Universidad ORT Uruguay

Facultad de Ingeniería

Escuela de Tecnología

Obligatorio

Big Data

Guillermo Tomas – 212655

Juan Barrera – 212114

Santiago González – 283000

Docente

Eduardo García

ATI

Julio 2024

Índice

Caso de estudio

En el presente documento ha seleccionado un conjunto de datos relacionados a la historia de la F1.

Los datos recopilados presenten analizar son los siguientes:

- winners.csv
- fastatest_laps.csv
- teams.csv
- drivers.csv

En base al análisis, ya sea por su contenido y estructura es necesario el planteamiento de preguntas claves para el procesamiento de estas y poder describir lo que representan por medio de gráficas y mapas.

Objetivo

El objetivo general del obligatorio es poder aplicar las herramientas y técnicas aprendidas en Big Data.

Planteamiento del problema

En base al análisis de los datos recopilados es necesario aplicar herramientas aprendidas durante el curso, tales como obtener, manejar, analizar y acceder a grandes volúmenes de datos con muy baja latencia, en primera instancia utilizando un conjunto de datos que tiene **44551** registros y luego realizarse el planteo teórico para escalar el problema para un mayor volumen de datos.

El contenido obligatorio se divide en tres partes, las cuales se describen a continuación:

Parte 1: Análisis, limpieza y presentación de datos en Python usando PANDAS

1. **Ingesta de datos**
    a. Tomar los datos seleccionados junto al docente.
2. **Análisis y limpieza:**
    a. Realizar un análisis utilizando Pandas en un Jupyter Notebook.
    b. Identificar el tipo y estructuras de datos de cada columna y su significado dentro del dominio.
    c. Detectar y limpiar valores nulos o faltantes.
    d. Revisar registros duplicados.
    e. Verificar claves primarias únicas.
3. **Almacenamiento de Archivos:** Guardar los archivos .csv resultantes en una carpeta específica. Estos datos se los consideran depurados para poder procesarlos para su futura manipulación.
4. **Visualizaciones:**

a. Crear visualizaciones en otro notebook utilizando herramientas dadas en clase u otras de elección del equipo.
b. Las visualizaciones deben responder las preguntas seleccionadas.

Parte 2: Análisis, limpieza y presentación de datos en Python usando SPARK

1. **Ingesta de datos**
    a. Uso de Hadoop Distributed File System (HDFS) e ingreso de datos.
2. **Análisis y limpieza**
    a. Se realiza manipulan lo datos de similar manera que la parte 1.
3. **Almacenamiento de Archivos:** Guardar los archivos .csv resultantes en una carpeta específica en el HDFS. Estos datos se los consideran depurados para poder procesarlos para su futura manipulación.
4. **Visualizaciones:** Se aplican las mismas herramientas de representación igual que en la parte 1.Solo se han aplicado sobre algunas de las preguntas definidas.

Parte 3: Desarrollo de un Dashboard

1. **Implementación:**
    a. Desarrollar un dashboard para responder las preguntas planteadas y generación de datos específicos para implementar.
    b. Implementarlo en Tableau Public.

Parte 4: Modelado de Datos

1. **Elección del Modelo:** Elegir una forma de modelar los datos: Normalizada, Diagrama Estrella, Data Vault, o OBT.
2. **Descripción en Hive:**
    a. Describir cómo se modelarían los datos en Hive.
    b. Indicar qué tablas se crearían y su tipo (externas, internas).

Preguntas formuladas

En base a los datos a analizar se realiza la formulación de las siguientes preguntas a responder:

- ¿Cuáles son los mejores tiempos promedio por circuito?
- ¿Cuáles son los autos con los mejores tiempos?
- ¿Cuáles son los autos con más carreras ganadas?
- ¿Cuál es el promedio de mejores tiempos de pilotos ganadores por año?
- ¿Cuáles son los equipos(teams) con más premios en los últimos 5 años? ¿Y en los últimos 10?
- ¿Quiénes son los pilotos con mayor desempeño en los circuitos, basado en sus tiempos?
- ¿Cómo se pueden listar los circuitos y los pilotos en un mapa?
    - Circuitos: Representación con puntos
    - Pilotos: Representación con mapa de calor de los países que tienen pilotos que participan y su cantidad.

- ¿Cuáles son las escuderías que ganan con mayor frecuencia en carreras en Europa? ¿Cuáles son sus puntajes máximos?
- ¿Quiénes son los 10 pilotos con más carreras ganadas? Dado este Top 10, ¿Cuáles son los 3 circuitos que más dominan?
- ¿Como es la evolución de victorias de los 5 equipos más ganadores de los últimos 10 años?

Hay un total de 10 preguntas, algunas de ellas tienen cierto grado de complejidad y otras en menor medida

Análisis

En la siguiente sección se realiza la descripción de los datos a procesar en forma cruda.

Análisis de archivos en Pandas

Los archivos crudos, o raw, se guardan en la carpeta input_data. Aquí se describe nuestra interpretación de cada uno.

Ganadores (winners.csv)

Detalles del archivo:

- Columnas separadas por " ,"
- **Grand Prix:** Refiere al nombre del circuito (Tipo de dato: object)
- **Date:** Fecha de la carrera yyyy-mm-dd (Tipo de dato: object)
- **Winner:** Es el Nombre completo del piloto que ganó (Tipo de dato: object)
- **Car:** Es el equipo de F1o vehículo que compite (Tipo de dato: object)
- **Laps:** Número de vueltas (Tipo de dato: float64)
- **Time:** Es el tiempo durante la carrera 00:00:00.000 (Tipo de dato: object)
- **Name Code:** Código que identifica a un piloto tomando las tres primeras letras de su apellido. (Tipo de dato: object)
- Presenta 7 columnas y 1107 filas

Vueltas más rápidas (fastest_laps.csv)

Detalles del archivo:

- Columnas separadas por ","
- **Grand Prix:** Refiere al nombre del circuito (Tipo de dato: object)
- **Driver:** Es el Nombre completo del piloto (Tipo de dato: object)
- **Car:** Es el equipo de F1o vehículo que compite (Tipo de dato: object)
- **Time:** Es el tiempo de una vuelta en el circuito 00:00.000 (Tipo de dato: object)
- **Name Code:** Código que identifica a un piloto tomando las tres primeras letras de su apellido. (Tipo de dato: object)
- Presenta 5 columnas y 1105 filas.

Equipos o Escuderías (teams.csv)

Detalles del archivo:

- Columnas separadas por “ ,”
- **Pos :** Refiere a la posición del equipo en el ranking de F1 (Tipo de dato: object)
- **Team:** Es el Nombre del equipo (Tipo de dato: object)
- **PTS:** Los puntos que acumula en el ranking (Tipo de dato: float64)
- Presenta 3 columnas y 684 filas.

Pilotos (dirvers.csv)

Detalles del archivo:

- Columnas separadas por “ ,”
- **Pos:** Refiere a la posición del equipo en el ranking de F1 (Tipo de dato: object)
- **Driver:** Es el Nombre completo del piloto (Tipo de dato: object)
- **Nationality:** Nacionalidad del piloto en código ISO 3166-1 alfa -3 (Tipo de dato: object)
- **Car:** Es el equipo de F1o vehículo que compite (Tipo de dato: object)
- **PTS:** Los puntos que acumula en el ranking (Tipo de dato: float64)
- **Name Code:** Código que identifica a un piloto tomando las tres primeras letras de su apellido. (Tipo de dato: object)
- Presenta 6 columnas y 1655 filas.

Datos auxiliares

Para complementar los datos existentes, se acordó agregar otros datos adicionales para la etapa de procesamiento y representación. En esta sección se describe lo incorporado.

Ubicaciones de los circuitos (grand_prix_locations.csv)

Detalles del archivo:

- Columnas separadas por “,”
- **Grand Prix:** Refiere al nombre del circuito (Tipo de dato: object)
- **Country Code:** Código del país en ISO 3166-1 alfa -3 del país donde se ubica el circuito (Tipo de dato: object)
- **Latitude:** Latitud de la ubicación del circuito (Tipo de dato: float64)
- **Longitude:** Longitud de la ubicación del circuito (Tipo de dato: float64)
- Este archivo se unirá con fastest_laps.csv y winners.csv para poder disponer de la ubicación de geográfica de los circuitos.
- Este archivo se crea especialmente para resolver la representación en un mapa.

Ubicación de los países (country_locations.csv)

Detalles del archivo:

- Columnas separadas por “ ,”
- **Nationality:** Código del país en ISO 3166-1 alfa -3, se mantiene el mismo nombre de columna (Tipo de dato: object)
- **Latitude:** Latitud de la ubicación del país (Tipo de dato: float64)
- **Longitude:** Longitud de la ubicación del país (Tipo de dato: float64)
- Este archivo se unirá con drivers.csv para poder disponer de la ubicación de geográfica de los pilotos.
- Este archivo se crea especialmente para resolver la representación en un mapa.

Limpieza de archivos en Pandas

Los archivos limpios quedan guardados en la carpeta output_data.

winners.csv

Datos nulos:

- Columna Time tiene 3 datos nulos, Se eliminaron los registros que los contenían
- Columna Laps tiene 3 datos nulos, Se eliminaron los registros que los contenían

Datos duplicados: No tiene datos duplicados

fastest_laps.csv

Datos nulos:

- Columna Time tiene 1 dato nulo, se eliminó el registro que lo contenía

Datos duplicados: No tiene datos duplicados

teams.csv

Datos nulos: No tiene datos nulos

Datos duplicados: No tiene datos duplicados

dirvers.csv

Datos nulos:

- Columna Car tiene 11 datos nulos, se asignó el valor “GENERIC” por defecto.

Datos duplicados: Tiene 6 duplicados, se eliminaron.

Datos modificados: Para poder usar algunas de las herramientas fue necesario modificar los códigos de los países según la ISO 3166-1 alfa -3, se aplicaron sobre los siguientes registros:

- Se modifica aquellos registros 'Nationality' con valor 'GER' pasan a ser 'DEU'
- Se modifica aquellos registros (Nationality) con valor 'RSA' pasan a ser 'ZAF'
- Se modifica aquellos registros (Nationality) con valor 'RHO' pasan a ser 'ZWE'
- Se modifica aquellos registros (Nationality) con valor 'RAF' pasan a ser 'RUS

Comentarios:

- Se realiza la verificación de los datos luego de aplicar la limpieza para comprobar que realmente no contengan nulos ni valores duplicados.
- Los criterios aplicados fueron de consenso general del grupo.

Almacenamiento

Se define ordenar los archivos ya procesado para su manipulación guardarlos en una carpeta especifica con la denominación **output_data**, donde estos luego podrán usarse para crear las visualizaciones correspondientes a las preguntas, tales como gráficos de puntos de eje simple, doble eje, evolutivos y representación geográfica.

Los archivos almacenados estarán en formato CSV separados por “,”, son los siguientes:

- drivers_clean
- fastest_laps_clean
- teams_clean
- winners_clean

Visualizaciones

En base a las preguntas planteadas se decide realizar la representación utilizando gráficas y mapas, para esto fue necesario el uso de matplotlib , folium y pydeck.

Matplotlib es una biblioteca de Python utilizada para crear visualizaciones estáticas, animadas e interactivas en forma de gráficos. En nuestro caso, se han utilizado varios tipos de gráficos:

- **Gráficos de puntos (scatter plots):** Se usan para representar la relación entre dos variables. Cada punto en el gráfico representa una observación en el conjunto de datos.
- **Gráficos de doble eje:** Permiten mostrar dos conjuntos de datos diferentes con escalas distintas en el mismo gráfico, lo que facilita la comparación entre dos variables que tienen diferentes unidades de medida.
- **Gráficos evolutivos (line plots):** Se usan para mostrar cómo cambia una variable a lo largo del tiempo, representando tendencias y patrones evolutivos en los datos.

Folium es una biblioteca de Python que se utiliza para crear mapas interactivos. En nuestro caso, se ha empleado un mapa de calor (heatmap), que es una forma de visualizar la densidad de puntos en un área geográfica. Los mapas de calor son útiles para identificar áreas de alta concentración de datos sobre un mapa.

Pydeck es una herramienta de visualización geoespacial de alto nivel que permite crear mapas interactivos en 2D y 3D. En tu contexto, se ha usado un mapa geográfico para ubicar puntos específicos en el mapa en base a la ubicación de Latitud y Longitud.

Para utilizar cualquiera de las herramientas mencionadas fue necesario dar formato algunos valores de las columnas de los Dataframe en referencia al tiempo y fecha.

Para las representaciones elegidas que se utilizarán en Tableau Public, se decidió generar archivos específicos en formato CSV. Estos archivos se almacenan en la carpeta **output_data/tableau_public** e incluyen:

- Circuitos
- Historico_Victorias
- Pilotos
- Equipos_Victorias_MaxPTS

Trabajar con los datos ya procesados en Tableau Public agiliza y facilita su uso, permitiendo centrarse en la representación de los datos en lugar de en la lógica para manipularlos.

Spark

Para manejar grandes volúmenes de datos, es crucial optar por herramientas que ofrezcan eficiencia y escalabilidad. En este contexto, Apache Spark se destaca sobre pandas debido a su capacidad para procesar datos distribuidos en clústeres de computadoras. Mientras que pandas es excelente para análisis de datos en memoria y funciona bien con conjuntos de datos pequeños a medianos, su rendimiento se degrada considerablemente cuando se trata de volúmenes de datos que superan la capacidad de la memoria de una sola máquina. Spark, en cambio, utiliza un modelo de procesamiento distribuido que permite manejar terabytes y petabytes de datos sin problemas. Esto se logra a través de su arquitectura basada en Resilient Distributed Datasets (RDDs), que distribuye el procesamiento de datos en múltiples nodos, lo que no solo acelera las operaciones, sino que también ofrece tolerancia a fallos y recuperación automática.

Otra ventaja significativa de utilizar Spark es su integración con Hadoop Distributed File System (HDFS). HDFS proporciona un sistema de almacenamiento distribuido que permite a Spark acceder y procesar grandes conjuntos de datos de manera eficiente. La configuración del HDFS implica la distribución de datos en bloques a través de varios nodos en un clúster, lo que garantiza alta disponibilidad y redundancia. Además, Spark es altamente escalable; se pueden añadir más nodos al clúster para aumentar la capacidad de procesamiento y almacenamiento sin necesidad de reconfigurar la arquitectura de la aplicación. Esta capacidad de escalar horizontalmente convierte a Spark en una herramienta ideal para entornos de big data, donde la cantidad de datos y la demanda de procesamiento pueden crecer exponencialmente. En resumen, para manejar grandes cantidades de datos, Spark ofrece ventajas claras en términos de rendimiento, escalabilidad y robustez en comparación con pandas.

Configuración

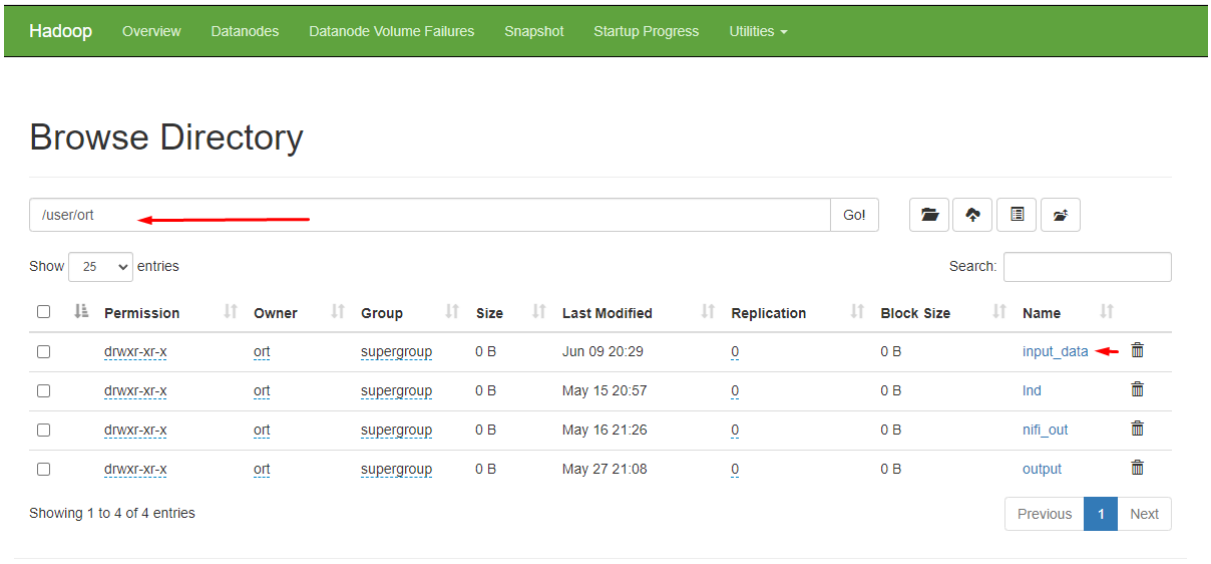
Para el uso de Spark por medio de un entorno de desarrollo Jupyter Notebook es necesario que se carguen los datos en el HDFS para que se puedan manipular en el mismo entorno para su etapa de procesamiento.

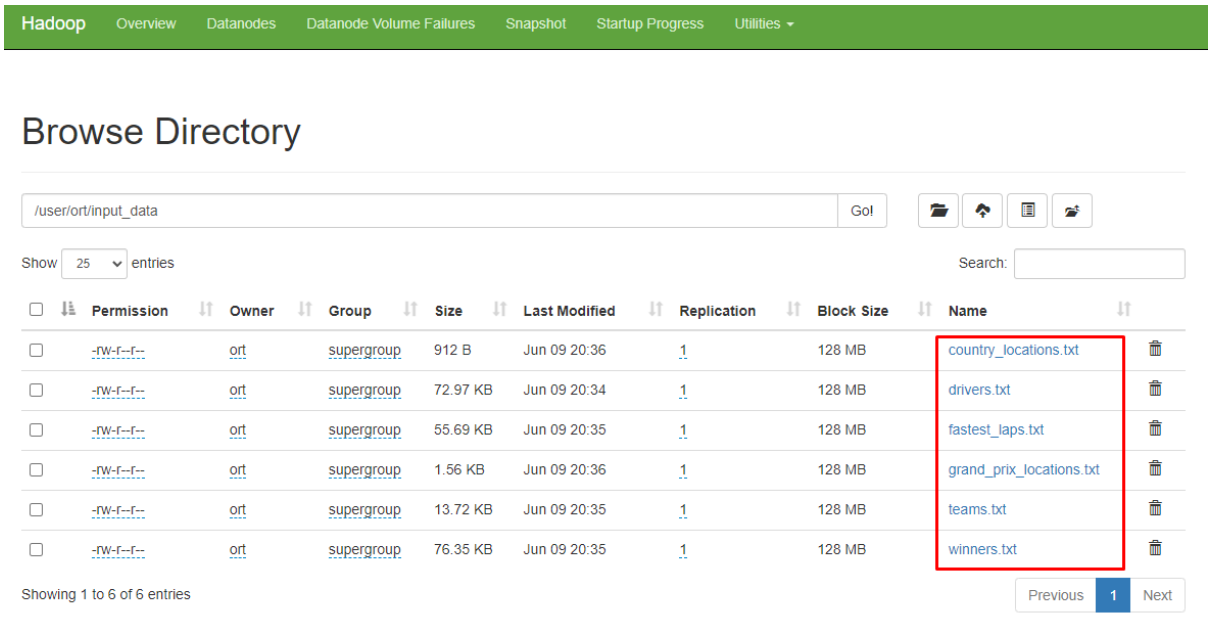
Hadoop Distributed File System (HDFS)

Se detallan los pasos realizados para poder agregar datos el HDFS para luego poder trabajar en Spark en Pyhton.

Acceso:

- Habilitación de la máquina virtual Azure y acceso por medio de la consola https://portal.azure.com/#cloudshell. Datos para el ingreso:
    - SSH ort@74.235.90.139
    - Password: tallerORT1!!
- Creación de las carpetas para el conjunto de datos a procesar.
    - **Input_data_temp :** Se agregan a la máquina virtual todos los datos a procesar y los auxiliares. Aplicar el comando mkdir Input_data_temp
- Agregar los archivos a procesar en la carpeta creada en el paso anterior:
    - vi drivers.txt, copiar los datos manipulados. Insertar y grabar los datos
    - vi fastest_laps.txt, copiar los datos manipulados. Insertar y grabar los datos
    - vi teams.txt, copiar los datos manipulados. Insertar y grabar los datos.
    - vi winners.txt, copiar los datos manipulados. Insertar y grabar los datos.
    - vi grand_prix_locations.txt, copiar los datos manipulados. Insertar y grabar los datos.
    - vi country_locations.txt, copiar los datos manipulados. Insertar y grabar los datos.
    - **Comentarios:** La ingesta de datos para el HDFS es manual, desde la máquina virtual al clúster.
- Verificación de las carpetas actuales de HDFS, aplicando el comando hdfs dfs -ls.
- Creación de una nueva carpeta para realizar los trabajos requeridos para el obligatorio hdfs dfs -mkdir user/ort/input_data.
- Copiar desde la máquina virtual al HDFS todos los archivos ubicados en Input_data_temp a la carpeta input_data.
    - [ort@taller-bigdata-vm Input_data_temp]$ hdfs dfs -put drivers.txt /user/ort/input_data/
    - [ort@taller-bigdata-vm Input_data_temp]$ hdfs dfs -put fastest_laps.txt /user/ort/input_data/
    - [ort@taller-bigdata-vm Input_data_temp]$ hdfs dfs -put teams.txt /user/ort/input_data/
    - [ort@taller-bigdata-vm Input_data_temp]$ hdfs dfs -put winners.txt /user/ort/input_data/
    - [ort@taller-bigdata-vm Input_data_temp]$ hdfs dfs -put grand_prix_locations.txt /user/ort/input_data/
    - [ort@taller-bigdata-vm Input_data_temp]$ hdfs dfs -put country_locations.txt /user/ort/input_data/
- Verificar desde la url : http://hdfs.bigdata.ort.edu.uy/ si correctamente se han cargado los archivos en referencia:



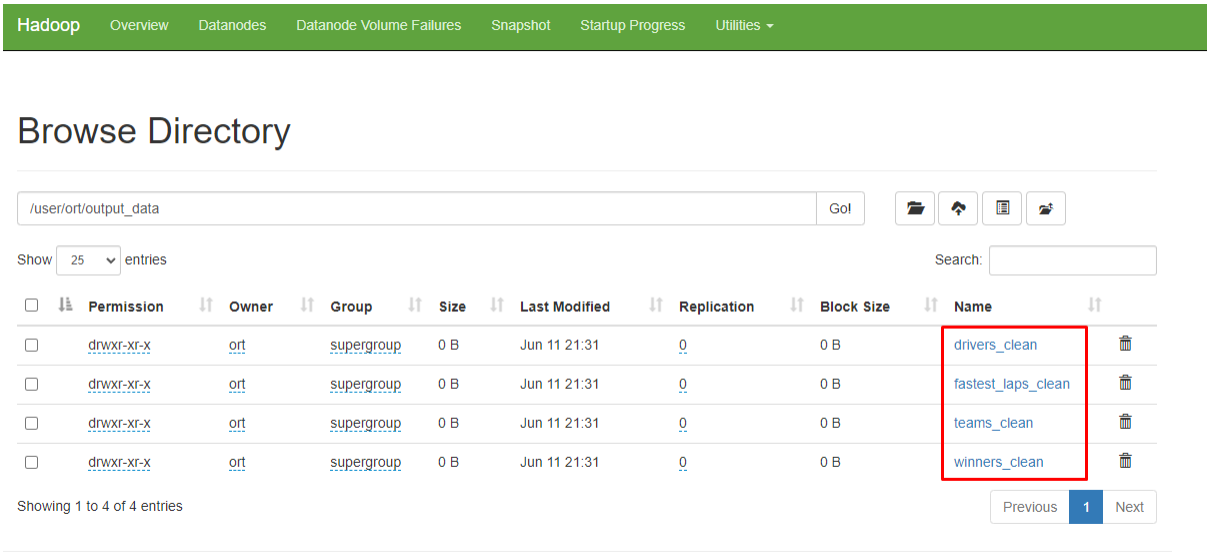


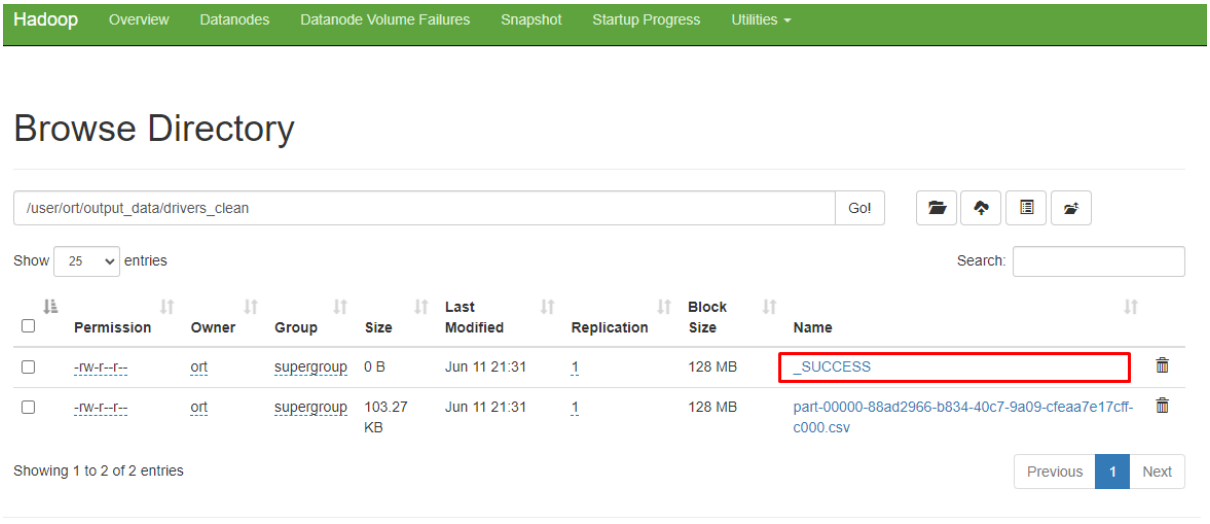
Entorno de desarrollo Jupyter Notebook

Se describen los pasos realizados para llevar a cabo las consultas y acciones para manipular los datos.

- Una vez agregado los archivos a procesar en el ambiente del HDFS , entramos al entorno de desarrollo Jupyter Notebook en la siguiente dirección http://jupyter.bigdata.ort.edu.uy . Se podrá visualizar adjunto a este documento el archivo.
- Crear un archivo para utilizar y realizar las consultar correspondientes usando SPARK , nombre: Part2_Limpieza_y_analisis.
- Se verifica que la lectura de los archivos sea efectiva.
- Previo almacenar los archivos analizar se genera en el HDFS una carpeta para el guardado:
    - Aplicar el comando hdfs dfs -mkdir /user/ort/output_data

- Se realiza el análisis de los datos desde Spark haciendo analogía con lo hecho en Pandas, se enumera lo realizado :
    - Columnas presentes por cada Data Set
    - Tipos de datos
    - Análisis de la cantidad de Nulos y Duplicados en los datos
    - Cantidad de filas y columnas
    - Validar datos si efectivamente no hay nulos y/o duplicados.
    - Realizar el JOIN entre los datos analizar para validarlos se los muestra en pantalla.
    - Se procese a guardar los datos en el HDFS con similar nombre en la etapa anterior en Pandas.
- Para ajustar el guardado, es necesario saber con certeza la ubicación del HDFS , para eso se acciona el comando [ort@taller-bigdata-vm ~]$ hdfs getconf -confKey fs.defaultFS . Resultado : hdfs://localhost:9000
- Verificación del correcto guardado de archivos en HDFS en el navegador de archivos





- En una segunda etapa se crea un nuevo archivo para realizar el análisis y presentación de datos Part 2 Análisis y presentación de datos. Se procese a realizar todas consultas realizadas en cada uno de los archivos a trabajar igual a los realizado por Pandas.

Análisis de archivos en Spark

Ídem al Análisis de archivos en Pandas, tiene similar comportamiento al Dataframe y funciones similares.

Limpieza de archivos en Spark

Ídem al Limpieza de archivos en Pandas, tiene similar comportamiento al Dataframe y funciones similares.

Visualización de archivos en Spark

Una vez realizado el análisis previo, es posible convertir los Dataframes resultantes del análisis a formato Pandas utilizando la función ToPandas() de Spark, para poder representar gráficamente los resultados.

Gráficos

En base a las preguntas planteadas se enumeran cada una de ellas y se realiza un desarrollo breve como se realizó la consulta y cuál fue el resultado.

1. ¿Cuáles son los autos con más premios en los últimos 10 años?

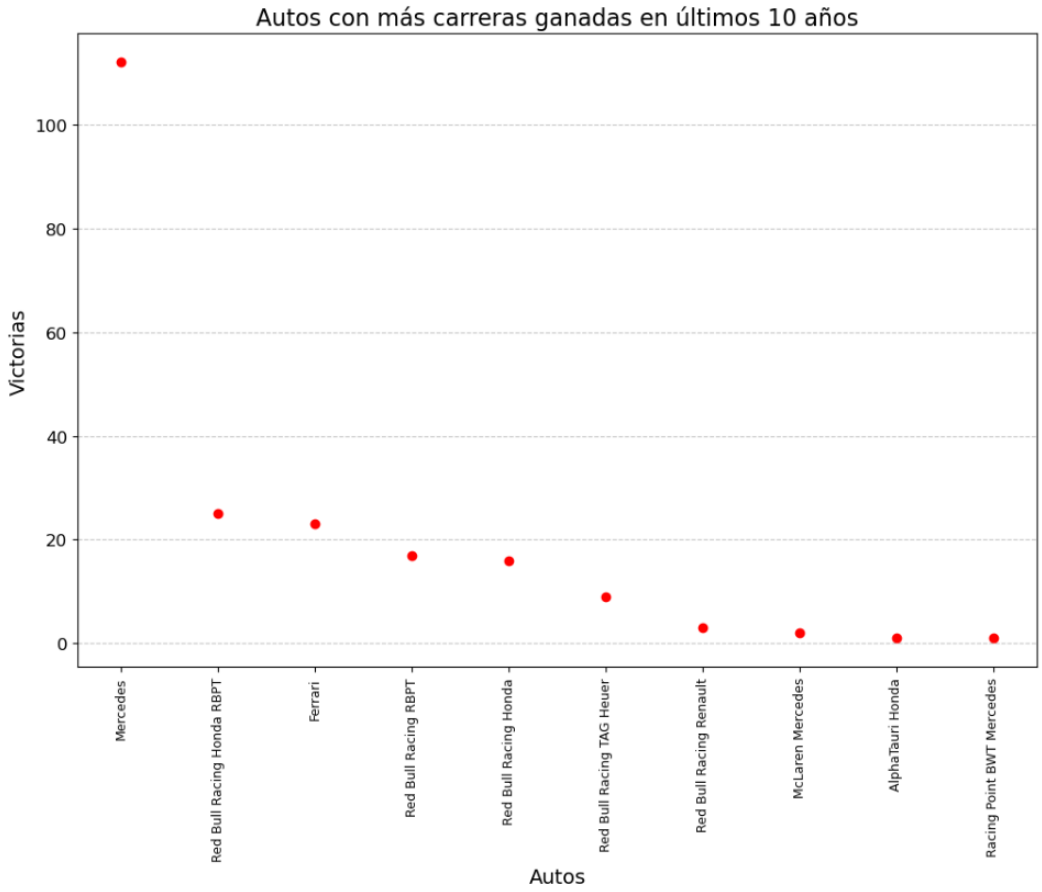
Para esta pregunta utilizamos el archivo winners_clean.csv.

En este caso se realizó una copia del dataframe de winners sobre el que se agregaron dos columnas Date y Year, sobre Date se asignó el formato DateTime y en Year se guarda la componente del año de la columna Date.

Luego se seleccionan los registros que cumplan que el año (Columna Year) este entre el actual y 10 años atrás.

Finalmente se cuentan las apariciones de los autos (columna Car).

Resultado

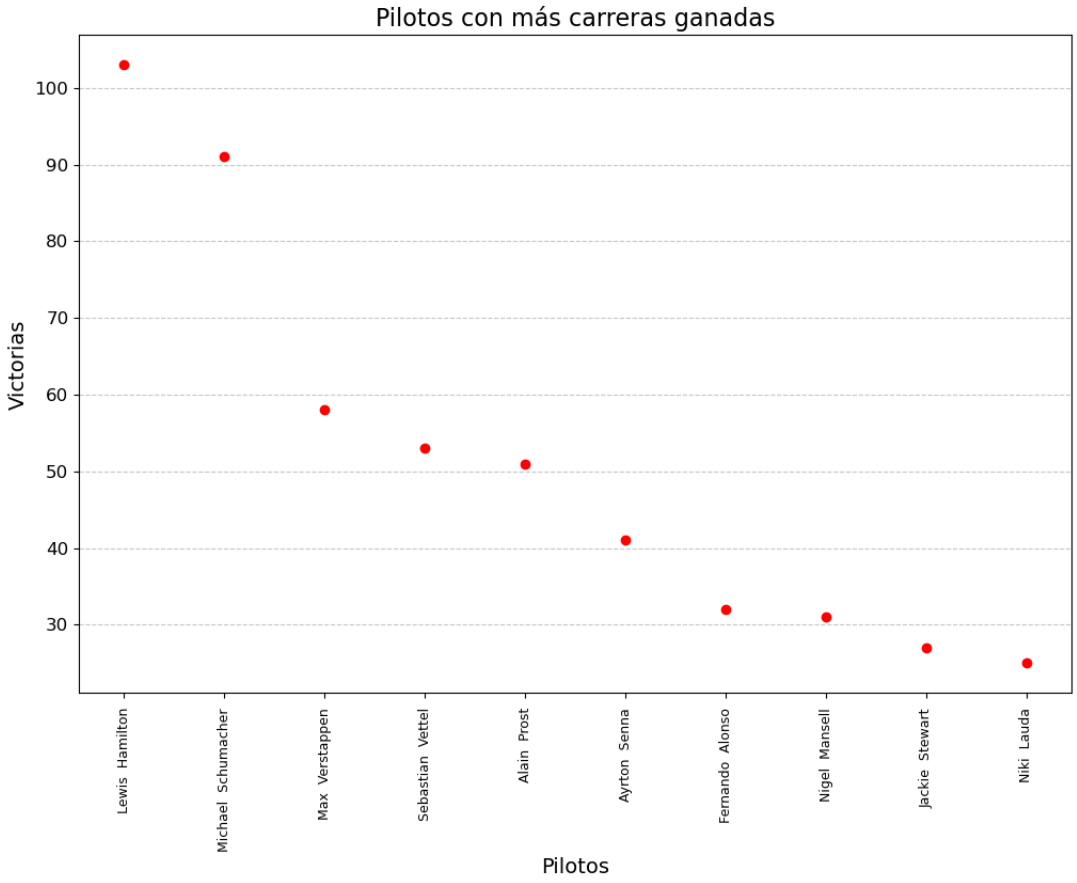


2. ¿Quiénes son los 10 pilotos con más carreras ganadas? Dado este Top 10, ¿cuáles son los 3 circuitos que más dominan?

Para esta pregunta utilizamos el archivo winners_clean.csv.

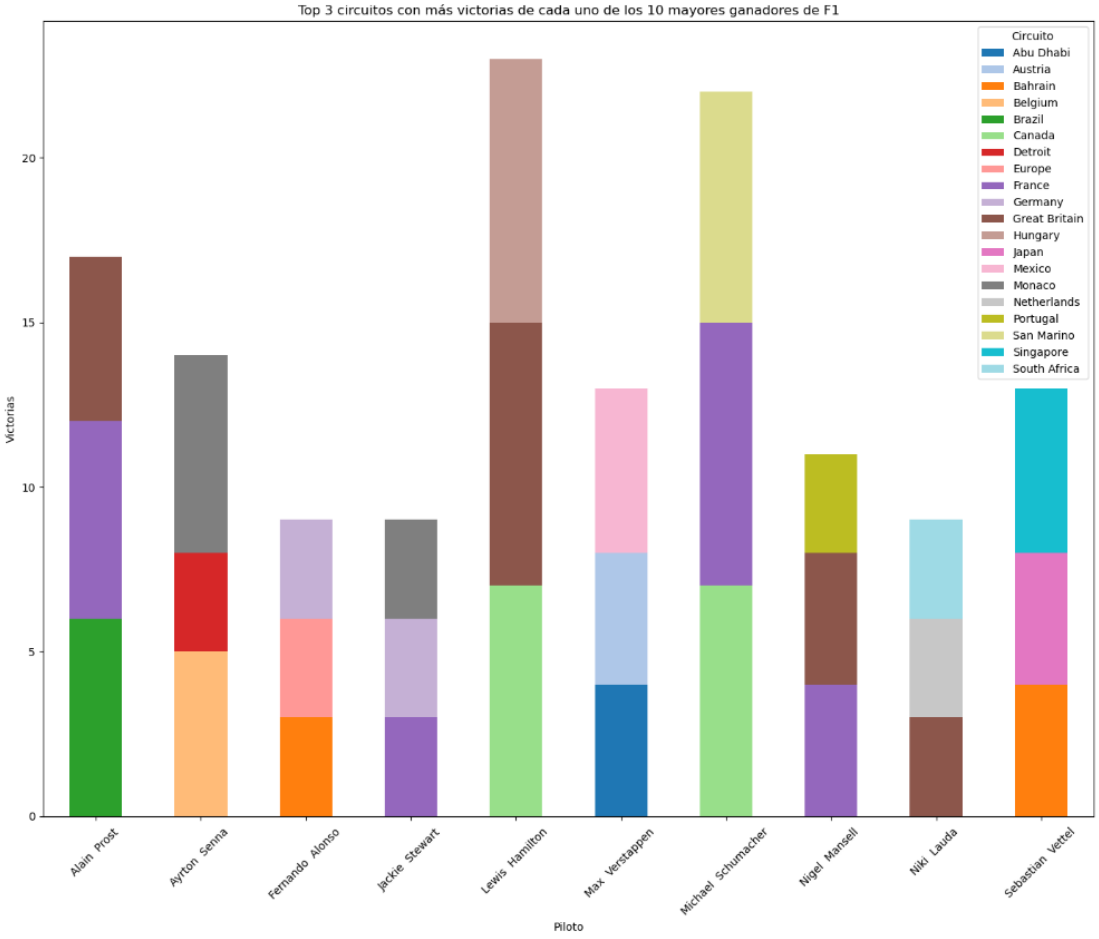
Para la primera consulta se obtienen las cantidades de victorias por piloto (Columna Winner) y se toman los primeros 10.

Resultado



Para la segunda consulta se agrupa por piloto y circuito (columnas Winner y Grand Prix) y se cuentan las victorias por piloto por circuito. Finalmente se agrupa por piloto (Columna Winner) y se toman los primeros 3 registros para cada circuito.

Resultado



3. ¿Cuáles son las escuderías que ganan con mayor frecuencia en carreras en Europa? ¿Cuáles son sus puntajes máximos?

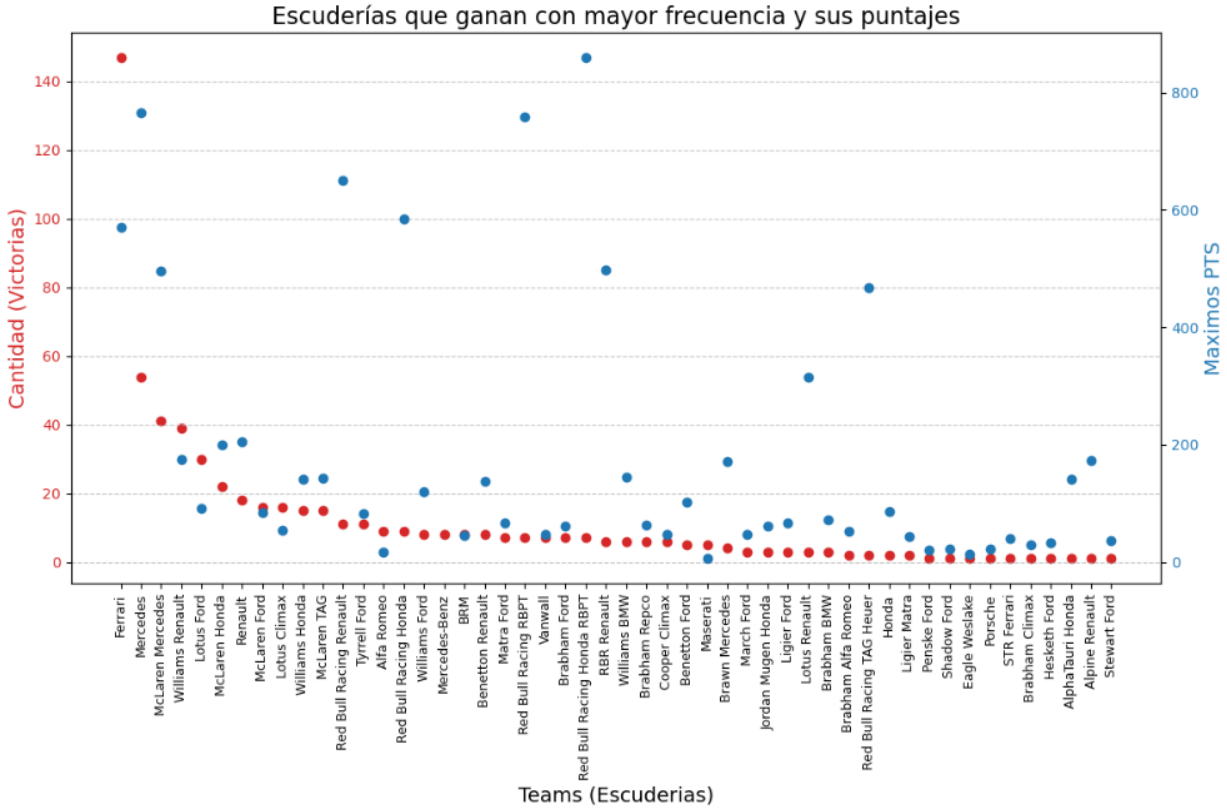
Para esta pregunta utilizamos el archivo winners_clean.csv y teams_clean.csv.

Primero se definen los códigos de países de Europa en una lista y luego se filtra un dataframe para obtener los registros de carrearas ganadas en Europa.

Se cuentan las victorias por cada equipo o team (Columna Car) y se ordena de manera descendente.

Luego se seleccionan los equipos, pero buscándolos en el dataframe de teams para así agruparlos y obtener los puntajes máximos.

Resultado

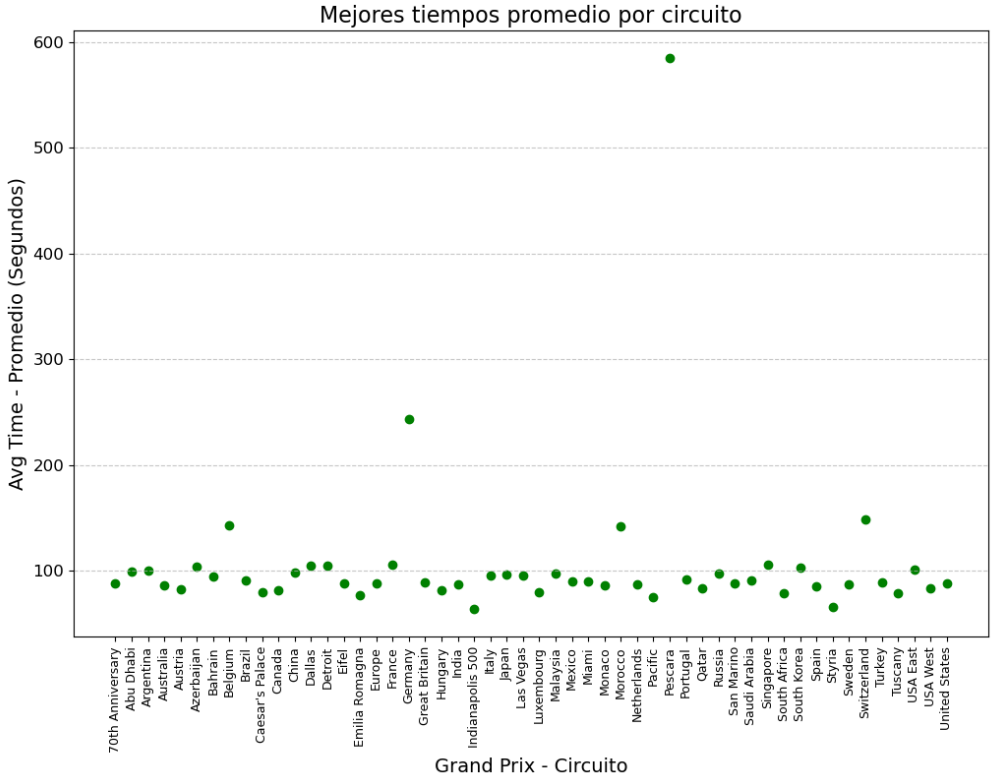


4. ¿Cuáles son los mejores tiempos promedio por circuito?

Para esta pregunta se utilizó el archivo fastest_laps_clean.csv.

Para realizar esta consulta se calcula el promedio de tiempo de vueltas por cada circuito, y se grafica en orden ascendente.

Resultado

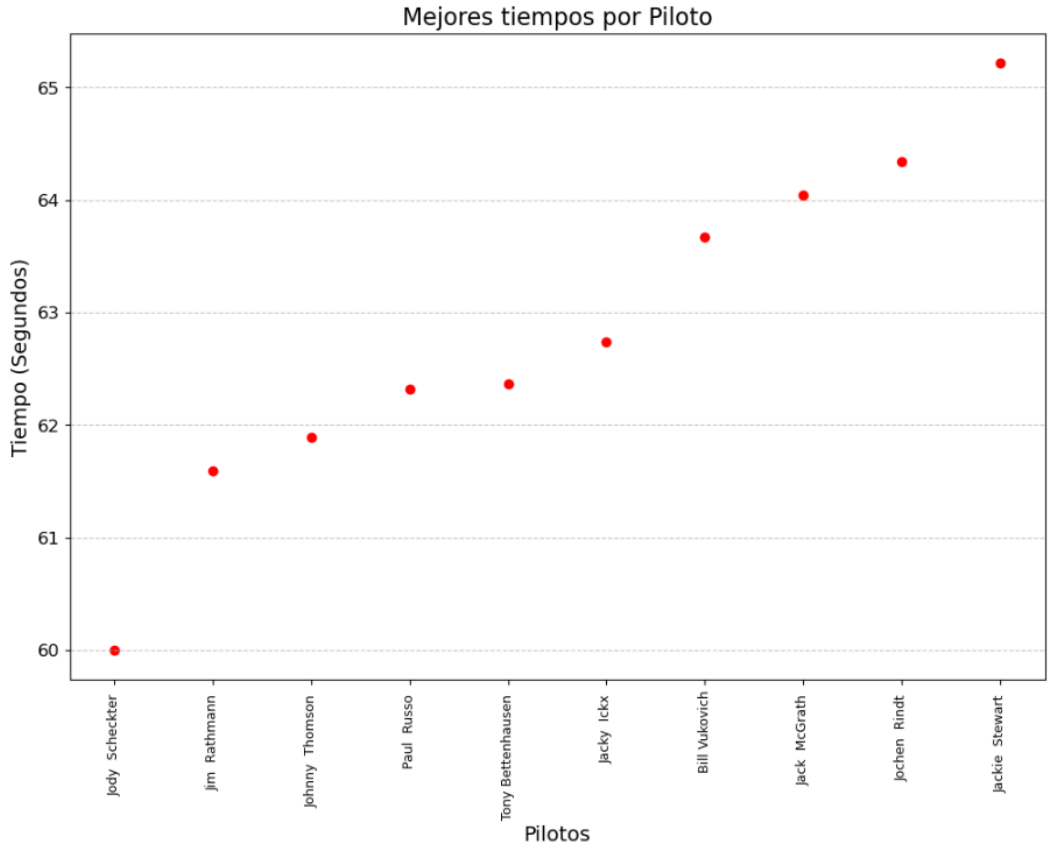


5. ¿Quiénes son los pilotos con mayor desempeño en los circuitos, basado en sus tiempos?

Para esta pregunta utilizamos el archivo fastest_laps_clean.csv.

Se agrupa por piloto (Columna Driver) seleccionando los mínimos sobre el tiempo en segundos (Columna Time_sec)

Resultado



6. ¿Cómo se pueden listar los circuitos y los pilotos en un mapa?

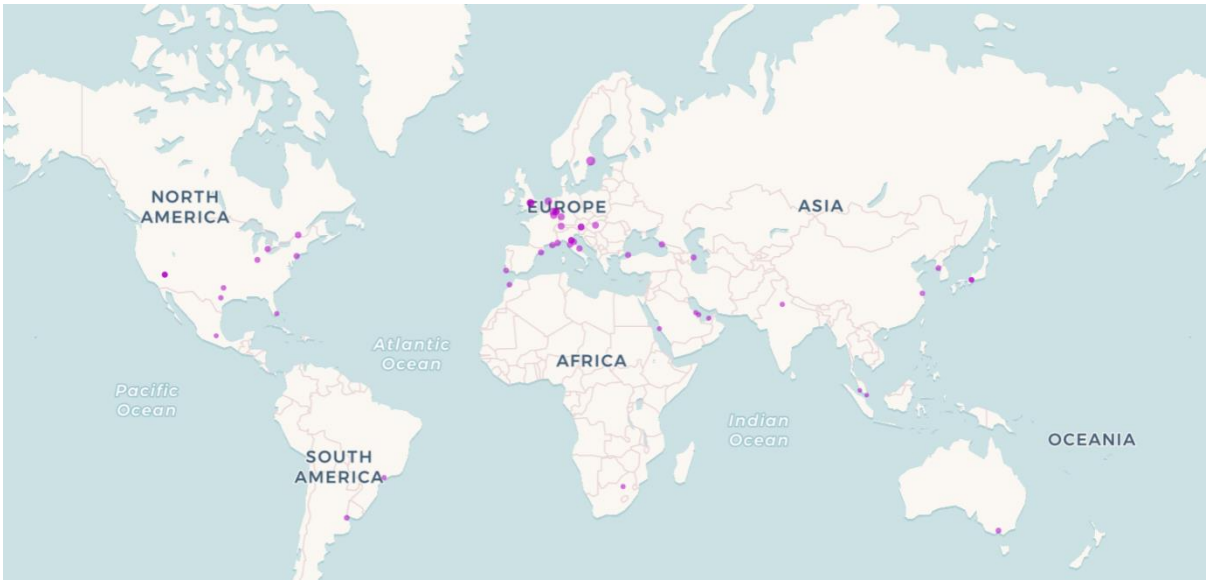
Para esta pregunta utilizamos el archivo fastest_laps_clean.csv y drivers_clean.csv.

Para plasmar en un mapa los circuitos primero se seleccionan las columnas Grand Prix, Latitude y Longitude del dataframe fastest_laps.csv y con pydeck se crea una capa de puntos donde cada punto es un circuito.

Para los pilotos primero se agrupan por Nacionalidad Latitud y Longitud (Columna Nationality, Latitude, Longitude) y se agregan los nombres de los pilotos y la cantidad de pilotos en otras columnas.

Resultado

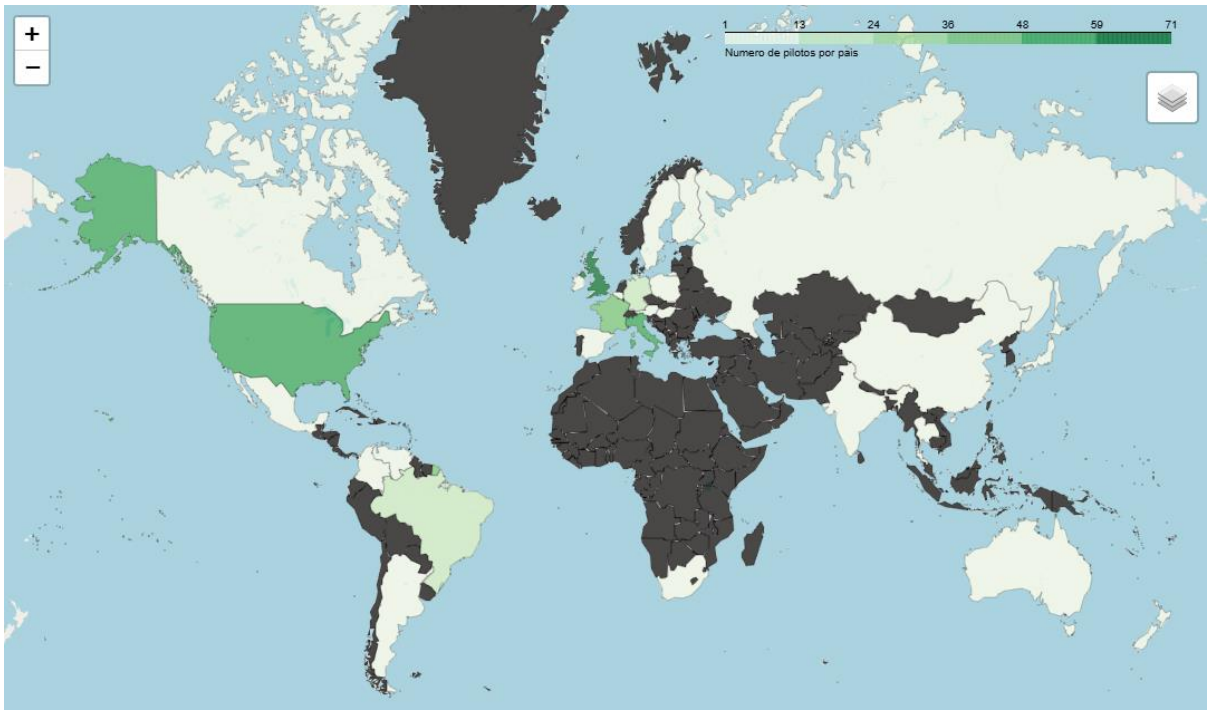
Circuitos



Se utiliza la librería PyDeck para crear un mapa interactivo que visualiza la ubicación de diferentes circuitos de carreras de Fórmula 1. El mapa se genera a partir de un DataFrame que contiene información sobre los Grandes Premios, incluyendo sus latitudes y longitudes. Primero, se seleccionan las columnas relevantes y se eliminan duplicados y valores nulos para garantizar la calidad de los datos.

Luego, se utiliza PyDeck para crear una capa de dispersión (ScatterplotLayer) que muestra los circuitos en el mapa con puntos de tamaño y color específicos, indicando sus ubicaciones geográficas. El mapa se centra en una vista global con la posibilidad de ajustar el nivel de zoom y estilo del mapa (como claro, oscuro, satélite, o carretera) a través de una selección interactiva en la barra lateral. Finalmente, el mapa se guarda en un archivo HTML, permitiendo su visualización y exploración interactiva en un navegador web. Este mapa facilita la identificación y análisis de la distribución global de los circuitos de carreras de Fórmula 1.

Pilotos



Se utilizo la librería Folium para crear un mapa coropléticos interactivo que visualiza la distribución de pilotos de carreras según su país de origen. Este tipo de mapa, conocido como "choropleth", colorea las regiones geográficas basándose en valores numéricos asociados, en este caso, el número de pilotos por país.

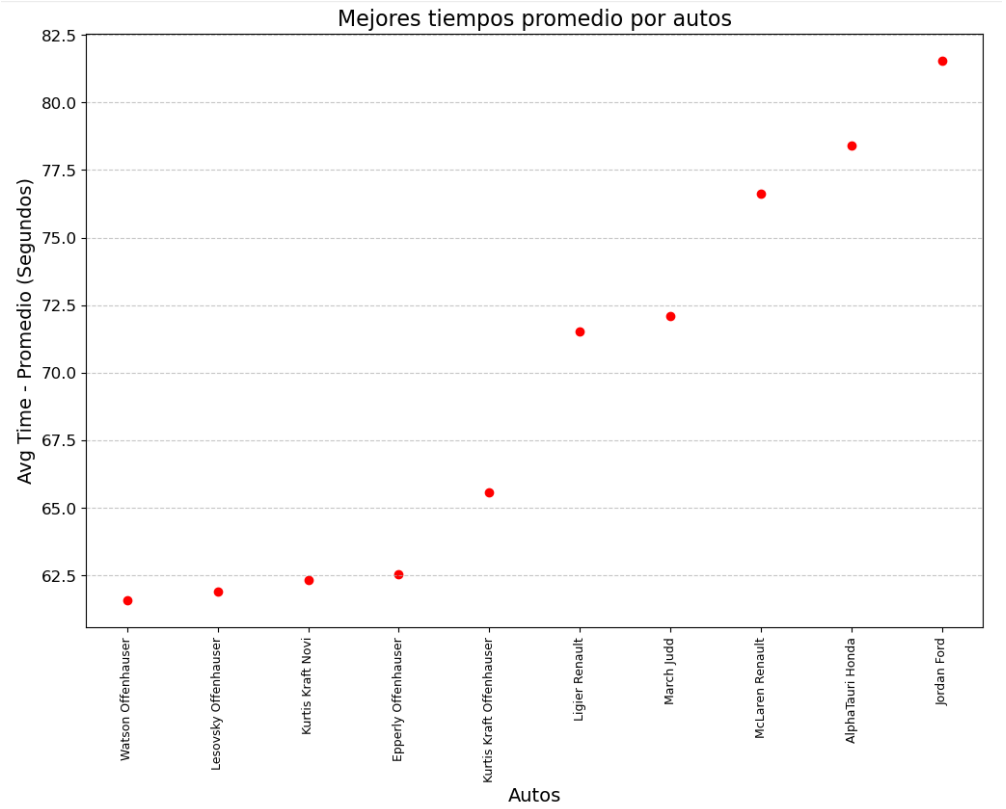
Primero, se agrupan los datos de pilotos por nacionalidad, latitud y longitud, contando el número de pilotos únicos y concatenando sus nombres. Luego, se carga un archivo GeoJSON que contiene los límites geográficos de los países. Usando Folium, se crea un mapa base centrado en el mundo y se añade una capa choropleth que colorea los países según el número de pilotos que tienen.

7. ¿Cuáles son los autos con los mejores tiempos? TOP 10

Para esta pregunta utilizamos el archivo fastest_laps.csv.

En esta pregunta calculamos los promedios de los tiempos según los autos (Columna Car), ordenamos de manera ascendente y tomamos los primeros 10.

Resultado

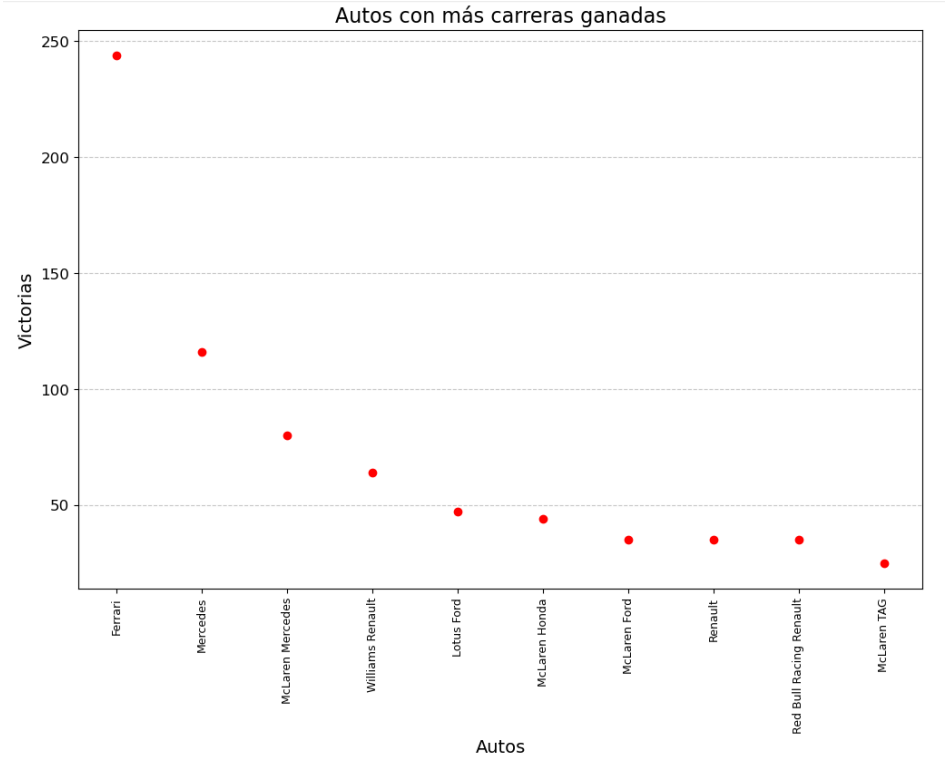


8. ¿Cuáles son los autos con más carreras ganadas? TOP 10

Para esta pregunta utilizamos el archivo winners.csv.

Se hizo un conteo de los registros por auto (Columna Car) y se tomaron 10.

Resultado



9. ¿Cuál es el promedio de mejores tiempos de pilotos ganadores por año?

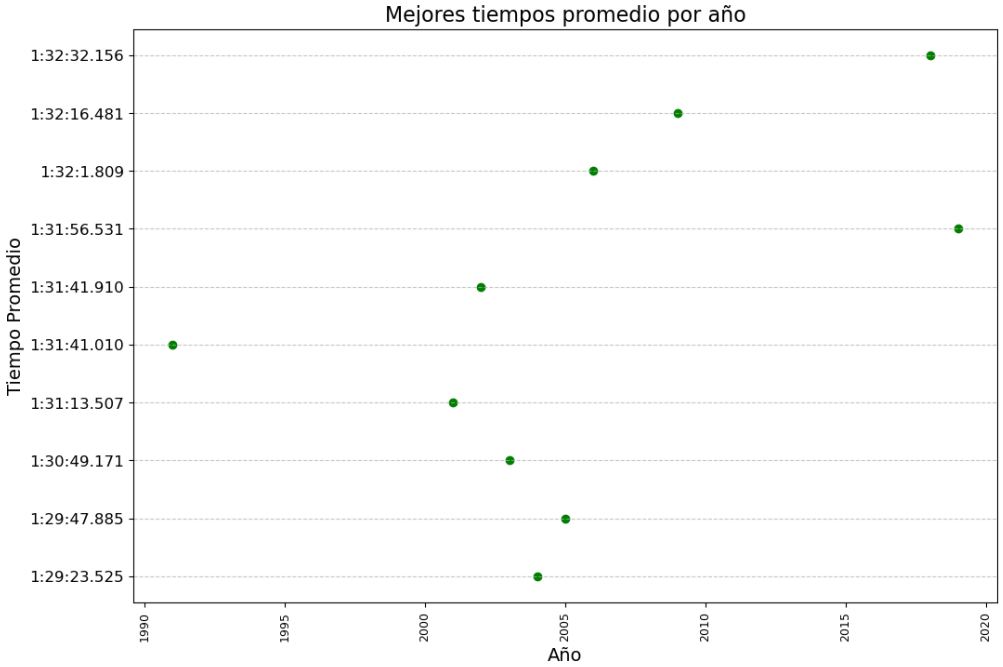
Para esta pregunta utilizaremos el archivo winners_clean.csv.

En este caso se realizó una copia del dataframe de winners sobre el que se agregaron dos columnas Date y Year, sobre Date se asignó el formato DateTime y en Year se guarda la componente del año de la columna Date.

Se agrega también el tiempo en segundos con las funciones mencionadas anteriormente y se calculan los promedios.

Luego se agrupa por año (Columna Year) el tiempo en segundos

Resultado



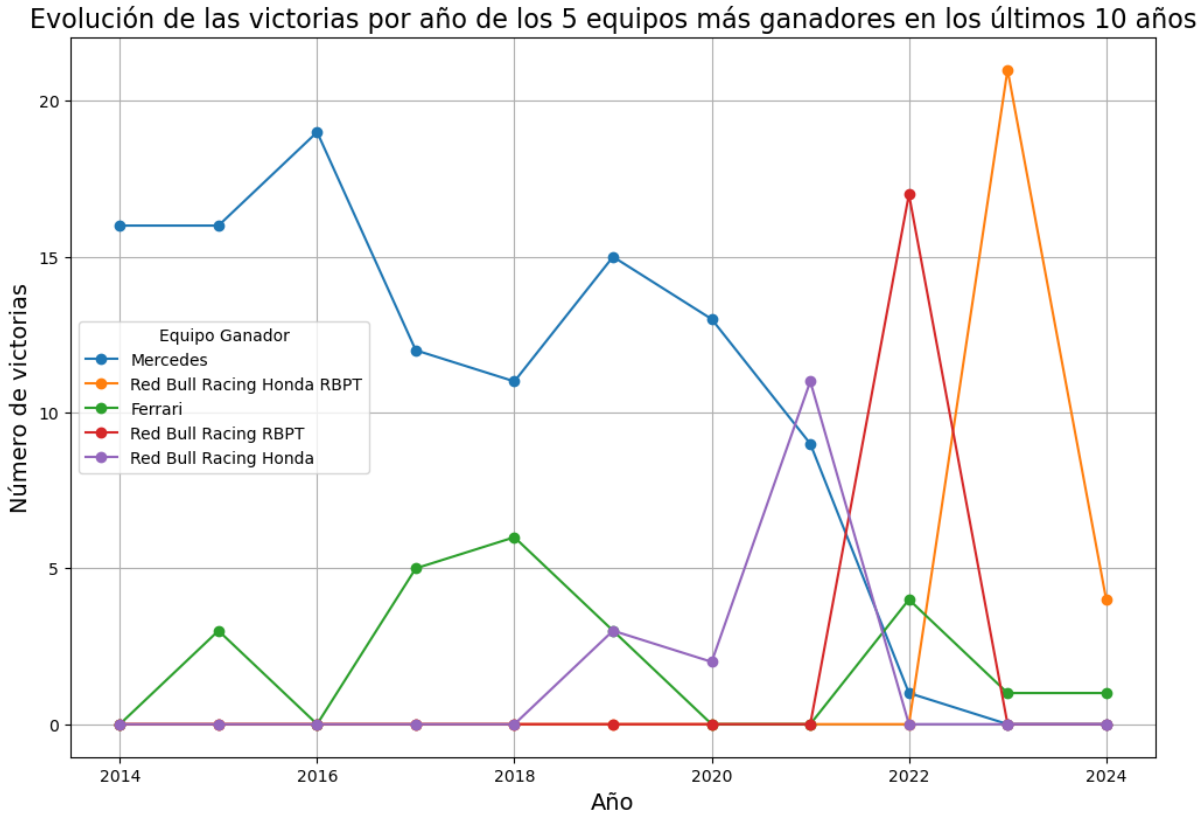
10. ¿Como es la evolución de victorias de los 5 equipos más ganadores de los últimos 10 años ?

Para esta consulta se utiliza el archivo winners_clean.csv

Se realiza una copia del dataframe original obtenido del archivo, al cual se le agrega una columna con el año. Una vez tengo este dato, lo uso para filtrar registros de los últimos 10 años.

Con este filtro, se calcula el promedio de los tiempos por año de cada circuito, y se grafica.

Resultado



Dashboard

Para la creación de un dashboard se ha seleccionado Tableau Public. Esta herramienta es ideal para representar datos de archivos CSV debido a su capacidad para crear visualizaciones interactivas y atractivas. Es fácil de usar, no requiere experiencia avanzada, ofrece una amplia variedad de visualizaciones, permite compartir fácilmente los resultados y es gratuito. Estas características lo hacen perfecto para transformar datos crudos en insights valiosos de manera eficiente.

Existen otras herramientas vistas en el curso tales como Superset (que se puede ejecutar localmente) o Google Looker Studio (herramienta en la nube).

Consultas y representación de datos

Para poder manipular los datos con facilidad en la herramienta Tableu consideramos de utilidad separar los datos procesados para poder presentar gráficamente algunas de las preguntas plasmadas.

En cada bloque de función que particularmente se necesite guardar se genera un archivo csv particular con un nombre representativo. Se agrega dentro de **OutPut_Data/Tableu_Public** se guardan los archivos.

Las preguntas formuladas para poder resolver fueron las siguientes :

- ¿Cómo se pueden listar los circuitos y los pilotos en un mapa? Se aplico filtro por continentes para ambas representaciones de mapas.
    - **Pilotos:** Mapa de calor
    - **Circuitos:** Ubicación de puntos dispersos.
    - **Acciones extras:** Se agrego un nuevo archivo continentes.csv que se utiliza en Tableau para lograr el filtrado por continente haciendo la conexión sobre el código ISO 3166-1 alfa -3 del país.
- ¿Como es la evolución de victorias de los 5 equipos más ganadores de los últimos 10 años? Se aplico filtro por rango de fechas por años.
- ¿Cuáles son las escuderías que ganan con mayor frecuencia en carreras en Europa? ¿ Cuáles son sus puntajes máximos ? Se aplico filtrado por los equipos presentes hasta la fecha.

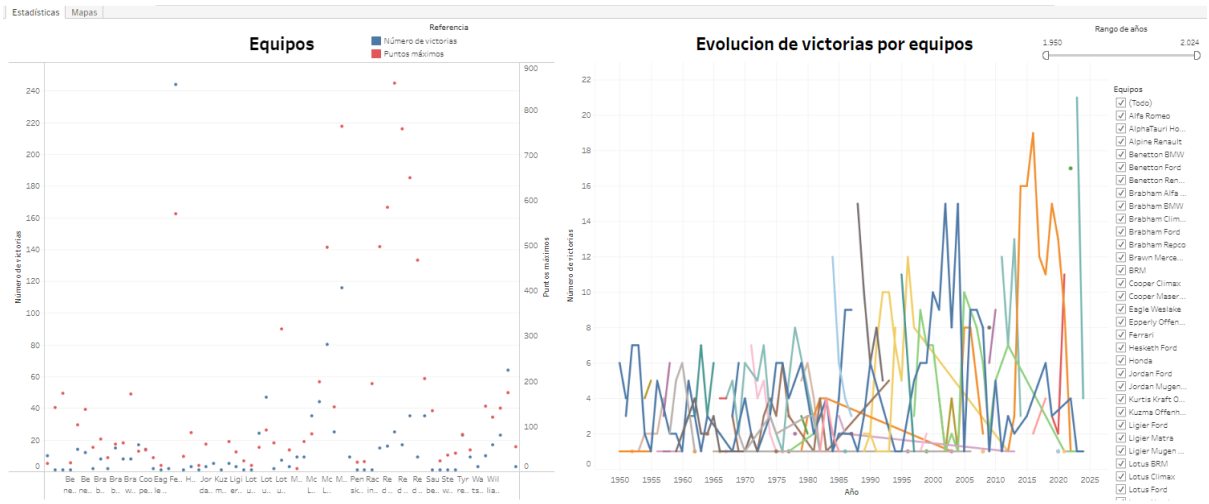
En el siguiente link se puede acceder a los dashboard creados por medio de la herramienta :

https://public.tableau.com/app/profile/santiago.gonzalez8207/viz/Dashboard-Obligatorio-28300021211421265 5/Dashboard1

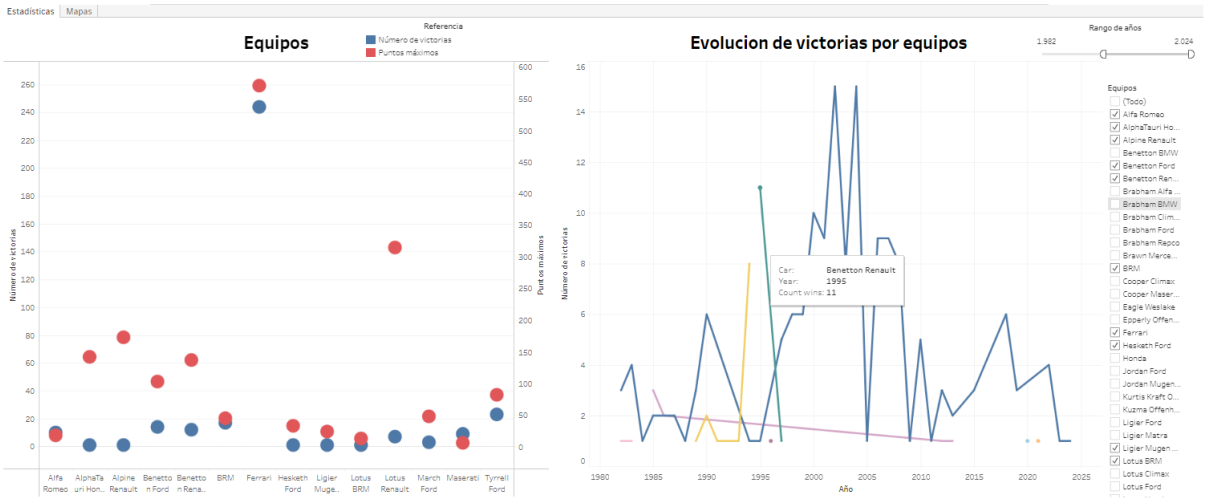
Existen dos dashboards, que se describen a continuación :

- **Estadísticas – Evolución de victorias por equipo y puntajes máximos**
    - Hay presentes dos inputs para filtrar un radio botón para variar entre años (1950 al 2024) y un check box para seleccionar el Teams que se requiere visualizar , ambos inputs se aplican a los dos gráficos presentes en la vista.

    - Ejemplos de filtrado:
        - **ALL - Todos**

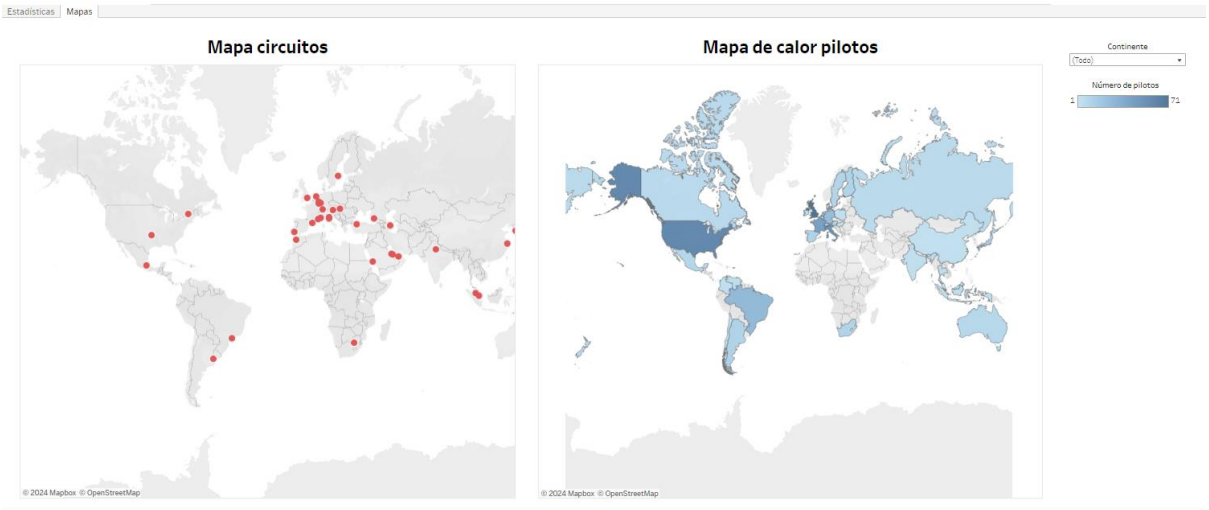


        - **Filtrado por tiempo 1982 al 2024 y la selección de 14 teams**

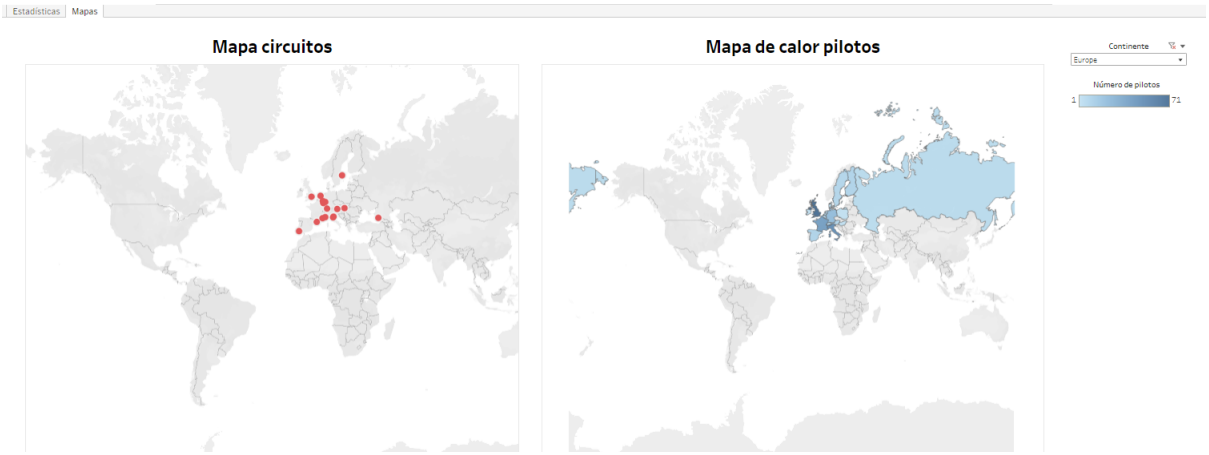


- **Mapas - Circuitos y Pilotos**
    - Hay presente un input de filtrado una lista desplegable de los continentes presentes que permite aplicar filtro a la vista de ambos mapas. En el mapa de los pilotos muestra la cantidad que hay en cada país y aumenta la intensidad del color si existe mayor cantidad.

    - Ejemplos de filtrado:
        - **ALL - Todos**



        - **Europa**



Configuraciones extras

Para poder manipular los datos en la plataforma Tableu, fue necesario realizar agregación y/o modificaciones fuera de lo previsto. A continuación, se enumera y se realiza una breve descripción de lo realizado.

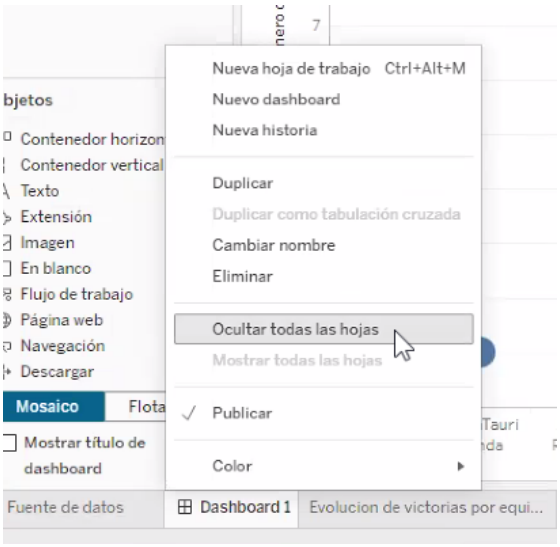
- **Conexión de los datos**

Desde la herramienta Tableu se puede vincular los datos por medios de conexiones, estas se pueden vincular por aquellas columnas que se definan con igual valor y la herramienta permite realizar de forma automática un join entre los datos a.

Se realizaron conexiones entre los archivos en las fuentes de datos de la siguiente manera:

- **Mostrar solo Dashboard**

Para ocultar las hojas de la presentación al publicar en Tableo es necesario ir sobre el Dashboard y seleccionar la opción “Ocultar todas las hojas” , se aplica sobre las hojas que utiliza esa vista.



Modelado de datos

La gestión y análisis de grandes volúmenes de datos, como los provenientes de las carreras de Fórmula 1, requiere de una estructura robusta y eficiente. El uso de Datasets históricos de Fórmula 1, que incluyen ganadores, conductores, vueltas rápidas y equipos, ofrece una oportunidad valiosa para aplicar técnicas de modelado de datos. Dentro de estas técnicas, el módulo normalizado se presenta como una opción clave para organizar y optimizar el acceso a estos datos Nuestra fuente de datos es simple y tiene relaciones bien definidas entre archivos (tablas) por lo que hacer joins no sería complejo, además tiene una gran capacidad de análisis, así que podría resultar útil este tipo de modelado.

El módulo normalizado, una metodología propuesta por Bill Inmon, se centra en replicar la fuente de datos original manteniendo la normalización. Este enfoque implica estructurar los datos en tablas relacionadas mediante claves primarias y foráneas, lo que minimiza la redundancia y asegura la integridad de los datos. La normalización es crucial para garantizar que los datos sean precisos y consistentes, facilitando su mantenimiento y actualizaciones.

La elección del módulo normalizado para los Datasets históricos de la Fórmula 1 se debe a varias razones. Primero, la naturaleza detallada y rica en relaciones de los datos de Fórmula 1 (como la conexión entre conductores, equipos y resultados de carreras) se adapta perfectamente a un modelo normalizado. Este modelo facilita la ejecución de análisis precisos y detallados, permitiendo explorar relaciones y tendencias a lo largo del tiempo sin pérdida de información. Segundo, la normalización ayuda a mantener la integridad y calidad de los datos, cruciales para análisis predictivos y estudios estadísticos. Finalmente, el uso de un modelo normalizado permite integrar y enriquecer estos datos con otras fuentes, creando una base sólida y escalable para futuros análisis y aplicaciones en el ámbito del deporte y la ingeniería.

Modelado en HIVE

Hive es una plataforma de almacenamiento de datos que facilita el análisis de grandes conjuntos de datos en un entorno Hadoop. Utiliza un lenguaje similar a SQL, llamado HiveQL, para permitir consultas y análisis sin necesidad de conocimientos avanzados de programación. Hive traduce estas consultas en trabajos MapReduce, Spark o Tez, optimizando el procesamiento de datos masivos. Su importancia en big data radica en su capacidad para gestionar y analizar petabytes de datos de manera eficiente, permitiendo a las organizaciones extraer información valiosa y tomar decisiones informadas a partir de grandes volúmenes de datos.

Creación de tablas

En base a lo visto en clase se definen la base de datos prevista y las tablas relacionadas al dominio a describir.

Se han tomado algunas consideraciones generales en los datos Team , Winners , Fast Test Laps y Drivers , se repite el atributo Car, consideramos que es común o sinónimo de Team , permite hacer una JOIN entre tables , se entiende que sería similar a una clase foránea. Entonces aquellos atributos con definición “car” pasar a ser “team”

```
CREATE DATABASE IF NOT EXISTS historiaF1;

USE historiaF1;

CREATE EXTERNAL TABLE winners (
    grand_prix STRING,
    date DATE,
    winner STRING,
    team STRING,
    laps INT,
    time STRING,
    name_code STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS CSV
LOCATION ‘/rfn/hive/input_data/winners/’;
```

```
CREATE EXTERNAL TABLE teams (
    pos INT,
    car STRING,
    pts INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS CSV
LOCATION ‘/rfn/hive/input_data/teams/’;

CREATE EXTERNAL TABLE fastest_laps (
    grand_prix STRING,
    driver STRING,
    team STRING,
    time STRING,
    name_code STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS CSV
LOCATION ‘/rfn/hive/archivos_hive/fastest_laps/’;

CREATE EXTERNAL TABLE drivers (
    pos INT,
    driver STRING,
    nationality STRING,
    team STRING,
    pts INT,
    name_code STRING
```

```
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS CSV
LOCATION ‘/rfn/hive/archivos_hive/drivers/’;

CREATE EXTERNAL TABLE grand_prix_locations (
   grand_prix STRING,
   country_code STRING,
   latitude DOUBLE,
   longitude DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS CSV
LOCATION ‘/rfn/hive/input_data/grand_prix_locations/’;

CREATE EXTERNAL TABLE country_locations (
   nationality STRING,
   latitude DOUBLE,
   longitude DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS CSV
LOCATION ‘/rfn/hive/archivos_hive/country_locations/’;

LOAD DATA INPATH '/input_data /winners.csv' OVERWRITE INTO TABLE winners;
LOAD DATA INPATH '/input_data/teams.csv' OVERWRITE INTO TABLE teams;
```

LOAD DATA INPATH '/input_data / fastest_laps.csv' OVERWRITE INTO TABLE fastest_laps;

LOAD DATA INPATH '/input_data/ drivers.csv' OVERWRITE INTO TABLE drivers;

LOAD DATA INPATH '/input_data /grand_prix_locations.csv' OVERWRITE INTO TABLE grand_prix_locations;

LOAD DATA INPATH '/input_data/country_locations.csv' OVERWRITE INTO TABLE country_locations;