['players']
    result = players_collection.find({'nationality': country})
    for player in result:
        print(player)
```

La primera consulta busca en la colección players todos los documentos donde el campo start_year es mayor a 2020. La consulta se realiza utilizando el operador de comparación $gt.

La siguiente consulta, busca jugadoras cuyos equipos tienen nombres que comienzan con "Manchester". La consulta utiliza una expresión regular con el operador $regex para buscar coincidencias en el campo ‘team.name’. La expresión regular '^Manchester' busca cadenas que comiencen con "Manchester".

La última consulta, devuelve todas las jugadoras con nacionalidad española. Para ello, se ejecuta una consulta donde el campo nationality coincide exactamente con el valor proporcionado.

Para evaluar el rendimiento de las consultas y entender su eficiencia, implementé una función que mide el tiempo de ejecución para cada una de las consultas.

```python
def measure_query_time(db):
    players_collection = db['players']

    start_time = time.time()
    result = list(players_collection.find({'start_year': {'$gt': 2020}}))
    end_time = time.time()
    print(f"Consulta 1: Tiempo de ejecución: {end_time - start_time:.4f} segundos. Resultados: {len(result)}")

    start_time = time.time()
    result = list(players_collection.find({'team.name': {'$regex': '^Manchester', '$options': 'i'}}))
    end_time = time.time()
    print(f"Consulta 2: Tiempo de ejecución: {end_time - start_time:.4f} segundos. Resultados: {len(result)}")

    start_time = time.time()
    result = list(players_collection.find({'nationality': 'es ESP'}))
    end_time = time.time()
    print(f"Consulta 3: Tiempo de ejecución: {end_time - start_time:.4f} segundos. Resultados: {len(result)}")
```

El funcionamiento de esta función es sencillo. Se captura el tiempo actual antes de ejecutar la consulta, utilizando la función time.time(). Después, se ejecuta la consulta y se almacena el resultado, y por último, se captura nuevamente el tiempo actual después de que la

consulta ha finalizado. Para saber lo que tardó en realizarse la consulta, se resta el tiempo que se capturó al final de la ejecución por el que se capturó al inicio.

Para mejorar la legibilidad del resultado de las consultas, implementé un menú que permite elegir cual de las 3 consultas realizar.

```
322     while True:
323 
324         print("\n=== MENÚ DE CONSULTAS ===")
325         print("1. Salir")
326         print("2. Filtrar por start_year mayor a 2020")
327         print("3. Filtrar por equipos que empiecen por 'Manchester'")
328         print("4. Filtrar por nacionalidad")
329 
330         choice = input("Elige una opción (1-4): ")
331 
332         if choice == "1":
333             print("Saliendo del programa...")
334             break
335         elif choice == "2":
336             print("\nFILTRAR POR start_year MAYOR A 2020")
337             find_players_started_after_2020(db=db)
338         elif choice == "3":
339             print("\nFILTRAR POR EQUIPOS QUE EMPIECEN POR 'Manchester'")
340             find_players_by_team_name(db=db)
341         elif choice == "4":
342             print(f"\nFILTRAR POR NACIONALIDAD es ESP")
343             find_players_by_nationality(db, 'es ESP')
344         else:
345             print("Opción no válida. Por favor, elige una opción del 1 al 4.")
```

7. Bibliografía

Enlaces consultados:
https://alud.deusto.es/course/view.php?id=27469
https://www.mongodb.com/docs/
https://openwebinars.net/blog/ventajas-y-desventajas-de-mongodb/

Dataset:
https://www.kaggle.com/datasets/thedevastator/uncovering-female-football-success-in-top-europe?select=all_players.csv

Prompts de chatgpt:
https://chatgpt.com/share/674e002c-c7b0-8012-b1a3-783065fb9946