ETL (Extracción, Transformación y Carga)

Esta fase se encarga de obtener los datos de sus fuentes originales, limpiarlos, estructurarlos y prepararlos para el análisis posterior.

1.1. Extracción (Extract)

Se obtienen datos de dos fuentes principales:

- **Datos de eventos de partidos (StatsBomb):**
    - Se utiliza la librería statsbombpy para acceder a la API de StatsBomb.
    - Se identifican las competiciones disponibles y se filtra específicamente por "La Liga" (temporada 2015/2016).
    - Se obtiene el match_id de todos los partidos de La Liga en esa temporada.
    - Se itera a través de los match_id para extraer los eventos detallados de cada partido (ej. pases, tiros, duelos, etc.).
    - Se almacenan los eventos en una lista de diccionarios que luego se convierte en un DataFrame de Pandas.
    - Se obtienen los datos de los equipos que participaron en cada partido,

- **Extracción de Datos de Jugadores desde Transfermarkt (RapidAPI)**

Este bloque de código se encarga de recopilar información de jugadores de fútbol de la plataforma Transfermarkt a través de API.

- **Extracción de IDs de Equipos de La Liga (2015):**
- Se realiza una solicitud GET a la API para obtener la tabla de clasificación de la liga española (La Liga, ID: ES1) de la temporada 2015.
- La respuesta JSON de la API se parsea para extraer los IDs de todos los equipos participantes en esa temporada y se almacenan en una lista.
- **Extracción de Plantillas de Equipos:**
- Se itera a través de cada `id` de equipo obtenido en el paso anterior.
- Para cada equipo, se realiza una nueva solicitud GET a la API para obtener la plantilla completa del equipo para la temporada 2015.
- Las respuestas JSON se almacenan en un diccionario, donde la clave es el ID del equipo.
- **Extracción de IDs de Jugadores:**
- Se recorre la estructura de datos anidada de las plantillas de los equipos para extraer todos los IDs de los jugadores de cada equipo y se añaden a una lista.
- **Extracción de Perfiles de Jugadores:**
- Se itera sobre cada id de jugador.
- Para cada jugador, se hace una solicitud GET a la API para obtener su perfil detallado.

- Los perfiles se almacenan en un diccionario.
- **Manejo de Errores/Reintentos:** Si algunas solicitudes de perfil de jugador fallan (indicado por un mensaje de error en la respuesta JSON), se identifican esos IDs (no_añadidos) y se reintentan las solicitudes. Si hay muchos es necesario generar una nueva API-Key ya que la API tiene un límite de peticiones en la versión gratuita.
- **Consolidación y Almacenamiento:**
- Se extrae la sección playerProfile de la información de cada jugador y se compila en una lista de diccionarios.
- Esta lista se convierte en un DataFrame de Pandas .
- Finalmente, el DataFrame se guarda como un archivo CSV.

- **Extracción de Datos de Equipos y Jugadores desde FBref (Web Scraping con Selenium y API)**

Este bloque se centra en obtener datos de equipos y estadísticas de jugadores de la plataforma FBref, utilizando tanto web scraping como su API.

- **Importación y Configuración de Selenium:** Se importan librerías para web scraping (`selenium`, `BeautifulSoup`)
- **Obtención de IDs de Equipos de FBref:**
  - Se navega a la URL de la tabla de la liga española 2015-2016 en FBref.
  - Se localizan todos los enlaces de los equipos dentro de la tabla y se extrae el ID de cada equipo de sus URLs.
- **Generación de API Key para FBref:** Se realiza una solicitud POST a la API de FBref para generar una clave de API que se utilizará para futuras solicitudes.
- **Extracción de Información de Equipos (Plantillas):**
  - Se itera sobre cada team_id y season_id obtenidos.
  - Se realiza una solicitud GET a la API de FBref (https://fbrapi.com/) para obtener la plantilla de cada equipo.
  - Las respuestas se almacenan en el diccionario jugadores.
- **Consolidación de Datos de Plantillas:**
  - Se recorre la estructura anidada de los datos de plantillas para extraer la información individual de cada jugador.
  - Se añade el season_id y team_id a cada registro de jugador.
  - Estos datos se consolidan en una lista de diccionarios y se convierten en un DataFrame de Pandas.
- **Extracción de Estadísticas de Temporada de Jugadores:**
  - Se realiza una nueva serie de solicitudes GET a la API de FBref para obtener estadísticas detalladas por temporada para cada jugador y equipo de la liga.

.

- **Normalización y Consolidación de Estadísticas de Jugadores:**
    - Se itera sobre los datos para extraer la información de cada jugador.
    - Se separan los datos en columnas individuales en un nuevo diccionario para cada jugador.
    - Estos diccionarios se añaden a una lista, que luego se convierte en un DataFrame de Pandas.
- **Almacenamiento:** El DataFrame se guarda como un archivo CSV.

- **Extracción de Clasificación de Liga y Lesiones desde Ceroacero.es (Web Scraping con Selenium)**

Este bloque se encarga de extraer la clasificación de la liga jornada a jornada y la información de lesiones de los equipos de la web Ceroacero.es.

- **Importación y Configuración de Selenium:** Se reimportan librerías y se configura la automatización del navegador.
- **Generación de URLs por Jornada:**
    - Se crea un diccionario que contiene una URL específica para cada una de las 38 jornadas de la liga española 2015-2016 en Ceroacero.es.
- **Extracción de la Clasificación por Jornada:**
    - Se itera a través de cada URL de jornada.
    - El navegador accede a cada URL. Se implementa lógica para aceptar cookies si aparecen.
    - Se localiza la tabla de clasificación.
    - Se extraen los encabezados de la tabla, con lógica para limpiar y asegurar nombres de columna apropiados.
    - Se extraen los datos de cada fila de la tabla (posición, equipo, puntos, partidos jugados, etc.). Se añade el número de jornada a cada registro.
    - Los datos de cada jornada se consolidan en un DataFrame.
- **Limpieza y Normalización de la Clasificación:**
    - Se elimina la columna Logo del DataFrame por no ser necesaria.
    - **Normalización de Nombres de Equipos:** Se comparan los nombres de los equipos de Ceroacero con los nombres de StatsBomb (cargados de Tabla_Partidos.csv) utilizando una función de normalización (src.normalizacion_nombres) para crear un mapeo uniforme.
    - Se utiliza este mapeo para estandarizar los nombres de los equipos en la columna Equipo de df_jornadas.
    - Se fusiona el DataFrame de jornadas con los IDs de clubes de df_wyscout (de Tabla_Partidos.csv) para añadir el clubid a la clasificación.
    - Se renombra la columna id_club_local_L a clubid.
- **Almacenamiento de Clasificación:** El DataFrame combinado se guarda como CSV.

- **Extracción de URLs y Nombres de Equipos para Lesiones:**
    - Se vuelve a iniciar Selenium para navegar a la página principal de la liga en Ceroacero.es.
    - Se extraen los identificadores únicos (segmentos de URL) y los nombres de todos los equipos presentes en la tabla.
    - Se genera una URL específica para la sección de "indisponibles" (lesionados/sancionados) de cada equipo.
- **Extracción de Datos de Lesiones por Equipo:**
    - Se itera sobre las URLs de "indisponibles" de cada equipo.
    - El navegador accede a cada URL, aceptando cookies si es necesario.
    - Se localizan las tablas de lesiones (si existen).
    - Se extraen los encabezados y los datos de cada fila (jugador, tipo de lesión, etc.).
    - Estos datos se concatenan en un DataFrame.
- **Extracción de Nombres Completos de Jugadores (para Lesiones):**
    - A partir de los href de los jugadores lesionados, se vuelve a utilizar Selenium para visitar la página de perfil de cada jugador en Ceroacero.es.
    - Desde el perfil, se extrae el nombre completo del jugador para asegurar consistencia.
- **Mapeo y Normalización Final de Lesiones:**
    - Se mapean los nombres completos de los jugadores y los nombres estandarizados de los equipos al DataFrame.
- **Almacenamiento de Lesiones:** El DataFrame final se guarda como CSV.

- **CSV Wyscout:**
    - Desde una web con acceso privado se descargan y concatenan en excel varios archivos con estadísticas de todos los equipos en cada partido de la liga.

1.2. Transformación (Transform)

Una vez extraídos los datos, se realizan diversas operaciones para limpiarlos, enriquecerlos y adaptarlos para el análisis:

- **Transformación de datos jugadores:**
    - Los datos extraídos con la información de cada jugador se analizan y modifican para una correcta carga a la BBDD.
    - Se modifican los nombres que aparecían duplicados como “Sergio Álvarez” ya que corresponden a dos jugadores diferentes con ID diferente.
    - Se normalizan nombres e IDs de equipos y jugadores para que coincidan con la tabla de Eventos.
    - Se eliminan columnas innecesarias.
- **Transformación de datos lesiones:**
    - Los datos extraídos con la información de todas las lesiones registradas se modifican para una correcta carga a la BBDD.

    - Se crea un ID único por lesión y tipo.
    - Se normalizan nombres e IDs de equipos y jugadores para que coincidan con la tabla de Eventos.
    - Se eliminan columnas innecesarias.
- **Transformación de datos equipos:**
    - Los datos extraídos con la información de cada jugador se analizan y modifican para una correcta carga a la BBDD.
    - Se normalizan nombres e IDs de equipos y jugadores para que coincidan con la tabla de Eventos.
    - Se eliminan columnas innecesarias.

1.3. Carga (Load)

Tras la transformación de los datos se carga cada archivo en una tabla de la BBDD creada en PostgreSQL.