PROGRAMACIÓN
HITO GRUPAL

Jose Antonio Ures
Allam E. Miranda Carrasco
Fernando Trujillo

Investigación

1. **Fuentes de datos**

   Data Lake
   Es un tipo de almacenamiento o repositorio de un gran conjunto de datos en bruto que todavía no tienen una finalidad definida.

   Su nombre hace referencia a la flexibilidad, acceso compartido y manejo en tiempo real de un gran volumen de datos entre diferentes usuarios que se benefician o trabajan con dichos datos. Evidentemente, debido a la gran cantidad de información que manipula, trabaja con herramientas del Big Data.
   Su principal objetivo radica en el almacenamiento esquematizado de los datos para poder procesar según los intereses de estudio.
   Uno de sus factores más importantes es que los datos almacenados se actualizan y transforman constantemente, se mantiene vivo gracias a su gestión permanente de la información.
   Es un tipo de repositorio muy utilizado en el ámbito empresarial debido a sus ventajas como estrategia empresarial al reducir costes, aumentar la asertividad en la toma de decisiones y permitir que los usuarios de la empresa unifiquen su conocimiento sobre la información obtenida.
   Sus principales características:
   - Ayuda al manejo del Big Data
   - La posibilidad de gestionar y transformar los datos mientras se encuentran almacenados.
   - La cercanía e interacción que permite con los usuarios.
   - Cuenta con muchas herramientas y productos para cumplir con su objetivo de almacenamiento eficaz.
   - Los metadatos se gestionan de manera automatizada ahorrando tiempo y trabajo a la hora de implementar este repositorio.
   - Modificar la privacidad de los datos y establecer quién tiene acceso, quién solo puede verlos, quién puede modificar o no la información original, etc.

2. **Hadoop y Spark**

   Hadoop
   Es una estructura de software de código abierto para almacenar datos y ejecutar aplicaciones en clústeres de hardware comercial. Proporciona almacenamiento masivo para cualquier tipo de datos, enorme poder de procesamiento y la capacidad de procesar tareas o trabajos concurrentes virtualmente ilimitados.

   Características:
   - Procesamiento distribuido.

- Eficiente.
- Económico.
- Fácilmente escalable.
- Tolerante a fallos.
- Open source.

Spark
Apache Spark es un motor de estadísticas unificado con el que se procesan datos a gran escala con módulos integrados para SQL, transmisión, aprendizaje automático y procesamiento de grafos. Spark puede ejecutarse en Apache Hadoop, Apache Mesos, Kubernetes, por sí solo, en la nube y en varias fuentes de datos.

Características:
- Procesamiento en memoria: Apache Spark es 100 veces más rápido en memoria y 10 veces más rápido en disco que Hadoop MapReduce, para ello necesita más recursos.
- Soporta múltiples lenguajes: Spark tiene APIs disponibles en los lenguajes Java, Scala, Python y R.
- Analítica avanzada: Para ello, soporta consultas SQL y su uso para Machine Learning con librerías de data science como MLlib y GraphX.
- Abstracción RDD (Resilient Distributed Dataset): consiste en una colección inmutable de elementos en memoria distribuída.
- Evaluación perezosa: Las transformaciones sobre los datos solo se resuelven al ejecutar una acción sobre ellos.

3. Python

Python y el Big Data es una de las combinaciones más valiosas a día de hoy. los datos y la información sirven para ser más eficientes, tomar buenas decisiones y conocer a tus clientes.
De hecho, el lenguaje de programación Python y el *big data* están muy relacionados. No por nada se lo considera el mejor lenguaje de programación para el análisis de datos.

Lo cierto es que escoger un lenguaje de programación para el *big data* depende del proyecto que tengas entre manos. No obstante, sea cual sea tu objetivo, Python siempre será una opción de lo más adecuada. Esto se debe a que, además de ser un lenguaje en desarrollo constante, su código simple y sus inmensas bibliotecas (SciPy, Pandas, Numpy o Scikit-Learn...) hacen que sea la opción preferida para la mayoría de los programadores. Algo lógico, ya que comparado con otros lenguajes, Python tiene un código y una sintaxis simple que hace que sea muy fácil de aprender. Con unas pocas líneas de código, puedes ejecutar programas sin más complicaciones.

Además, y esto es una gran ventaja, es de código abierto. Por ello, cualquier persona tiene acceso a sus recursos de forma gratuita. Y, gracias a la gran comunidad de usuarios de Python, podrás encontrar las respuestas a tus dudas fácilmente. Así mismo, otra gran ventaja del lenguaje es su gran velocidad de procesamiento

La forma más común de usar Python para el análisis de datos es para crear y gestionar rápidamente varias estructuras de datos. Por ejemplo, la biblioteca Panda ofrece una gran cantidad de herramientas para analizar, manipular e incluso representar las estructuras de datos y conjuntos de datos complejos.

Scala:

Scala es uno de los lenguajes de programación más usados para el manejo y desarrollo del Big Data. Esto es gracias a lo intuitivo, conciso y preciso que es como lenguaje de programación.

Powebi:

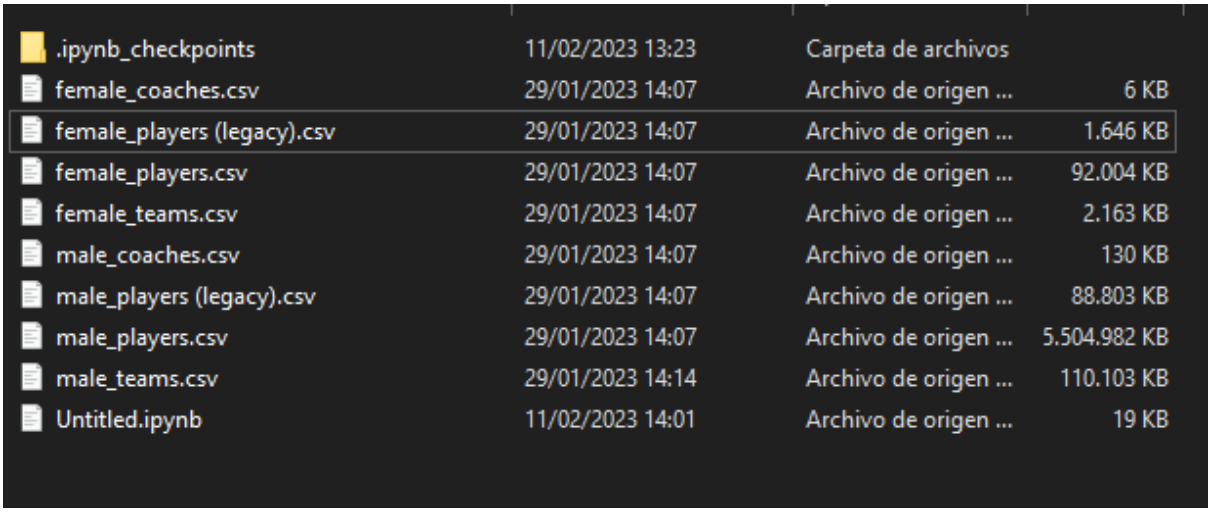
Power BI proporciona servicios de BI basados en la nube, conocidos como "Power BI Services", junto con una interfaz basada en escritorio, denominada "Power BI Desktop". Ofrece capacidades de almacenamiento de datos, incluyendo preparación de datos, descubrimiento de datos y paneles interactivos. En marzo de 2016, Microsoft lanzó un servicio adicional llamado "Power BI Embedded" en Azure, su plataforma en la nube. Uno de los principales diferenciadores del producto es la capacidad de cargar visualizaciones personalizadas.

Tableau:

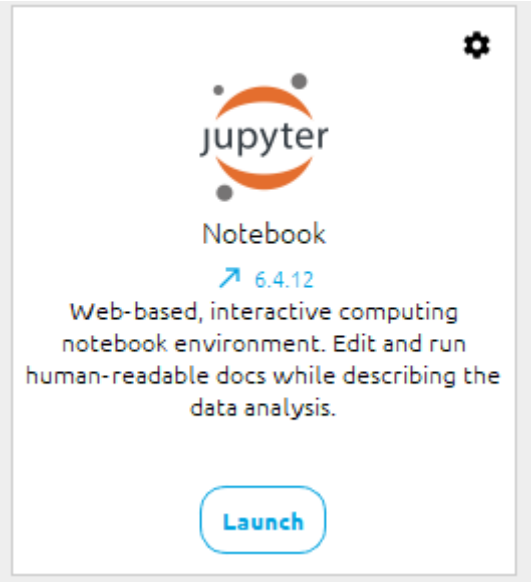
Tableau es una plataforma de análisis visual que transforma la manera en que usamos los datos para resolver problemas. Además, permite a las personas y las organizaciones sacar el máximo partido de los datos. Hace que sea más fácil explorar y administrar los datos. Asimismo, permite descubrir y compartir información más rápidamente a fin de generar grandes cambios en los negocios y en el mundo.

Implementación de la investigación

Para la fase 2 del proyecto usaremos distintas formas de ver los datos de un archivo csv pesado, en este caso vamos a usar una base de datos conseguida en **Kaggle** de todos los jugadores del mundo del FIFA 21, este archivo pesa unos 8GB dividido en distintos csv, en Big Data la mayor parte del trabajo se realiza en Jupyter NoteBook esto se debe a que la implementación de distintos softwares como puede ser Spark son más fáciles de implementar y da un rendimiento muy alto.



Tenemos una base de datos estructurada ya que no existe ningún dato no estructurado ya sea imágenes o audios, abrimos Jupyter con Conda ya que nos ofrece las librerías necesarias para el control de la información, en este caso Pandas y también haremos uso básico de PySpark.



Abrimos Jupyter desde Conda y entramos a nuestra carpeta del proyecto y vamos a crear un nuevo archivo de Python para importar las distintas librerías.

```
In [12]: import pandas as pd
```

Importamos pandas como pd y vamos a leer nuestro primer csv en este caso el de 'female_players.csv' que nos da la información de todas las jugadoras femeninas del mundo y todos los datos dentro del juego.

```
In [34]: df = pd.read_csv('female_players (legacy).csv', index_col='short_name')
```

Para esto vamos a crear un DataFrame, un DataFrame de Pandas es una estructura de datos en forma de tabla que permite almacenar y manipular datos de manera eficiente y sencilla en Python.

Sabiendo esto creamos nuestro dataframe de este csv y con index_col dejamos claro que queremos que nuestra primera columna sean los nombres de las jugadoras.

```
In [35]: df
```

Out[35]:

short_name	player_id	player_url	fifa_version	fifa_update	fifa_update_date	long_name	player_positions	overall	potential	value_eur	...	cdm
Alexia Putellas	227203	/player/227203/alexia-putellas-segura/230002	23	2	2022-09-26	Alexia Putellas Segura	CM, LW	92	92	NaN	...	83+3
S. Kerr	227125	/player/227125/sam-kerr/230002	23	2	2022-09-26	Samantha May Kerr	ST	91	91	134500000.0	...	61+3
A. Hegerberg	227310	/player/227310/ada-hegerberg/230002	23	2	2022-09-26	Ada Martine Stolsmo Hegerberg	ST	91	92	157000000.0	...	61+3
W. Renard	227316	/player/227316/wendie-renard/230002	23	2	2022-09-26	Wéndèleine Thérèse Renard	CB	91	91	89500000.0	...	87+3
A. Morgan	226301	/player/226301/alex-morgan/230002	23	2	2022-09-26	Alexandra Morgan Carrasco	ST	90	90	NaN	...	64+3
...	...	...	...	...	...	...	...	...	...	...	...	...
N. Ezurike	227385	/player/227385/nkem-ezurike/160002	16	2	2015-09-21	Nkem Ezurike	ST	64	71	NaN	...	41
V. Miranda	227485	/player/227485/valeria-miranda/160002	16	2	2015-09-21	Valeria Aurora Miranda Rodriguez	LB	64	72	NaN	...	60
L. Tucceri Cimini	228160	/player/228160/linda-tucceri-cimini/160002	16	2	2015-09-21	Linda Tucceri Cimini	LB, CB	64	71	NaN	...	64
F. Ibarra	228723	/player/228723/fabiola-ibarra/160002	16	2	2015-09-21	Claudia Fabiola Ibarra Muro	LM	63	68	NaN	...	43
A. Guajardo	228955	/player/228955/anisa-guajardo/160002	16	2	2015-09-21	Anisa Raquel Guajardo Braff	ST	61	66	NaN	...	42

3196 rows × 109 columns

Al hacer print de esto nos devuelve una tabla de más de 3000 filas con todos nuestros datos y columnas de el csv ahora vamos a realizar una serie de consultas como pueden ser el número de equipos que hay en cada país o cuantas jugadoras hay en cada país, cuál es la que más ritmo tiene, etc.

Para esto vamos a MatPlotLib y NumPy lo que nos permite ver gráficamente estas consultas, hay que dejar claro que estas librerías es para el manejo de datos de un volumen promedio o mediano para datos de mayor volumen es recomendable usar PySpark o Scala con Spark igualmente.

```
In [48]: df.sort_values(by=['potential'], ascending=False)
         df[['short_name', 'potential']]
```

Out[48]:

	short_name	potential
0	Alexia Putellas	92
1	S. Kerr	91
2	A. Hegerberg	92
3	W. Renard	91
4	A. Morgan	90
...	...	...
3191	N. Ezurike	71
3192	V. Miranda	72
3193	L. Tucceri Cimini	71
3194	F. Ibarra	68
3195	A. Guajardo	66

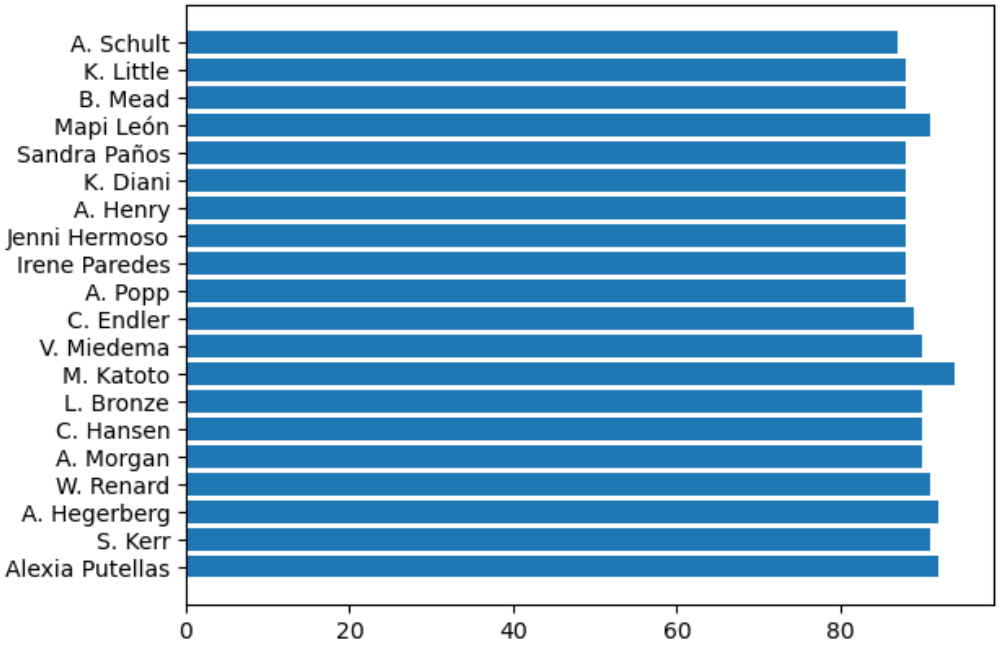
3196 rows × 2 columns

Con Pandas podemos crear consultas como esta donde hemos ordenado las jugadoras por potencial de mayor a menor pero esto se puede ver mas visual aun gracias a MatPlotLib

```
In [49]: import matplotlib.pyplot as plt
```

```
In [64]: tencol = df.head(20)
         x = tencol.short_name
         y = tencol.potential
         plt.barh(x, y)
         plt.show()
```

Con MatPlotLib podemos transformar estas consultas en gráficos más visuales por ejemplo creamos un DataFrame de 20 líneas que muestre los datos de antes pero en una barra horizontal.

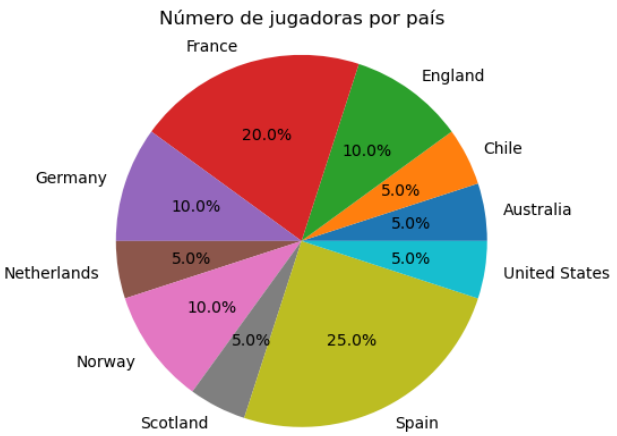


Nos da una tabla ordenada con los nombres de las 20 jugadoras y su barra con su respectivo potencial.

```
In [79]: jugadoras_nacionalidad = tencol.groupby('nationality_name').size().reset_index(name='Jugadoras')
         plt.pie(jugadoras_nacionalidad['Jugadoras'], labels=jugadoras_nacionalidad['nationality_name'],autopct='%1.1f%%')
         plt.axis('equal')
         plt.title('Número de personas por país')

         # Mostrar el gráfico
         plt.show()
```

Usando groupby vamos a agrupar el número de jugadoras por nacionalidad y llamarle a esa columna Jugadoras, posteriormente vamos a convertirlo en un gráfico de tarta colocando Jugadoras para lo que va dentro de los trozos y de labels o etiquetas la nacionalidad, también añadimos que nos de como porcentaje cada trozo.



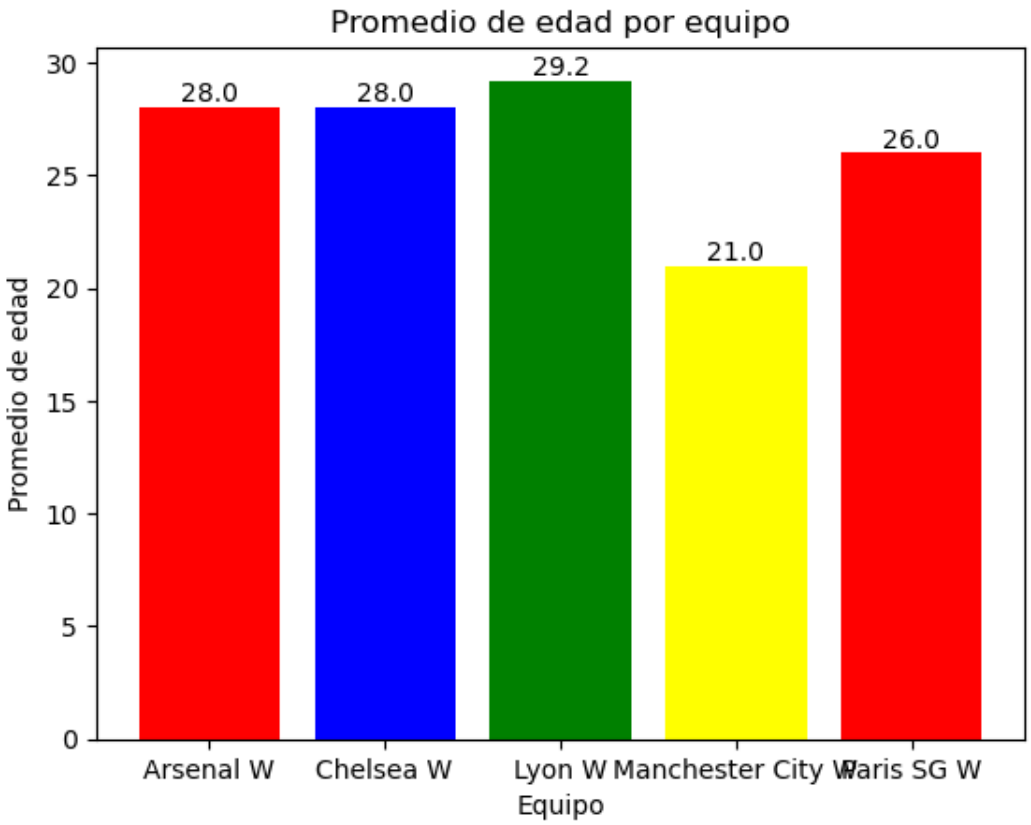
Este gráfico en tarta es muy visual y nos muestra los primeros 20 países y el porcentaje de jugadoras de cada uno.

```
In [91]: tencols = df.head(30)
         colores = ['red', 'blue', 'green', 'yellow']
         promedio_edad = tencols.groupby('club_name').mean().reset_index()
         for i, equipo in enumerate(promedio_edad['club_name']):
             plt.bar(equipo, promedio_edad.loc[i, 'age'], color=colores[i % len(colores)])
             plt.text(equipo, promedio_edad.loc[i, 'age'], round(promedio_edad.loc[i, 'age'], 2), ha='center', va='bottom')


         plt.xlabel('Equipo')
         plt.ylabel('Promedio de edad')
         plt.title('Promedio de edad por equipo')

         # Mostrar el gráfico
         plt.show()
```

Por último vamos a realizar una consulta un poco más visual vamos a pedir que nos de la media de edad de los primeros equipos con unos colores aleatorios elegidos por una lista para ello hacemos la media con .mean() y colocamos los colores en cada barra de la edad.

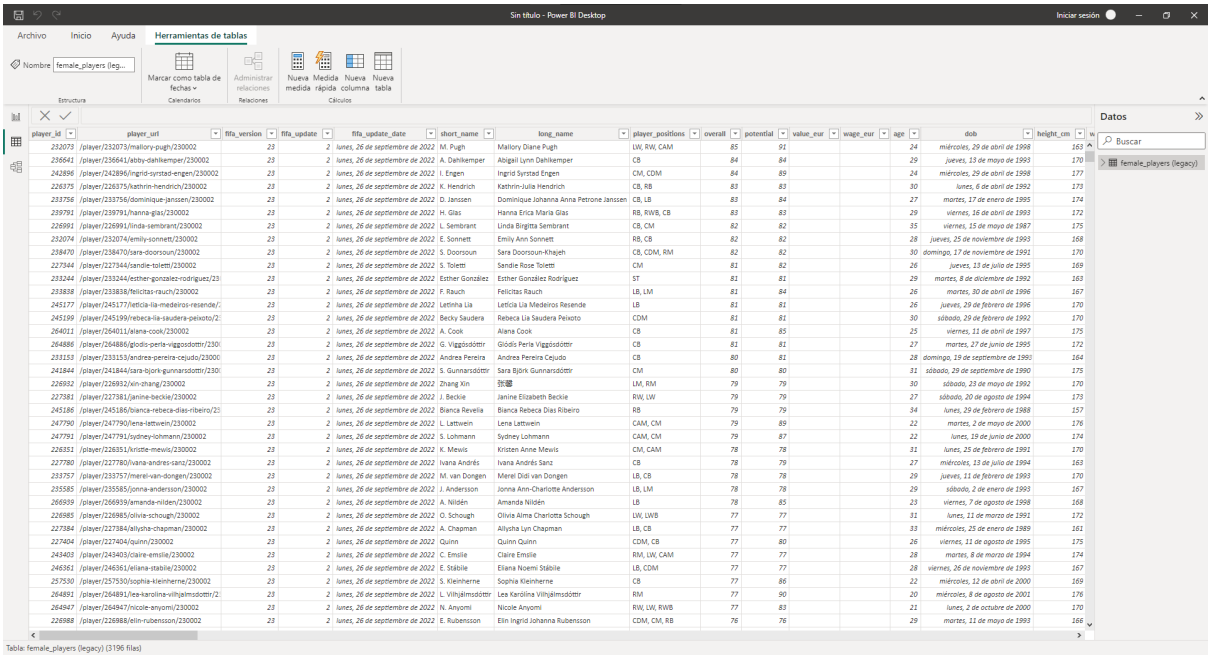


Tras ver estos usos de Pandas junto con MatPlotLib tenemos que dejar claro que este es un uso básico del manejo de datos para manejar datos a grandes escalas tendríamos que usar y es más recomendable usar Spark.

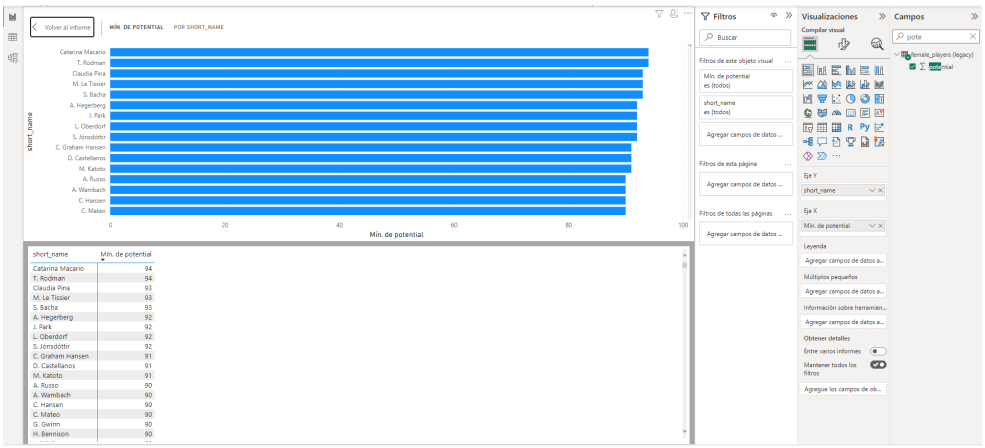
En resumen, Python y Pandas son una combinación de lenguaje de programación y biblioteca de análisis de datos que se enfocan en la simplicidad y la eficiencia en la manipulación de datos, mientras que Scala y Spark son una combinación de lenguaje de programación y marco de trabajo de análisis de datos que se enfocan en la escalabilidad y el procesamiento en paralelo de grandes cantidades de datos.

Tras analizar el manejo de datos con Python vamos a ver cómo se manejan los datos con PowerBi una herramienta mucho más visual y sin necesidad de código para ver gráficos de nuestros csv o base de datos.

Vamos a hacer las mismas consultas pero en PowerBI, primero visualizamos el csv que en PowerBi es muy simple.

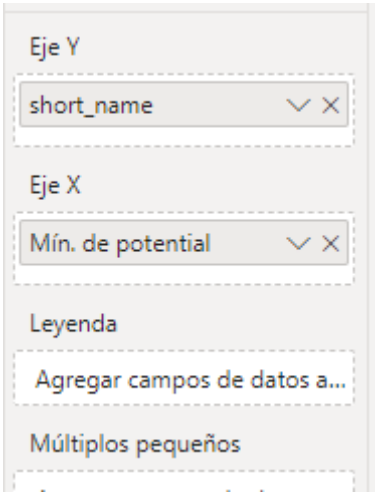


Nos da la información del csv completo. Hacemos la consulta para ver el potencial de las jugadoras ordenado de mayor a menor.



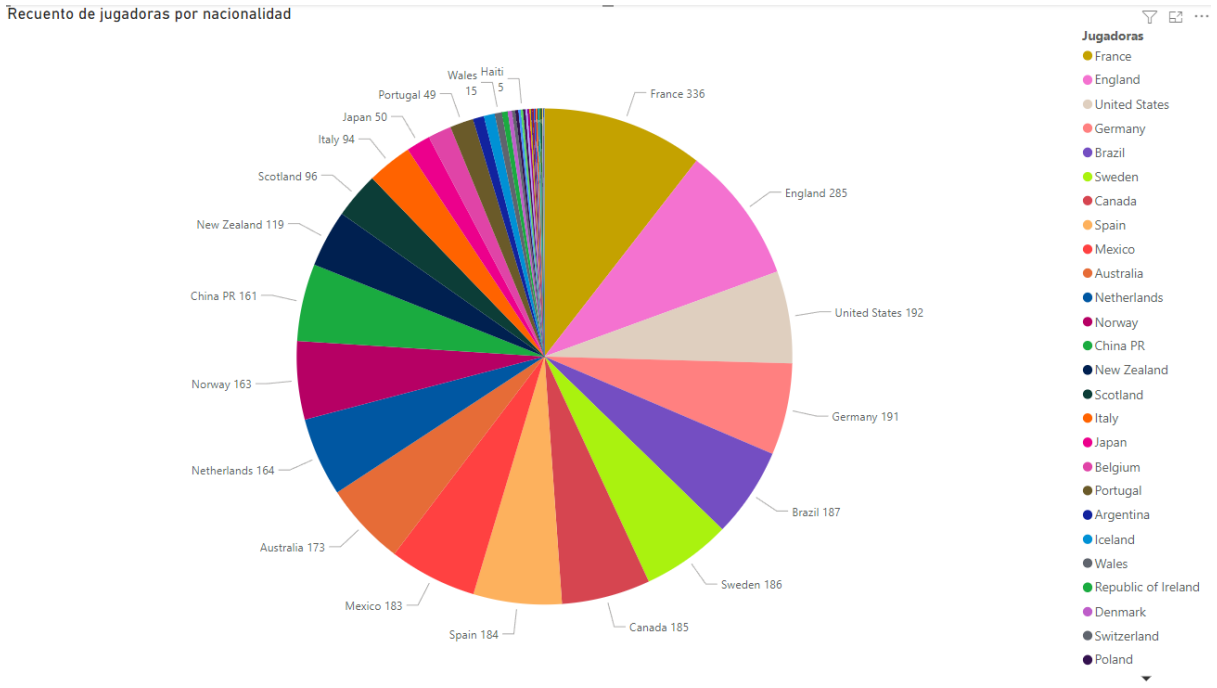
A la derecha tenemos las variables que vamos a usar en el eje x y eje y.

En el panel visual tenemos el gráfico de barras con los nombres de las jugadoras.



Y por último en la parte inferior la tabla con solo los valores elegidos.

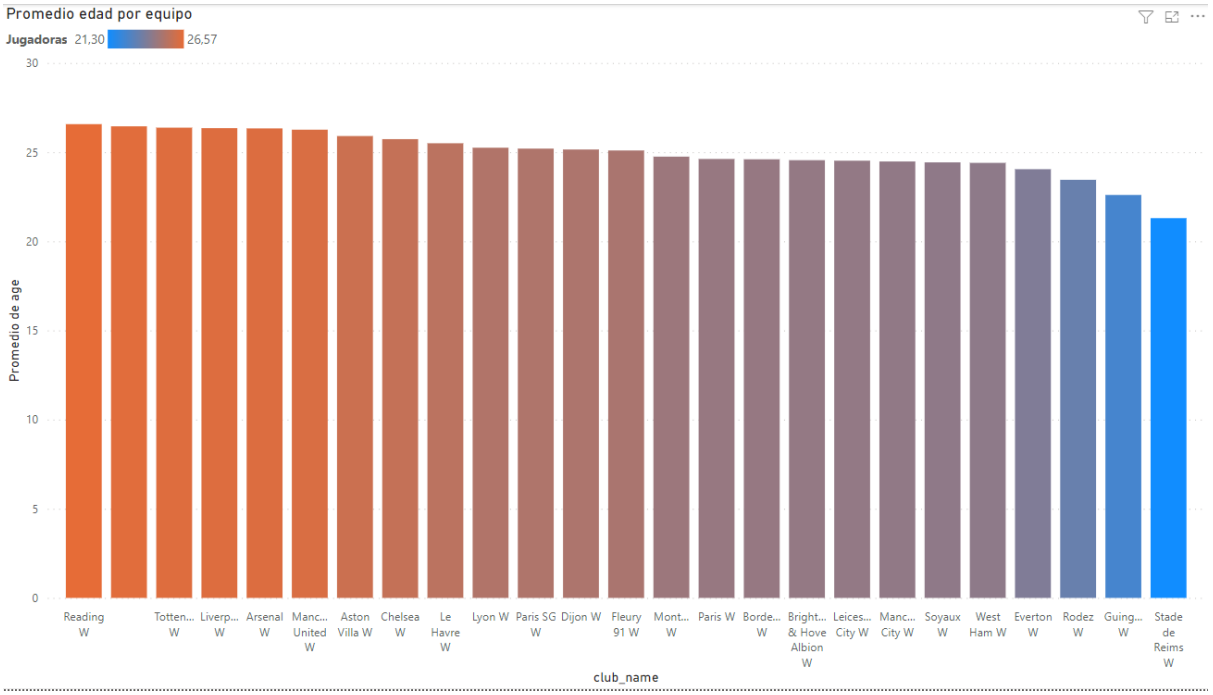
Hacemos la misma consulta de las jugadoras por país solo que en PowerBi nos da mucha más información de manera más visual. Vemos los países, el número de jugadoras y el listado entero.



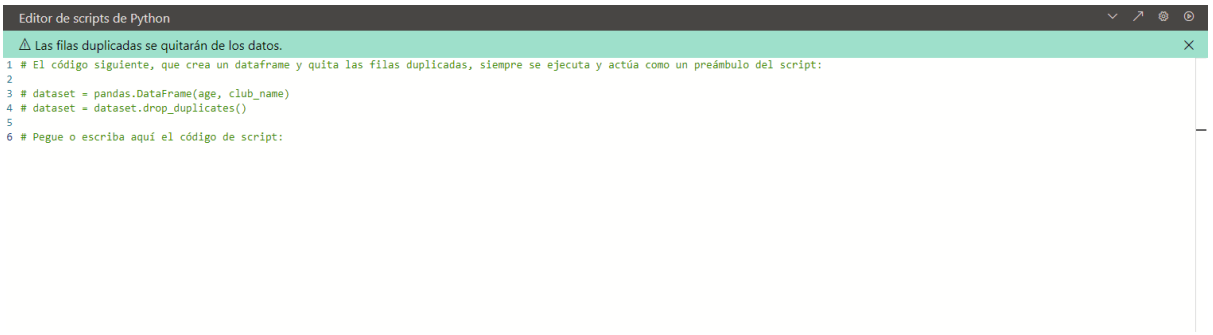
Tenemos también la tabla de los países y número de Jugadoras.

nationality_name	Recuento de short_name
France	336
England	285
United States	192
Germany	191
Brazil	187
Sweden	186
Canada	185
Spain	184
Mexico	183
Australia	173
Netherlands	164
Norway	163
China PR	161
New Zealand	119
Scotland	96
Italy	94
Japan	50
Belgium	49

Ahora haremos la consulta del promedio de edad por equipo.



Nos muestra un gráfico de barras con todos los equipos mucho más visual que en pandas y es más eficiente y rápido, PowerBi aparte de las propias herramientas que tiene, contiene un apartado para añadir código de Python.



En este panel podemos utilizar Pandas y cualquier librería de Python relacionada con análisis de datos.

Análisis de la Fase 2

Pandas y Power BI son dos herramientas muy útiles para el acceso y análisis de datos.

Pandas es una biblioteca de Python que proporciona estructuras de datos y funciones para manipular y analizar grandes conjuntos de datos. Es una herramienta muy popular y ampliamente utilizada en la comunidad de ciencia de datos debido a su facilidad de uso y su capacidad para realizar tareas de manipulación de datos de manera rápida y eficiente. Además, es posible integrar Pandas con otros módulos y bibliotecas de Python para realizar tareas avanzadas de análisis de datos.

Power BI es una plataforma de análisis de datos de Microsoft que permite visualizar y compartir información a partir de diferentes fuentes de datos. Es una herramienta intuitiva y fácil de usar que permite crear visuales de manera rápida y eficiente. Además, es posible integrar Power BI con otras aplicaciones de Microsoft, como Excel, para una experiencia de análisis de datos más completa.

Ambas herramientas tienen sus fortalezas y debilidades. Pandas es excelente para la manipulación y preparación de datos, mientras que Power BI es mejor para la visualización y presentación de resultados. Además, Power BI puede ser más adecuado para usuarios que no tengan experiencia en programación, mientras que Pandas es más adecuado para usuarios con conocimientos de programación y ciencia de datos.

Además de Pandas y Power BI, hay muchas otras herramientas y técnicas de análisis de datos que podrían ser mejores en función de las necesidades específicas de su proyecto. Algunas recomendaciones incluyen:

- Utilizar herramientas de análisis de datos en la nube, como Google BigQuery o Amazon Redshift, que permiten el procesamiento de grandes conjuntos de datos de manera escalable y eficiente en términos de costos.

- Utilizar herramientas de aprendizaje automático, como TensorFlow o PyTorch, para realizar tareas avanzadas de análisis de datos, como la clasificación de datos.

- Utilizar herramientas de visualización de datos avanzadas, como Tableau.

También cabe mencionar Scala que hubiese sido más escalable que python y eficaz.

Acceso a Datos mediante ficheros

En el pasado, el acceso a los datos desde los archivos dependía del sistema operativo y del lenguaje de programación utilizado. En general, se debía abrir el archivo, leer su contenido y luego cerrarlo explícitamente.

En el caso de los sistemas operativos de tipo UNIX, se podía utilizar el comando "cat" para visualizar el contenido de un archivo en la línea de comandos. Para acceder a los datos desde un programa, se podía utilizar la función "fopen" de C para abrir el archivo y luego leer su contenido mediante la función "fread".

En el caso de los sistemas operativos de tipo DOS y Windows, se podía utilizar el comando "type" para visualizar el contenido de un archivo en la línea de comandos. Para acceder a los datos desde un programa, se podía utilizar la función "Open" de Visual Basic para abrir el archivo y luego leer su contenido mediante la función "Input".

En resumen, el acceso a los datos desde los archivos era un proceso relativamente manual que requería un conocimiento detallado de los sistemas operativos y los lenguajes de programación utilizados. Sin embargo, con el tiempo, la mayoría de los lenguajes de programación han proporcionado librerías y herramientas más avanzadas para simplificar este proceso y hacerlo más accesible para los desarrolladores.

Acceso a Datos mediante bases de datos

La definición de bases de datos es «aquel conjunto de datos almacenados y estructurados según sus características o tipología para ser utilizados o consultados posteriormente«.

Hasta hace relativamente pocos años, las bases de datos eran analógicas, es decir, contenían información en papel o textos impresos. Sin embargo, con la llegada de la era digital y el Big Data se ha hecho imprescindible el uso de bases de datos informatizadas.

Su origen se remonta a 1884 con Herman Hollerith, que desarrolló el tabulador electromagnético de tarjetas perforadas con el fin de ayudar en el resumen de información y posteriormente a la contabilidad.

El acceso mismo a la base de datos no la realiza el usuario en sí, sino que es el servicio de información el que efectúa las transacciones comunicándose con el DBMS, y luego entrega al usuario los resultados en forma ordenada.

Acceso a Datos mediante api

API significa “interfaz de programación de aplicaciones”, en pocas palabras la API es la encargada de la comunicación de datos entre aplicaciones ya sea desde una aplicación web a móvil o viceversa.

Las APIs Rest son las más utilizadas ya que estas definen un conjunto de métodos como GET, POST, PUT Y DELETE que se refiere a la lectura, actualización y eliminación de los datos de nuestro cliente al servidor.

Lo primero a realizar será descargar la librería http por ejemplo de Flutter que nos ayudará a consumir los datos de la API. Para hacer uso de esta API tendrás que registrarte y luego suscribirte a esa API:

Una vez suscrito te brindarán un Token con el que se podrá consumir los datos libremente desde la aplicación, ‘X-RapidAPIK-Key’ este parámetro contendrá nuestro token.

Para realizar la consulta a los endpoint hacemos uso de Postman ya que nos ayuda mucho en la visualización de los datos mostrados por la API antes de realizar cambios en nuestra aplicación necesitamos habilitar los permisos de conexión a internet para android, para esto nos iremos al manifest.

Lo siguiente será construir la clase que consumirá los endpoint, para esta clase crearemos variables que contendrán la URL, el host y el token, esto es una buena práctica, ya que si nuestra URL base o token se llegaran a cambiar no será necesario modificar cada porción de código en la parte en que se agregó.

Para mostrar la información del provider necesitamos utilizar Provider.of<>(), este es utilizado para obtener los datos de nuestra dependencia, en nuestro ejemplo sería el método getCarsData.

Y finalmente utilizaremos el widget de Consumer para reconstruir el widget List Cars cada vez que se refleje un cambio en el provider CarProvider. En el Consumer debemos definir el Provider que queremos escuchar.

PodCast

Link Ibox:

Webgrafía

Data lake

- https://keepcoding.io/blog/data-lakes/

Diferencia entre datos estructurados y no estructurados

- https://ayudaleyprotecciondatos.es/bases-de-datos/diferencias-entre-datos-estructurados-y-no-estructurados/#:~:text=Los%20datos%20estructurados%20est%C3%A1n%20altamente,de%20recopilar%2C%20procesar%20y%20analizar.

Hadoop

- https://openwebinars.net/blog/que-es-hadoop/
- https://www.sas.com/es_es/insights/big-data/hadoop.html#:~:text=Hadoop%20es%20una%20estructura%20de,o%20trabajos%20concurrentes%20virtualmente%20ilimitados

Apache Spark

- https://aprenderbigdata.com/introduccion-apache-spark/

Acceso a datos mediante base de datos

- http://www.parada.cl/memoria/doc_3_4.html#:~:text=El%20acceso%20mismo%20a%20la,los%20resultados%20en%20forma%20ordenada.
- https://ayudaleyprotecciondatos.es/bases-de-datos/
- https://click-it.es/breve-historia-del-nacimiento-de-las-bases-de-datos/#:~:text=Pero%20si%20hablamos%20de%20bases,y%20posteriormente%20a%20la%20contabilidad.